ABSTRACT

Title of Dissertation: EVALUATION OF SELECTED SIDE-CHANNEL ANALYSIS METHODS FOR RANSOMWARE CLASSIFICATION AND DETECTION

Jennie E. Hill
Doctor of Philosophy, 2023

Dissertation Directed by: Professor Martin C. Peckerar
Department of Electrical and Computer Engineering

The physical implementation of computer hardware leads necessarily to physical behavior on the part of an operating computer. This physical behavior has physical characteristics, many of which become channels of information leakage that can be observed by an unintended receiver. This poses a serious threat to computer security. These "side-channels" of computer operations, such as current usage and power consumption, generation of heat and electromagnetic radiation, and events at the micro-architectural level, can be exploited to compromise the confidentiality of a system.

This work considers side-channel analysis techniques for the temperature, power, and micro-architectural side-channels for the purpose of classifying state-of-the-art ransomware on real world, non-virtualized Windows systems. Over three thousand ransomware and benign trials were collected to generate training and testing data sets, which required development of a process to synchronize collection of on-system (e.g. performance counters) and off-system (e.g. power)

measurements, safely transfer trial data from the encrypted system, and restore the system to a "clean" state without the use of virtualization techniques, which negatively impact the validity of side-channel measurements. Side-channels were evaluated on their effectiveness in accurately differentiating between ransomware and benign operations such as background operating system activity, 7zip encryption, and SPEC benchmarks, in a given time duration, with Matthews' Correlation Coefficient (MCC) used to measure overall classifier performance of five machine learning classification algorithms.

The temperature side-channel, accessed through thermal imaging, was found to be unsuitable for the ransomware detection/classification application due to its sensitivity to thermal noise, significant pre-processing requirements, and slow response times due to the loss of signal components above the low kHz range. This limited its ability to identify ransomware before encryption operations typically begin (within 2 seconds of execution, on average). The power side-channel, accessed by monitoring the current drawn by a solid state drive, generated best-case classification accuracy results of 96% (0.92 MCC) with 15 seconds of current data and $\geq$ 90% (MCC ≥ 0.8) for all five classifiers tested with at least 5 seconds of data. Tests demonstrated at least four seconds of data were required to attain a best case classification accuracy greater than 90%, and at 2 seconds the best-performing classifier attained an MCC of just 0.66 with 83.3% accuracy.

The micro-architectural side-channel was accessed through hardware performance counters, which provided the highest MCC and accuracy results in the shortest period of time. Hardware performance counters are registers built into a CPU's Performance Monitoring Unit, and measure events related to processor and memory system operations (e.g. CPU clock cycles, total instructions retired, memory accesses, cache hits/misses, branches taken, etc.). Over 230 hardware events were collected, tested, and ranked by their contribution to overall classifier performance. Each

classification algorithm was found to have a distinct performance counter feature ranking, and the selected features could be further optimized by desired detection window duration. Examination of results showed that, despite the quantity of features collected, classifier performance only marginally improved after 6 features for $\leq$ 2 seconds, with MCC $\geq$ 0.9 for 1 second of data for 3 of 5 classifiers tested with just 4-6 performance counter features, and a best-case MCC of 0.98 with 1 second of data and 4 performance counter features. MCC results for the shortest duration event window (0.1s) were found to be within 0-7% of the best case MCC result window (1-2 s) for each classifier, indicating that ransomware can be classified using only a tenth of a second of 4-6 performance event measurements with greater than 90% accuracy for four of five classifiers tested, which makes the implementation of this approach in a real-time ransomware detector feasible. With the financial impact of ransomware estimated to cost more than $30 billion globally this year, the usefulness of new detection techniques for non-virtualized computer systems has significant real-world implications.

EVALUATION OF SELECTED SIDE-CHANNEL ANALYSIS METHODS FOR RANSOMWARE CLASSIFICATION AND DETECTION

by

Jennie Elizabeth Hill

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2023

Advisory Committee:
Professor Martin C. Peckerar, Chair/Advisor
Professor Bruce L. Jacob, Co-Advisor
Professor Ankur Srivastava
Professor T. Owens Walker, III
Professor Donald Yeung
Professor Amitabh Varshney, Dean's Representative

Acknowledgements

This was absolutely not a solo effort. Many people deserve my sincere gratitude; I could not possibly capture them all... So here are some.

My advisors: Dr. Jacob and Dr. Peckarar, for being available to answer my texts at all hours and for pushing the pace when I was on the verge of procrastinating. Your guidance and mentorship have been both insightful and essential in helping me to get to this point.

My "CECSR" (Computer Engineering and Cyber Security Research) team - James, Justin, Dane, Rob, Owens, Ryan, Hau - all of whom were very generous, *extremely* patient mentors: For teaching me, letting me borrow equipment that I may or may not have returned, and reminding me that research is a journey rather than a destination. Your bald-faced humor, insistence on the insertion of randomly-chosen words (e.g. "bald") into papers, and ability to not take yourselves too seriously made the long days in the lab tolerable. Special thanks to ENS Brendan Farmer for far outperforming all expectations as a lab assistant, and to my small-but-mighty UMD crew of Devesh and Ananth for the selfless gifts of their time and guidance.

My Naval Academy colleagues, particularly: Jeff, for his technical expertise, critical lab support, and many years of friendship. Mike and Chris, for numerous hours of EI, well outside the scope of their normal duties. Hatem, Ethan, and Ann, for their unwavering support and encouragement. The ECE Department and PMP program leadership, for their support in making this opportunity a possibility in the first place, commitment to finding a program that was a realistic fit, and then ensuring I had the resources to finish the job.

Nick, Jon, and Sam: For adapting to living through a global pandemic in a single-parent household where mom turned the dining room into a lab for her PhD research, and for all the additional responsibilities they assumed in the process (and especially to Sam for his many hours of company when we finally had a real lab to go to). Together, they made sure I never got too much sleep or quiet time, and provided me with an endless supply of stories to tell my research group when I should have been working.

My family: My parents for making multiple trips to Annapolis to assist with cooking/driving/childcare, my mom for doubling as a copy-editor, my brothers for their consistent phone support availability, and to Amanda for being like-a-sister for longer than either of us cares to admit (unless we met when we were 4 or younger).

My CW crew, QuaranTEAM (Alana, Sara, Chad), and friends: Cindy, Ora, Meredith, Kristin, & Kristen, to name just a few. If you ever stumble across this and think maybe you should be part of this list, you're right. Thank you all, seriously, for your support. I really couldn't have done it without you. And adderall. It was prescribed. But mainly you.

Last and least, to generative AI for producing a solid Acknowledgements section draft that incorporated humor and sarcasm. And for the warning to not use it.

Table of Contents

Acknowledgements ii

Table of Contents iii

List of Tables vii

List of Figures viii

List of Abbreviations xviii

Chapter 1: Introduction 1
- 1.1 A Brief History of Side-channels . . . 1
  - 1.1.1 Considering Side-channels in the framework of Electronic Warfare . . . 3
- 1.2 Side-channel Analysis to detect Ransomware . . . 5
- 1.3 Contributions . . . 8
- 1.4 Organization . . . 9

Chapter 2: Side-channels 10
- 2.1 Side-channels . . . 10
  - 2.1.1 Power Side-channel . . . 11
  - 2.1.2 Electromagnetic Side-channel . . . 11
  - 2.1.3 Acoustic Side-channel . . . 12
  - 2.1.4 Optical Side-channel . . . 12
  - 2.1.5 Temperature Side-channel . . . 12
  - 2.1.6 Micro-architectural Side-channel . . . 13
- 2.2 Side-channel Attack, Analysis, and Defense . . . 13
- 2.3 Side-channel Metrics . . . 15
  - 2.3.1 Statistical Tests . . . 15
  - 2.3.2 Entropy, Conditional Entropy, & Guessing Entropy . . . 16
  - 2.3.3 Mutual Information . . . 17
  - 2.3.4 Success Rate . . . 17
  - 2.3.5 Welch's T-Test . . . 18
  - 2.3.6 Side-channel Vulnerability Factor (SVF) . . . 18
  - 2.3.7 Spatial Thermal Side-channel Factor (STSF) . . . 18
  - 2.3.8 Cache Side-channel Vulnerability (CSV) . . . 19
  - 2.3.9 Signal Available to the Attacker (SAVAT) . . . 19

2.3.10 Thermal-Security-in-Multi-Processors (TSMP) . . . 19
2.3.11 Maximal Leakage . . . 19
2.3.12 Information Leakage Rate . . . 20
2.3.13 Local Differential Privacy . . . 20
2.3.14 Trust Coverage . . . 20
2.3.15 Holistic Assessment Criterion . . . 21
2.3.16 Machine Learning Metrics . . . 21

Chapter 3: The Temperature Side-channel **22**
3.1 Overview . . . 22
3.2 Characteristics of the Temperature Side-channel . . . 22
3.3 Sensing the Thermal Channel . . . 23
3.3.1 Internal Temperature Sensing Methods . . . 24
3.3.2 External Temperature Sensing Methods . . . 25
3.4 Temperature Attacks . . . 28
3.4.1 Temperature Side-channel Attacks . . . 28
3.4.2 Covert Channels . . . 29
3.4.3 Physical Attacks . . . 30
3.5 Temperature Side-channel Defense . . . 30
3.5.1 Design-time Defense . . . 31
3.5.2 Run-time Defense . . . 31

Chapter 4: The Micro-architectural Side-channel **33**
4.1 Micro-architectural Side-channel . . . 33
4.1.1 Timing Side-channel . . . 34
4.1.2 Memory (Access) Side-channel . . . 34
4.2 Hardware Performance Counters for the Micro-architectural Side-channel . . . 34
4.2.1 Profiling Tools . . . 35
4.2.2 Intel®VTune™ Profiler Primer . . . 36
4.3 Considerations When Using Performance Counters for Security Applications . . . 39

Chapter 5: Malware and Ransomware **42**
5.1 Malware Overview . . . 42
5.1.1 Ransomware . . . 44
5.2 Malware Detection via Side-Channel Analysis . . . 45
5.2.1 Malware Detection with the Power Side-channel . . . 45
5.2.2 Malware Detection with the EM Side-channel . . . 45
5.2.3 Micro-architectural Side-Channel for Malware Detection (via Hardware Performance Counters) . . . 46
5.2.4 HPC-based Machine Learning Classification . . . 48
5.3 Ransomware Detection . . . 49
5.3.1 Ransomware Detection with Hardware Sensors and Performance Counters 50

Chapter 6: Temperature Side-channel Analysis Experiments **54**

6.1 TSCA Study 1: Temperature Side-channel Analysis to detect File Operations on an SSD with Thermal Imaging . . . 54
6.1.1 Experimental Set-up . . . 54
6.1.2 Classification Results . . . 56
6.1.3 Conclusions and Future Work . . . 60
6.2 TSCA Study 2: Temperature Side-channel Analysis to detect Simulated Malware on a Single Board Computer . . . 61
6.2.1 TSCA Study 2 Methodology . . . 62
6.2.2 Thermal Signature of Write Operations . . . 66
6.3 TSCA Experiment Conclusions and Future Research Directions . . . 70

Chapter 7: Investigating Vtune HPCs for Ransomware Detection 72
7.1 Proof-of-Concept A: Vtune Data Collection of Ransomware Trials . . . 72
7.1.1 Hardware Setup . . . 72
7.1.2 Data Collection Procedure . . . 75
7.1.3 System Restore Process . . . 76
7.1.4 Initial Observations . . . 76
7.2 Proof-of-Concept B: Simultaneous Current and Vtune Data Collection of ransomware in Non-virtualized and Virtualized Environments . . . 78
7.2.1 Non-Virtualized Hardware Setup . . . 79
7.2.2 Virtualized System Set-up . . . 81
7.2.3 Data Collection Procedure . . . 83
7.2.4 Vtune Performance Counter Analysis . . . 84
7.2.5 Proof-of-Concept B Observations . . . 84
7.2.6 Drive Restore Procedure Using Image for Linux . . . 85
7.3 Study 1: Automate, Expedite, and Expand Vtune collection . . . 86
7.3.1 Data Collection Procedure . . . 86
7.3.2 Experiment Results . . . 90
7.4 Study 2: Expanded Vtune HPC Collection with Additional Ransomware Samples 90
7.4.1 Experiment Results . . . 91
7.5 Analysis of Performance Counters for Classification of Ransomware Op-erations for Studies 1 (23-class) and 2 (31-class) . . . 92
7.5.1 Feature Generation . . . 93
7.5.2 Classification Algorithm Identification . . . 94
7.5.3 HPC Feature Selection . . . 100
7.5.4 Single-HPC F1-based Feature Reduction . . . 118
7.6 Summary . . . 124

Chapter 8: Selecting Vtune HPCs for Ransomware Detection 126
8.1 Study 3: 35-class, 90 HPC Feature Collection with Networked Database transfer and Parallel Report Generation . . . 126
8.1.1 Training Data Verification . . . 127
8.2 Training Data Validation . . . 130
8.2.1 Matthews' Correlation Coefficient (MCC) . . . 131
8.2.2 HPC Feature Selection Process . . . 133

8.2.3 Cross-Validation of Training Data . . . 136
8.2.4 Classifier Training . . . 139
8.3 Classifier Evaluation Results . . . 142
8.3.1 Test Data Set . . . 142
8.4 Towards Start-time Ransomware Detection . . . 148
8.4.1 Identification of Operation Start Time . . . 150
8.4.2 Cross Validation and Testing Results . . . 151
8.4.3 Results . . . 153
8.5 Top Performance Counter Feature Comparison . . . 158
8.5.1 25 second Start Time Feature Set . . . 158
8.5.2 Performance Counter Ranking Discussion . . . 159
8.6 MASC Experiments Summary and Future Research Directions . . . 159

Chapter 9: Current-based Power Side-channel Analysis Experiments 167
9.1 Proof of Concept: Power Side-channel Analysis via Current Measurements for Detection of SSD File Operations . . . 167
9.1.1 Method . . . 168
9.1.2 Results . . . 172
9.1.3 Follow-on Power Side-channel Analysis Work . . . 176
9.2 Study: Classification of Read, Write, and Idle Operations with Current-based Power Side-channel Analysis . . . 177
9.2.1 Experimental Setup . . . 177
9.2.2 Analysis . . . 179
9.2.3 Results . . . 183
9.3 Proof of Concept: Accessing the Power Side-channel via Current Draw for Ransomware Identification . . . 188
9.3.1 Method . . . 188
9.3.2 Current Side-Channel for Ransomware Detection . . . 196
9.4 Study: 35-class Current-draw-based Side-channel Analysis for Ransomware Classification 197
9.4.1 Part 1: Power Feature Generation and Classifier Identification . . . 199
9.4.2 Part 2: Current-based Power side-channel Analysis for Ransomware vs. Benign Operations . . . 201
9.4.3 Study 2 Results and Initial Conclusions . . . 212
9.5 PSCA Experiment Summary and Future Research Directions . . . 223

Chapter 10: Conclusions and Future Work 227
10.1 Future Work . . . 229

Appendix A: Table of Vtune Hardware Performance Counter Names, Descriptions, and associated Feature Numbers for each Study, adapted from [1] 232

Bibliography 258

List of Tables

6.1 Comparison between the thermal cameras used in experiments. 64

8.1 Accuracy, Precision, Recall, F1, and MCC metrics for the Confusion Matrix shown in Figure 8.7. 133

9.1 Solid State Drives Tested 169
9.2 Solid State Drives Tested 178
9.3 Current data from trials were converted to Power Spectral Density using Welch's Method with varying combinations of data duration, window size, and maximum frequency. 200
9.4 Current data from all training and testing trials were converted to Power Spectral Density using Welch's Method with varying combinations of data duration, window size, and maximum frequency. All power features were generated in both Watts and dBW. 204

List of Figures

1.1 Electromagnetic wave, reprinted from [2]. 3
1.2 Electromagnetic Spectrum, adapted from [3]. 4
1.3 Relationship between Electronic Warfare terminology and Side-Channel terminology. 6

6.1 Experimental set up to capture thermal images of SSDs with a FLIR A325sc thermal camera. 55
6.2 Thermal images of a Samsung 970 EVO Plus SSDs during idle, read, write, operations. All images were captured with a FLIR A325sc thermal camera. . . . 56
6.3 Thermal images of a Samsung 970 EVO Plus SSDs during idle, read, write, operations. All images were captured with a FLIR A325sc thermal camera. . . . 57
6.4 Consolidated Confusion Matrix showing the ability to differentiate activity (read/write) from non-activity (idle) for all SSD, operation, and file size combinations for the 3 classifiers trained and tested. 58
6.5 10-fold Cross Validation Results for 3 classifiers trained with read, write, and idle operations at file sizes of 100, 500, and 1000MB. Each row shows results for a single classifier at increasing read/write sizes. 59
6.6 Test Data Results for 3 classifiers trained with read, write, and idle operations at file sizes of 100, 500, and 1000MB. Each row shows results for a single classifier at increasing read/write sizes. 60
6.7 System diagram. 63
6.8 Measurement system setup. The green (upper) box in the left image indicates the thermal camera, while the white (lower) box highlights the single board computer. The right images show the top and side views of the camera and board set up. . . 63
6.9 Comparison of images captured with (a) FLIR A325sc, (b) Seek Thermal CompactPro, and (c) FLIR T530 cameras. FLIR image temperature scaled to 75-135 °F with FLIR Research Studio software. SeeK CompactPRO scaled to 75-135 °F with SeeK Thermal iPhone app. FLIR T530 scaled from 80-110 °F in FLIR Tools. Blue in SeeK image indicates temperatures $< 75°$. 65
6.10 File operations performed on a BeagleBone Black, as observed with a SeeK CompactPRO thermal camera. 66
6.11 Thermal signature of a 100 MB file write over time. The left image shows a BeagleBone Black with three components highlighted: (1) Power Management Integrated Chip, (2) Embedded Multi Media Card, and (3) Processor, all of which produce distinct heat signatures over the course of the write operation. 66

6.12 Accuracy of SeeK Compact Pro and FLIR A325sc cameras, using both automated and manual detection techniques. . . . 68
6.13 Precision of SeeK Compact Pro and FLIR A325sc cameras, using both automated and manual detection techniques. . . . 69
6.14 Recall of SeeK Compact Pro and FLIR A325sc cameras, using both automated and manual detection techniques. . . . 69

7.1 Data Collection Summary for Proof of Concept A . . . 73
7.2 Collection apparatus. Includes the data recorder and current probe/amplifier to measure current supplied to the drive, but those capabilities were not incorporated into the proof-of-concept Vtune collection. . . . 73
7.3 Data Collection and System Restore Procedure for manually executed 7zip and Sodinokibi randomized trials. . . . 76
7.4 Mean Event Rate for a 7zip encryption trial vs. REvil/Sodinokibi ransomware, from analysis of VTune microarchitectural hardware performance counters. . . . 77
7.5 Data Collection Summary for Proof of Concept B . . . 78
7.6 Experimental setup for monitoring power supplied to the test OS SSD. Current is measured with a current probe, amplified, and acquired and stored with the data recorder. Vtune microarchitectural hardware performance counters are collected from Vtune Profiler running on-system and then transferred off-system for additional analysis. . . . 79
7.7 Data Collection and System Restore Procedure for manually executed trials. . . . 83
7.8 Vtune Hardware Performance Counters collected solely in 20 of 20 non-idle trials. 84
7.9 Vtune Hardware Performance Counters collected solely in 15 of 15 non-idle trials. 85
7.10 Data Collection Summary for Study 1 . . . 86
7.11 SPECworkstation 3.1 Benchmarks incorporated into data collection. . . . 88
7.12 Data Collection and System Restore Procedure Study 1. . . . 89
7.13 Data Collection Summary for Study 2. . . . 90
7.14 Data Collection, System Restore, and Report Generation Procedure for 31-class data set (Study 2). . . . 92
7.15 Start Time - Window Size - Stop Time Combinations for Study 1. Each column indicates a time interval for which HPC feature sets were generated (12 total). . . . 93
7.16 Start Time - Window Size - Stop Time Combinations for Study 2. Each column indicates a time interval for which HPC feature sets were generated (12 total). . . . 94
7.17 MATLAB Classification Models Evaluated in Classification Learning Application, adapted from [4]. . . . 97
7.18 MATLAB Classification Learning Application Cross-Validation Accuracy for 23-class data set (Study 1) . . . 98
7.19 MATLAB Classification Learning Application Cross-Validation Accuracy for 31-class data set (Study 2) . . . 99
7.20 Five Classification Algorithms were selected based on their performance across both data sets. . . . 100

7.21 Weighted importance score calculation example for the 31-class (220 HPC) data set and the MRMR algorithm. Feature ranking rows 11-210 are hidden for the 31-operation classification ranking results, while feature rows 11-220 are hidden for the binary classification ranking results. On the right hand side, weighted importance scores are summed across each row for each HPC to get the total importance score for each HPC and ranking. Importance scores are then summed by column to get a weighted importance score for each HPC for all rankings and data sets. 103
7.22 Weighted importance scores by algorithm for the to 10 HPCs in the 31-class experiment, ranked from highest to lowest importance score value. 103
7.23 Single-HPC Feature Classification Accuracy for top 20 Features using Bagged Tree Ensemble Classification Algorithm for multi-class and binary versions of Studies 1 and 2 104
7.24 Single-HPC Feature Classification Accuracy for top 20 Features using Subspace Discriminant Ensemble Classification Algorithm for multi-class and binary versions of Studies 1 and 2 105
7.25 Cross Validation Accuracy for incremental performance counters in order from highest to lowest individual accuracy values for Bagged Tree Ensemble Classifier. 23-class data sets are shown in the top two plots, with 31-class data sets in the bottom two plots. The multi-class version of each data set is on the left, while the binary version of each data set is on the right. 106
7.26 Cross Validation Accuracy for incremental performance counters in order from highest to lowest individual accuracy values for Subspace Discriminant Ensemble Classifier. 23-class data sets are shown in the top two plots, with 31-class data sets in the bottom two plots. The multi-class version of each data set is on the left, while the binary version of each data set is on the right. 107
7.27 Comparison of Bagged Tree Classifier Cross Validation Accuracy for incremental performance counters in order from highest to lowest individual accuracy values for windows of 0.1 s, 0.5 s, 1 s, and 2 s before, at, and after operation start time. 23-class multi-class data sets are shown across the top, with 31-class multi-class data sets across the bottom. 108
7.28 Comparison of Bagged Tree Classifier Cross Validation Accuracy for incremental performance counters in order from highest to lowest individual accuracy values for windows of 0.1 s, 0.5 s, 1 s, and 2 s before, at, and after operation start time. 23-class BINARY data sets are shown across the top, with 31-class BINARY data sets across the bottom. 109
7.29 A Confusion Matrix is a standard way to visualize supervised machine learning outcomes. False Negatives (FN) are placed in Quadrant I, True Positives (TP) are placed in Quadrant II, False Positives (FP) are placed in Quadrant III, and True Negatives (TN) are in Quadrant IV 111
7.30 Example "worst case" Confusion Matrix for single-HPC classification tests. . . . 112
7.31 Best single-HPC Confusion Matrix resulting from training 5 classification algorithms with a single HPC feature at a time. This Confusion Matrix shows the cross validation results of a Fine KNN classifier trained with 2 seconds of data and the single HPC feature BR_INST_RETIRED.NEAR_RETURN_PS. 113

7.32 F1 scores for each HPC feature and start/stop time window were calculated and then averaged across all start/stop time windows to obtain an average F1 score for HPC feature ranking purposes. 114
7.33 Cross validation accuracy, precision, recall, and F1 plots for binary 23-class (Study 1) and 31-class (Study 2) data sets trained with Bagged Tree Ensemble classifier and incrementing the number of HPCs used for training. HPCs were ranked from high to low by F1 score averaged over start/stop time window sizes. . 115
7.34 Cross validation accuracy, precision, recall, and F1 plots for binary 23-class (Study 1) and 31-class (Study 2) data sets trained with Subspace Discriminant Ensemble classifier and incrementing the number of HPCs used for training. HPCs were ranked from high to low by F1 score averaged over start/stop time window sizes. 115
7.35 Cross validation accuracy, precision, recall, and F1 plots for binary 23-class (Study 1) and 31-class (Study 2) data sets trained with Linear Support Vector Machine classifier and incrementing the number of HPCs used for training. HPCs were ranked from high to low by F1 score averaged over start/stop time window sizes. 116
7.36 Cross validation accuracy, precision, recall, and F1 plots for binary 23-class (Study 1) and 31-class (Study 2) data sets trained with Fine K-Nearest Neighbor classifier and incrementing the number of HPCs used for training. HPCs were ranked from high to low by F1 score averaged over start/stop time window sizes. . 116
7.37 Cross validation accuracy, precision, recall, and F1 plots for binary 23-class (Study 1) and 31-class (Study 2) data sets trained with Narrow Neural Network classifier and incrementing the number of HPCs used for training. HPCs were ranked from high to low by F1 score averaged over start/stop time window sizes. . 117
7.38 Top 20 F1-ranked HPCs for each classification algorithm. Study 1 (23-class) HPC features are on the left column and Study 2 (31-class) HPC features are on the right. Corresponding feature numbers from the opposing study are listed in gray next to the HPC number for each study. 117
7.39 Top F1-ranked HPC for each classification algorithm for Study 2 (31-class data set) only. These were used as a starting point to identify the fewest number of HPC features to generate the highest possible F1 score. 119
7.40 HPC Features selected for each classifier to achieve the highest possible F1 score with the fewest number of features, ranked by F1 score. 119
7.41 Cross Validation F1 for each classifier, with HPC features selected to achieve the highest possible F1 score with the fewest number of features. The HPC Features selected for training each classifier are shown in Figure 7.40. 120
7.42 Top 3 Cross Validation F1 scores for each classifier and F1 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features. 121
7.43 Comparison of HPC features with three highest F1 cross validation scores for 1st 4 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features using Bagged Tree Ensemble Classifier. 122

7.44 Comparison of HPC features with three highest F1 cross validation scores for 1st 4 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features using Subspace Discriminant Ensemble Classifier. . 122
7.45 Comparison of HPC features with three highest F1 cross validation scores for 1st 4 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features using Linear Support Vector Machine Classifier. . . 123
7.46 Comparison of HPC features with three highest F1 cross validation scores for 1st 4 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features using Fine K-Nearest Neighbor Classifier. 123
7.47 Comparison of HPC features with three highest F1 cross validation scores for 1st 4 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features using Narrow Neural Network Classifier. 124

8.1 Study 3 included all ransomware samples used in Studies 1 and 2, plus 4 new ransomware versions. 127
8.2 Data Collection Summary for Study 3. 128
8.3 Data Collection, System Restore, and Networked Report Transfer/Generation Procedure for 35-class data set (Study 3). 128
8.4 Summary of 20 Training Data Trials for each of 35 classes in Study 3 (HPC 48) `INST_RETIRED.ANY`. 129
8.5 Example Plot of HPC 48 (`INST_RETIRED.ANY`) Individual Sodinokibi Trials for Training Data Verification Trial 151 was identified as anomalous and repeated. 130
8.6 Start Time - Window Size - Stop Time Combinations for Study 3. Each column indicates a time interval for which HPC feature sets were generated (12 total). . . 131
8.7 Example: Confusion Matrix for Matthews' Correlation Coefficient (MCC) vs. F1 Score for imbalanced data. 133
8.8 Top 10 ranked HPC features for each classifier for 5 rounds of cross validation and training. 135
8.9 MCC scores averaged across each start/stop window for each classifier plotted against the number of optimized HPCs used to train the classifier. 136
8.10 Cross validation confusion matrices for Bagged Tree Ensemble Classifier trained with 6 optimized HPCs. Window size increases from left to right. 137
8.11 Cross validation confusion matrices for Subspace Discriminant Ensemble Classifier trained with 6 optimized HPCs. Window size increases from left to right. 137
8.12 Cross validation confusion matrices for Linear SVM Classifier trained with 6 optimized HPCs. Window size increases from left to right. 138
8.13 Cross validation confusion matrices for Fine KNN Classifier trained with 6 optimized HPCs. Window size increases from left to right. 138
8.14 Cross validation confusion matrices for Narrow Neural Network Classifier trained with 6 optimized HPCs. Window size increases from left to right. 138
8.15 Average MCC by window size for classifiers trained with 6 optimized HPCs. . . . 139
8.16 Percentage of Correct Predictions by Classifier and Start/Stop Window for all ransomware trials. 140
8.17 Percentage of Correct Predictions by Classifier and Start/Stop Window for all benign trials. 141

8.18 Classification Accuracy for classifiers evaluated with separate test data, collected in randomized order on two different systems with identical hardware. 144
8.19 Classification MCC for classifiers evaluated with separate test data, collected in randomized order on two different systems with identical hardware. 144
8.20 Average MCC by window size for classifiers tested with 6 optimized HPC features. 145
8.21 Confusion Matrix for Bagged Tree Ensemble classifier trained with 6 optimized HPC features for window sizes of 0.1s, 0.5s, 1s, and 2s at operation trigger time. . 145
8.22 Confusion Matrix for Subspace Discriminant Ensemble classifier trained with 6 optimized HPC features for window sizes of 0.1s, 0.5s, 1s, and 2s at operation trigger time. 146
8.23 Confusion Matrix for Linear SVM classifier trained with 6 optimized HPC features for window sizes of 0.1s, 0.5s, 1s, and 2s at operation trigger time. 146
8.24 Confusion Matrix for Fine KNN classifier trained with 6 optimized HPC features for window sizes of 0.1s, 0.5s, 1s, and 2s at operation trigger time. 147
8.25 Confusion Matrix for Narrow Neural Network classifier trained with 6 optimized HPC features for window sizes of 0.1s, 0.5s, 1s, and 2s at operation trigger time. . 147
8.26 Percentage of Correct Predictions for 6 HPC feature Classifiers for all (12) sets of Start/Stop Time Windows for all ransomware trials. 149
8.27 Percentage of Correct Predictions for 6 HPC feature Classifiers for all (12) sets of Start/Stop Time Windows for all benign trials. 149
8.28 Top 10 ranked HPC features for each classifier for 5 rounds of cross validation and training, with features generated for durations of 0.1, 0.5, 1, and 2s from visually identified operation start time. 151
8.29 Cross Validation Accuracy (top) and MCC (bottom) for classifiers trained with the top 1-9 HPC features by MCC rank generated for durations of 0.1, 0.5, 1, and 2s from visually identified operation start time. 152
8.30 Test Data Accuracy (top) and MCC (bottom) for classifiers trained with the top 1-9 HPC features by MCC rank generated for durations of 0.1, 0.5, 1, and 2s from visually identified operation start time. 152
8.31 Summary of Test Data Accuracy (top) and MCC (bottom) for classifiers trained with the top 1-9 HPC features by MCC rank generated for windows of 0.1, 0.5, 1, and 2s beginning at 15s, adapted from Section 8.3.1 for comparison purposes. . 153
8.32 % of Correct Predictions for benign operations, using 0.1, 0.5, 1, and 2s windows at actual operation start time (identified visually) and 9 HPC features. Operations for which 100% of trials were predicted correctly across all feature sets are in bold. Operation/classifier combinations which predicted less than 80% of trials correctly are in orange, while operation/classifier combinations that had fewer than 50% of trials predicted correctly are in red. 155
8.33 % of Correct Predictions for ransomware operations, using 0.1, 0.5, 1, and 2s windows at actual operation start time (identified visually) and 9 HPC features. Operations for which 100% of trials were predicted correctly across all feature sets are in bold. Operation/classifier combinations which predicted less than 80% of trials correctly are in orange, while operation/classifier combinations that had fewer than 50% of trials predicted correctly are in red. 155

8.34 Comparison of % of Correct Predictions for benign operations, with 15s Start Time Results (from Figure 8.27) on the left, and visually identified actual operation start time shown on the right. Just with just 6 HPC features were required for the results on the left, while 9 HPC features were required to obtain the results on the right. 156
8.35 Comparison of % of Correct Predictions for ransomware operations, with 15s Start Time Results (from Figure 8.27) on the left, and visually identified actual operation start time shown on the right. Just with just 6 HPC features were required for the results on the left, while 9 HPC features were required to obtain the results on the right. 157
8.36 Titles and descriptions of top 3 ranked performance counters for all combinations of classifiers at 15 seconds, 25 seconds, and actual operation start time. 160
8.37 Summary of all training trials and all 35-classes of benign and ransomware operations for performance counter 2: BR_INST_RETIRED.FAR_BRANCH_PS. The 20 benign operations are displayed in the top 5 rows, while the 15 ransomware operations are displayed in the bottom 4 rows. 161
8.38 Summary of all training trials and all 35-classes of benign and ransomware operations for performance counter 64: L2_RQSTS.CODE_RD_MISS. The 20 benign operations are displayed in the top 5 rows, while the 15 ransomware operations are displayed in the bottom 4 rows. 162
8.39 Summary of all training trials and all 35-classes of benign and ransomware operations for performance counter 67: L2_RQSTS.PF_HIT. The 20 benign operations are displayed in the top 5 rows, while the 15 ransomware operations are displayed in the bottom 4 rows. 163

9.1 Experimental setup for monitoring power supplied to the test SSD from the host computer. Current is measured with a current probe, amplified, and acquired and stored with the data recorder. 168
9.2 Seven SSDs were tested. Drives were selected to cover multiple form factors (2.5” SSD and M.2 module), interface types (SATA III and PCIe 3.0 x4), and technologies (3D NAND and 3DXP), as indicated above. 170
9.3 Data Collection Flow Chart 171
9.4 Representative Reads of 500 MB files for all Crucial, Optane, and Samsung SSDs tested. Fifteen trials were conducted at each of 5 file sizes. Power is provided through the 12V line to all PCIe drives (all Optane drives, the Crucial P5, and the Samsung 970 EVO Plus), and through the 5V line for the SATA III SSDs (Crucial MX500 and Samsung 850 EVO). Note: Y-axis for each drive is scaled to show detailed signature and should not be used for direct power comparison. 172
9.5 Representative Writes of 500 MB files for all Crucial, Optane, and Samsung SSDs tested. Fifteen trials were conducted at each of 5 file sizes. Power is provided through the 12V line to all PCIe drives (all Optane drives, the Crucial P5, and the Samsung 970 EVO Plus), and through the 5V line for the SATA III SSDs (Crucial MX500 and Samsung 850 EVO). Note: Y-axis for each drive is scaled to show detailed signature and should not be used for direct power comparison. 174

9.6 Longest pulse average power for (a) writes and (b) reads as a function of file size. Values plotted are averages across trials (n = 15). Error bars are +/- 2 standard errors. . . . 175
9.7 Data Flow for collection of read, write, and idle operations on four SSDs under test. 179
9.8 Representative time domain (left) and spectrogram (right) plots for 1 GB read operations for the Samsung, Western Digital, Crucial, and Optane drives, respectively (top to bottom). (Values smaller than -115 dB in the spectrogram plots are thresholded to -115 dB to facilitate visualization by improving image contrast.). . 180
9.9 Representative time domain (left) and spectrogram (right) plots for 1 GB write operations for the Samsung, Western Digital, Crucial, and Optane drives, respectively (top to bottom). (Values smaller than -115 dB in the spectrogram plots are thresholded to -115 dB to facilitate visualization by improving image contrast.). . 181
9.10 Representative time domain (left) and spectrogram (right) plots for the idle state for the Samsung, Western Digital, Crucial, and Optane drives, respectively (top to bottom). (Values smaller than -115 dB in the spectrogram plots are thresholded to -115 dB to facilitate visualization by improving image contrast.). . . . 182
9.11 Classification results: Drive/Operation/File size . . . 186
9.12 Full classification results for the individual test data sets: *2022 Drive 1* data and *2022 Drive 2* data. . . . 187
9.13 Classification by operation. . . . 188
9.14 Classification results by operation for the individual test data sets: *2022 Drive 1* data and *2022 Drive 2* data. . . . 189
9.15 Classification by drive. . . . 190
9.16 Classification results by by drive model for the individual test data sets: *2022 Drive 1* data and *2022 Drive 2* data. . . . 191
9.17 Classification results by drive model and operation for both *2022 Drive 1* data and *2022 Drive 2* data sets. . . . 192
9.18 Classification results by drive model and operation for the individual test data sets: *2022 Drive 1* data and *2022 Drive 2* data. . . . 193
9.19 Collection apparatus. . . . 194
9.20 Current Signatures of idle, 7zip encryption, and live ransomware samples (Darkside, Sodinokibi, WannaCry). Data is recorded for 150 seconds, with encryption and Ransomware triggered manually at the 30 second mark. . . . 194
9.21 1 second section of Current Signatures of idle, 7zip encryption, and live ransomware samples (Darkside, Sodinokibi, WannaCry). . . . 195
9.22 Frequency Spectrum of idle, 7zip encryption, and live ransomware (Darkside, Sodinokibi, WannaCry) at sampling rate of 200kHz with amplitude truncated after 1 mA. . . . 195
9.23 1kHz of Frequency Spectrum of idle, 7zip encryption, and live ransomware (Darkside, Sodinokibi, WannaCry) at sampling rate of 200kHz. . . . 196
9.24 Confusion matrix for 5-way classification of idle, 7zip encryption, and live ransomware (Darkside, Sodinokibi, WannaCry). . . . 197
9.25 Current for a representative trial for each operation type from trial initiation for 175 seconds. . . . 198

9.26 Current for a representative trial for each operation type from operation initiation at 60 second mark for 5 seconds. . . . 198
9.27 Data Collection Summary for simultaneous current and Vtune HPC collection for 36-classes, used to fine-tune the feature generation and select well-performing classifiers for further exploration. One ransomware class (WannaCry_1022) was eventually dropped from the final study because it did not perform encryption operations. . . . 199
9.28 Cross Validation Accuracy for non-exhaustive combinations of trail duration, Welch Window size, maximum feature frequency, and dB units, recorded from MATLAB Classification Learning Application. . . . 202
9.29 Top 20 Cross Validation Accuracy results (based on Coarse Tree validation ranking) for non-exhaustive combinations of trail duration, Welch Window size, maximum feature frequency, and dB units, recorded from MATLAB Classification Learning Application. . . . 203
9.30 Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for benign operations: OS only, 7zip, SPEC01 7zip, SPEC02 octave, SPEC03 Blender. . . . 205
9.31 Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for benign operations: SPEC04 CalculiX, SPEC05 Convolution, SPEC06 FFTW, SPEC07 fsi, SPEC08 handbrake. . . . 206
9.32 Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for benign operations: SPEC09 Kirchhoff, SPEC10 lammps, SPEC11 luxrender, SPEC12 namd, SPEC13 WPCcfd. . . . 207
9.33 Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for benign operations: SPEC14 poisson, SPEC15 python36, SPEC16 rodinaLifeSci, SPEC17 rodiniaCFD, SPEC18 srmp. . . . 208
9.34 Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for ransomware operations: Babuk 1022, DarkSide Spr21, DarkSide 0521, Gibberish 0422, HiddenTear 0422. . . . 209
9.35 Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for ransomware operations: Phobos 0122, Phobos 0522, Phobos 0922, Snatch 0422, Snatch 0521. . . . 210
9.36 Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for ransomware operations: REvilSodinokibi Spr21, Sodinokibi 0722, Sodinokibi 0222, WannaCry Spr22, WannaCry 0622. . . . 211
9.37 Cross Validation Accuracy for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features. . . . 213
9.38 Cross Validation Accuracy for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features in dB. . . . 214
9.39 Cross Validation MCC for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features. . . . 215
9.40 Cross Validation MCC for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features in dB. . . . 216

9.41 Percent of Total Correct Predictions for all combinations of Welch Window and Cutoff Frequency for 1-3s, 5s, 7s, 10s, and 15s for benign operations, with PSD conversion in dB. 217
9.42 Test Data Classification Accuracy for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features. 219
9.43 Test Data Classification Accuracy for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features in dB. . . . 220
9.44 Test Data Classification MCC for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features. 221
9.45 Test Data Classification MCC for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features in dB. 222
9.46 Top 3 results for each trial segment length for eleven different durations (ranging from 1-15 seconds) tested. Results are ranked from low-to-high segment length and then by high-to-low MCC. 224

List of Abbreviations

3DXP	3D Cross Point
AES	Advanced Encryption Standard
BIOS	Basic Input/Output System
BPI	Blind Power Identification
CER	Cross Entropy Ratio
CLI	Command Line Interface
CPU	Central Processing Unit
CSV	Cache Side-channel Vulnerability
DSS	Digital Signature Standard
DVS	Dynamic Voltage & Frequency Scaling
EA	Electronic Attack
EDR	Endpoint Detection & Response
EM	Electromagnetic
eMMC	embedded Multi-Media Card
EMS	Electromagnetic Spectrum
EMSC	Electromagnetic Side-Channel
EP	Electronic Protection
ES	Electronic Warfare Support
EW	Electronic Warfare
FFT	Fast Fourier Transform
FLIR	Forward Looking Infrared
FPGA	Field Programmable Gate Array
GE	Guessing Entropy
GPU	Graphics Processing Unit
HAC	Holistic Assessment Criterion
HMD	Hardware-level Malware Detection
HPC	Hardware Performance Counter

IC	Integrated Circuit
IDQ	Instruction Decode Queue
IFL	Image for Linux
ILR	Information Leakage Rate
IO	Input Output
IP	Intellectual Property
IR	Infrared
KNN	K-Nearest Neighbor
LED	Light-Emitting Diode
LPF	Low-Pass Filter
MASC	Microarchitectural Side-Channel
MCC	Matthews' Correlation Coefficient
MI	Mutual Information
ML	Machine Learning
MOE	Measure of Effectiveness
NETD	Noise Equivalent Temperature Difference
NVMe	Non-Volatile Memory express
OHM	Open Hardware Monitor
OS	Operating System
PCA	Principal Component Analysis
PCIe	Peripherial Component Interconnect express
PCTA	Progressive Correlation Thermal Analysis
PI	Perceived Information
PSC	Power Side-Channel
PSCA	Power Side-Channel Attack
QLC	Quad Level Cell
RaaS	Ransomware-as-a-Service
RAT	Resource Allocation Table
RMS	Root Mean Square
RO	Ring Oscillator
RSA	Rivest Shamir Adleman
SATA	Serial AT (or Advanced Technology) Attachment
SAVAT	Signal AVailable to the ATtacker
SBC	Single-Board Computer

SCA	Side-Channel Attack
SCP	Secure Copy SCAN
Side-Channel ANalysis	
SCD	Side-Channel Defense
SQ	Super Queue
SR	Success Rate
SSD	Solid State Drive
SSH	Secure SHell
STSF	Spatial Thermal Side-channel Factor
SVF	Side-channel Vulnerability Factor
SVM	Support Vector Machine
TSC	Temperature Side-Channel
TSCA	Temperature Side-Channel Attack
TSMP	Thermal-Security-in-Multi-Processors
TVLA	Test Vector Leakage Assessment
UOPS	Micro-operations
USB	Universal Serial Bus

Chapter 1: Introduction

Computers exude information, by design. They have become fully integrated into our lives - from word processing to gaming, streaming music and movies, big data crunching, and accessing unlimited information through the internet with the click of a button, often in the palm of your hand - because we have come to depend on the myriad services computer systems provide. Much of the information that is obtained from computers is intentional - documents, movies, and games are displayed to monitors, music streams through headphones and speakers, and network packets are transmitted wirelessly to and from routers, all originating from the device and sent through the proper channel to an intended receiver.

The physical implementation of computer hardware that provides these services leads necessarily to physical behavior on the part of an operating computer. This physical behavior has physical characteristics, many of which become channels of information leakage that can be observed by an *unintended* receiver. This can pose a serious threat to computer security.

1.1 A Brief History of Side-channels

These "side-channels" of computer operations, such as current usage and power consumption, generation of heat and electromagnetic radiation, and events at the micro-architectural level, can be exploited to compromise the confidentiality of a system. The identification of side-channels

as avenues to gather and exploit valuable information is relatively new - it was 1996 when Paul Kocher used differences in the timing of computer performance optimizations (a function of the chosen micro-architectural implementation) to find the entire secret key of asymmetric encryption algorithms such as Diffie-Hellman, Rivest Shamir Adleman (RSA), and Digitial Signature Standard (DSS) [5], followed closely by analyzing the power consumed during cryptographic operations to extract keys from dozens of products [6]. In the 2000s the National Security Agency declassified the TEMPEST program [7], and electromagnetic signal leakage was identified as another viable side-channel [8].

In the mere 20 years that followed, a number of additional side-channels have been commonly accepted: Acoustic, optical, temperature, memory/cache, and micro-architectural; and advantages, disadvantages, and methods of exploiting and securing each are active research areas. The related nomenclature, however, is highly inconsistent. Side-channel attacks are classified a number of ways : Active vs. Passive, Invasive vs. Non-invasive [9], Simple vs. Differential [10], Profiled vs. Non-Profiled [11], but the terms are often used in overlapping contexts. Some works refer to Side-channel Analysis and Constructive Side-channel Analysis while others refer to the same techniques as passive side-channel attacks and side-channel defense, respectively. Meanwhile, what actually constitutes a side-channel is also ambiguous - with multiple names used interchangeably for the same leakage vector, sometimes compounded by conflating the name of the side-channel with an attack method. "Mutual information," generally considered statistical measure of the amount of information shared between two random variables, is at times considered a metric [12, 13] , a side-channel [14], an analysis method [15, 16], and an attack method [17]. There are as many as six different terms used for the similar side-channel methods related to micro-architecture/memory/cache/access/timing/transient execution, with no

clear guideline or definition to distinguish between them. The highly publicized Spectre [18] and Meltdown [19] attacks, released in 2018, exemplify this issue. In a fairly new, rapidly evolving field, every novel attack that is identified sends researchers scrambling for a solution. It's not at all surprising that the field is disorganized and terminology is inconsistent; the focus is always reacting to the next threat. For the purposes of this work, side-channels will organized and discussed in the framework of Electronic Warfare, as discussed in Section 1.1.1, below.

1.1.1 Considering Side-channels in the framework of Electronic Warfare

Electronic Warfare (EW) is the term given to military operations to control the electromagnetic spectrum (EMS), or the range of frequencies of electromagnetic radiation. Electromagnetic waves consists of perpendicular electric and magnetic fields, as shown in Figure, and the EMS is organized by frequency (and corresponding wavelength), seen in Figure 1.2.

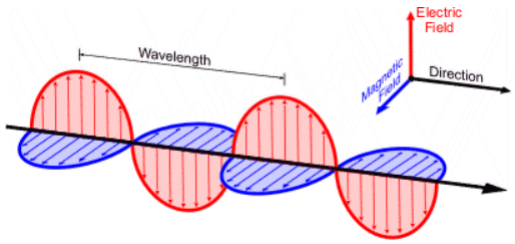


Figure 1.1: Electromagnetic wave, reprinted from [2].

EW focuses on maintaining friendly control of the EMS while denying its use to adversaries, and is further divided into 3 categories: Electronic Attack (EA), Electronic Warfare Support (ES), and Electronic Protection (EP) [20]. EW has been used in a military capacity for over a hundred years. The earliest documented attempt to use EW in a military capacity was during the Russo-Japanese War in 1905, when a Russian Captain requested (but was denied) permission to transmit a signal to interfere with a Japanese plane that had spotted the Russian Fleet in the

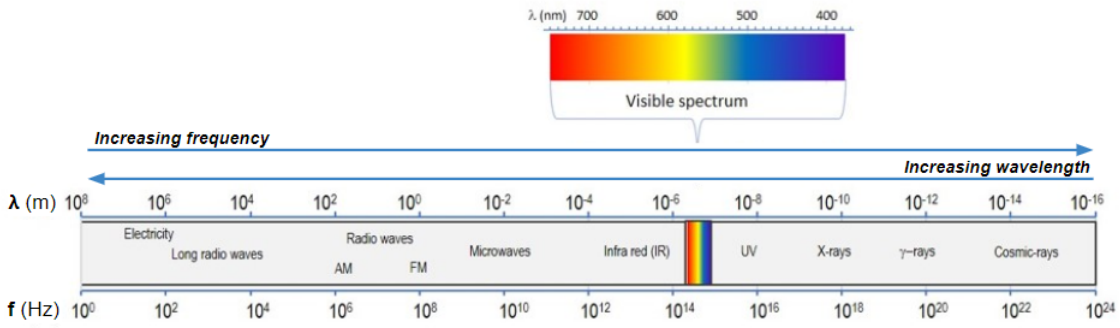


Figure 1.2: Electromagnetic Spectrum, adapted from [3].

Tsushima Straight, which ended in a crushing Russian defeat that ultimately ended the war in Japan's favor [21]. It has been widely used to gain an operational advantage since World War II, when both the Allies and Axis powers employed EW against the navigational systems of bomber aircraft, among other uses [22].

In the century since EW was first introduced, a time-tested framework has been developed to define and organize the EW field. EW focuses on maintaining friendly control of the EMS while denying its use to adversaries, and is further divided into 3 categories: Electronic Attack, Electronic Support, and Electronic Protection [20].

- **Electronic Attack (EA)** - EA is the use of the EMS or directed energy to attack an adversary, either by preventing their access to the EMS or by preventing the adversary from denying friendly access to the spectrum [23].

- **Electronic Warfare Support (ES)** - ES is the use of the EMS for information gathering, specifically for the purpose of threat recognition, planning, or supporting future operations [23].

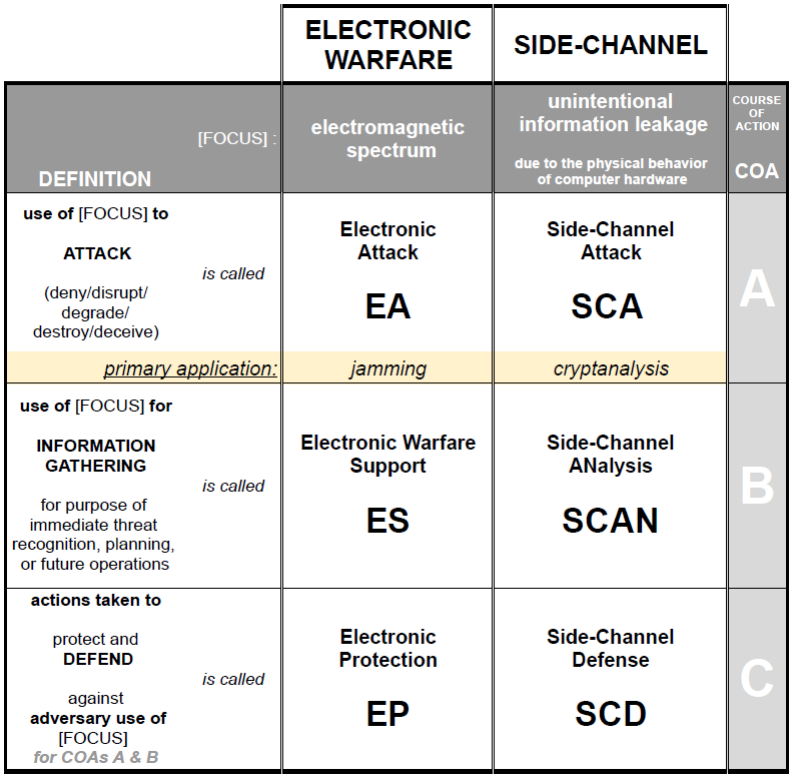
- **Electronic Protection (EP)** - EP refers to actions taken to protect personnel, facilities, and equipment from any effects the use of the EMS for EA or ES [23].

Although side-channels are quite recent in comparison, similarities exist. EW is focused on actions involving the range of electromagnetic radiation in the EMS with EA indicating "jamming." Side-channels, in comparison, center on information leakage from the physical behavior of computer hardware, where attacks primarily constitute obtaining secret encryption keys or privileged system access. This work will consider side-channels in the framework of EW, where the distinct characteristics of the power, acoustic, electromagnetic, temperature, optical, and micro-architectural side-channels are analogous to the divergent properties and propagation methods exhibited by the different frequency bands in the EMS. To organize inconsistent terminology for side-channel methods, actions to (actively) attack side-channels with the goal of cryptanalysis or privileged system access will still be called Side-Channel *Attacks* (SCA). Use of side-channels for the purposes of information gathering and threat recognition will be termed Side-Channel *ANalysis* (SCAN), and actions taken to defend side-channels - either through minimizing side-channel leakage or by reducing the relationship between the leakage and sensitive information - will be considered categorized as Side-Channel *Defense* (SCD). For reference purposes, relationships between EW and SC methods are shown in Figure 1.3.

1.2 Side-channel Analysis to detect Ransomware

Side-channel *analysis* is valuable in determining what kind of unintentional information is leaked from a system, and can be used to detect modifications to a system or validate the expected operation [24–26]. In order to provide scope and a pertinent, real-world appliction for which to evaluate the effectiveness of disparate side-channel methods, this research focuses on side-channel *analysis* of selected side-channels (power, temperature, and micro-architectural),

Figure 1.3: Relationship between Electronic Warfare terminology and Side-Channel terminology.



with specific applications to the detection of state-of-the-art, real ransomware running on a live, non-virtualized system.

The impact of even the perceived threat of ransomware attacks was brought to the front of national consciousness in May 2021, when the largest US refined fuel pipeline operator, Colonial Pipeline, shut down its fuel distribution system for 5 days in response to a ransomware attack on its administrative system by the DarkSide ransomware group, causing panic buying, price surges, and fuel outages [27]. Myriad reports of attacks on critical infrastructure, school systems, hospitals, government systems, and private companies occur almost weekly; the first human death was attributed to a ransomware attack when an German woman's ambulance was re-routed from a nearby Dusseldorf hospital after it was attacked by WannaCry ransomware in September 2020 [28]. In November 2022, the U.S. Treasury released that the total cost of ransomware attacks

on U.S. financial institutions alone in 2021 increased 200% over the prior year to $1.2 *billion* dollars [29], and these numbers are estimated to be well below the true figure as it includes only data that was required to be reported by U.S. banks [30]. The global impact of ransomware is so significant that the White House has hosted two International Counter Ransomware Initiative Summits in the past two years [31].

Typical ransomware detection methods depend on up-to-date anti-virus software signatures created from previously discovered versions of ransomware, and are unlikely to prevent brand new "zero-day" ransomware attacks. This work considers side-channel analysis techniques for the temperature, power, and micro-architectural side-channels for the purpose of classifying state-of-the-art ransomware on real world, non-virtualized Windows systems. Over three thousand ransomware and benign trials were collected to generate training and testing data sets, which required development of a process to synchronize collection of on-system (e.g. performance counters) and off-system (e.g. power) measurements, safely transfer trial data from the encrypted system, and restore the system to a "clean" state without the use of virtualization techniques, which negatively impact the validity of side-channel measurements. Side-channels were evaluated on their effectiveness in accurately differentiating between ransomware and benign operations such as background operating system activity, 7zip encryption, and SPEC benchmarks, in a given time duration, with Matthews' Correlation Coefficient (MCC) used to measure overall classifier performance of five machine learning classification algorithms. With the financial impact of ransomware estimated to cost more than $30 billion globally this year [32], the usefulness of side-channel analysis to detect ransomware in a non-virtualized computer system has significant real-world implications.

1.3 Contributions

This work is the first hardware-based ransomware classification and detection exploration to leverage side-channel analysis of multiple (power, temperature, micro-architectural) side-channels on hardware without use of virtualization techniques. Each side-channel is evaluated for its usefulness in classifying ransomware based on detection time required to achieve at least 90% classification accuracy. Specifically, the following contributions are insights that came from this work:

1. This work developed a process to collect system-wide data from the power and micro-architectural side-channels simultaneously during ransomware execution without the use of virtualization.

2. The micro-architectural side-channel, accessed by collecting hardware performance counters through Intel's VTune Profiler, produced test accuracy of 99% in $\leq$ 1 second. Over 200 hardware events were collected and systematically tested for their ability to differentiate ransomware from benign operations. The top 10 events were ranked by classifier, and best events were found to change based on classifier and time window size. Accuracy results for a 0.1s window were within 10% of best case results for each classifier.

3. The power side-channel, accessed by collecting current supplied to a solid state drive, produced test accuracy of 96% in 15 seconds, and required at least 4 seconds to pass 90%.

4. Temperature side-channel-related research was surveyed and organized, but proof-of-concept efforts to utilize the Temperature Side-channel for ransomware detection indicated it is not

a viable path for time sensitive applications without the use of pre-processing, which is outside the scope of this work.

5. Since the terminology used to refer to side-channel topics is inconsistent at best, a preliminary attempt to organize side-channel concepts in the framework of Electronic Warfare is described and utilized in this work.

1.4 Organization

The remainder of this work is organized as follows: Chapter 2 provides a literature review and background on side-channel types, methods, and metrics. Chapter 3 considers the less-utilized temperature side-channel in depth, and Chapter 4 provides additional background on the micro-architectural side-channel and use Hardware Performance Counters (HPCs) to obtain micro-architectural side-channel information. Chapter 5 includes background and related literature on side-channel analysis-based methods to detect malware and ransomware, along with additional analysis of all previous works that have investigated the use of HPCs for ransomware detection. Experiments accessing the temperature side-channel through thermal imaging are included in Chapter 6. Chapters 7 and 8 describe the experiments and results leveraging micro-architectural side-channel hardware performance counter event data to classify ransomware operations. Evaluation of power side-channel-based current-draw analysis and its feasibility for ransomware classification and detection are provided in Chapter 9. This work concludes with Chapter 10, which summarizes results and discusses future research directions to employ side-channel analysis techniques for detection of ransomware on a non-virtualized system.

Chapter 2: Side-channels

2.1 Side-channels

The physical implementation of computer hardware leads necessarily to physical behavior on the part of an operating computer. This physical behavior has physical characteristics, many of which become channels of information leakage that can be observed by an unintended receiver.

Digital computations result in physical effects such as current usage and power consumption, generation of heat and electromagnetic radiation, and events at the micro-architectural level. Although these effects are not intended to convey information about the operations being performed, they create the opportunity to do so through side-channel analysis. A side-channel is a source of observable information leakage through methods other than the intended communications channel. Whereas the communications channel itself is the medium through which information is passed between a transmitter and receiver, a side-channel is a medium through which information may be leaked or observed due to the physical effects caused by the operation or implementation of a digital process, rather than through direct access to the device hardware or software itself. Typical side-channels include power consumption, electromagnetic emissions, acoustic emissions, optical emissions, heat generated, and micro-architectural effects: execution time required and memory resources consumed. These commonly used side-channels are considered below.

2.1.1 Power Side-channel

The power side-channel is based on the current draw of CMOS devices in a transitory state. As data is processed, the millions of these transitions taking place have a direct impact on a device's power consumption. Power side-channel analysis infers activity or operations based on measuring the power consumed by a device. The power side-channel has been widely studied for over 20 years and recently summarized in comprehensive survey papers [33, 34]. It allows for multiple types of power and current collection methods to determine the ground truth of the amount of power consumed by a device with high fidelity; however, it is typically invasive to measure and limited to a single device or chip.

2.1.2 Electromagnetic Side-channel

The electromagnetic (EM) side-channel is a result of the electromagnetic field generated by the flow of current. It has also been widely studied and recently surveyed [35, 36], is considered the most useful side-channel when power measurements are unavailable, and is particularly helpful when an implementation is resistant to side-channel power analysis [8]. The EM side-channel does not require direct access to a device, and is usually measured with a near-field probe placed in close proximity to the device under test. A newer, related application of the EM side-channel is called "Backscattering," which is a result of EM signals being reflected and simultaneously combined with effects of the circuit activity at that moment [37, 38].

2.1.3 Acoustic Side-channel

A classic form of eavesdropping, the acoustic side-channel is a result of the sound that is created in multiple computation scenarios [39], including determining keystrokes on a keyboard [40–42], finger taps on the touch-screen of a smart phone [43], inferring what is printed on a traditional [44] or 3D [45, 46] printer, and even cryptanalysis resulting in the extraction of encryption keys [47–49]. The most recent application of the acoustic side-channel is the first active acoustic SCA by Cheng et. al. [50], which used inaudible audio signals to enable a smartphone to use SONAR to track human movements and disclose unlock patterns.

2.1.4 Optical Side-channel

Most examples of the optical side-channel consider the analysis of the photons emitted when a transistor changes state, which has been applied to AES [51–53] and RSA [54] encryption. The optical side-channel has also been applied to supplement power analysis when using a charge-coupled device light-sensing camera [55], and by using the status of router LEDs to covertly communicate and exfiltrate data [56].

2.1.5 Temperature Side-channel

The speed at which a microprocessor operates, combined with the movement of charge required to change the state of transistors gives rise to heat, or a temperature-based side-channel. An in-depth discussion of the temperature side-channel is provided in Chapter 3.

2.1.6 Micro-architectural Side-channel

Microarchitecture is the specific design of a microprocessor, and consists of all the digital logic, arithmetic, and data path and control circuits required to implement an instruction set in a given processor. The micro-architectural side-channel leverages the distinct hardware features inherent in the particular microprocessor design, particularly with regards to the time it takes to execute instructions, contention inherent is the sharing of hardware resources, and the functionality of the memory subsystem. In contrast to previous side-channels, this side-channel is software-based and does not necessarily require physical proximity to the target device. The Micro-architectural side-channel consists of both Timing and Memory Access related side-channel information and is considered in detail in Chapter 4.

2.2 Side-channel Attack, Analysis, and Defense

Side-channel attacks take advantage of the physically observable characteristics of computer tasks on specific hardware implementations, and were originally considered to be sortable into two orthogonal classifications according to [9]:

1. **Active.** *vs.* **Passive.** Active attacks deliberately tamper with the proper functioning of the device in question. Passive attacks observe the behavior of the device without causing any disruption to the device itself.

2. **Invasive.** *vs.* **Non-Invasive.** Invasive attacks require gaining access to the inside components of a chip, usually by depackaging. Non-invasive attacks exploit information which is externally observable.

More recent classification methods consider additional orthogonal axes:

1. **Active.** *vs.* **Passive.**

2. **Invasive.** *vs.* **Semi-Invasive.** *vs.* **Non-Invasive.**

3. **Simple.** *vs.* **Differential.** This classification is based on the method used to analyze the side-channel data. A simple analysis generally utilizes a single side-channel trace, where information can be extracted directly from the side-channel observations. A differential analysis, on the other hand, uses many side-channel traces and statistical (or ML) approaches to find the correlation between the side-channel information and the secret data. [10]

4. **Non-Profiled.** *vs.* **Profiled.** A non-profiled attack uses traces measured directly from the target device (e.g. Simple or Differential Power Analysis), while a profiled attack depends on the use of a separate device to construct a profile of the target device, which is then used to attack the target. Template attacks and Deep learning-based attacks are powerful examples of profiled side-channel attacks [11].

Most frequently, active side-channel attacks are used to ascertain the secret key of an encryption algorithm. This is so frequent that the term "side-channel attack" has become synonymous with cryptanalysis. The passive, non-invasive attack described above essentially constitutes information gathering, and is better described as side-channel *analysis*, allowing for timely insights into the effectiveness of offensive or defensive measures. Side-channel defense is the use of side-channel insights to to defend against side-channel attacks. Defensive measures either aim to eliminate side-channel leakage itself or to eliminate the relationship between the side-channel leakage and sensitive information.

2.3 Side-channel Metrics

Side-channel attacks pose a considerable threat to the security of a system, so the ability to understand, quantify, and compare the threat vectors posed by side-channel leakage is crucial to understanding the a system's vulnerabilities.

A thorough survey of over 80 technical privacy metrics was released in 2018 [57], which considered all measures that described any level of privacy across six privacy domains: Communications systems, databases, location-based services, smart metering, social networks, and genome privacy, and grouped metrics by the output measures of uncertainty, information gain/loss, data similarity, indistinguishibility, error, time, accuracy/ precision, and the adversary's probability of success. This survey captured many of the metrics that have been used to quantify privacy, although many did not specifically focus on side-channel leakage. The pertinent metrics from [57], along with a wide variety of other statistical and novel side-channel leakage-based metrics are summarized in this section.

2.3.1 Statistical Tests

The earliest work on side-channel metrics looked at the ability of statistics to quantify the immunity of a device to leakage, as minimal leakage provides limited opportunity for the creation of side-channels. Leakage tests were designed to detect general vulnerability to timing and power consumption attacks, as well as correlation tests (Hamming Weight, external parameters). These tests were based on existing statistical tests for randomness (F-test, R-test) and significance (distance of means, goodness of fit, sum of ranks). Failure of any of the designed leakage test indicated probable emission of secret information; however, passing any of these tests does NOT

imply a device does not leak information [58].

2.3.2 Entropy, Conditional Entropy, & Guessing Entropy

Entropy is the average amount of information provided by the outcome of a random event, or the measure of uncertainty of a random variable [59]. A decrease in uncertainty corresponds to an increase in information. Entropy as a side-channel metric originated from Shannon's description of a communications channel and was explored as a desirable measure of predictability in the context of atmospheric science and weather forecasts by [60] and for early side-channel attack modeling by [61]. It is used as the basis of many metrics, including Gu's Spatial Thermal Side-channel Factor (STSF) in section 2.3.7 [62]. [57] summarized arguments against its use as a privacy metric due to the heavy influence of outlier values, which can make it misleading and difficult to use in comparison, as it only gives a general indication of uncertainty with no insight into how accurate the estimate is.

2.3.2.1 Conditional Entropy

Conditional Entropy is a measure of how much information is needed to describe the outcome of an event X given the known outcome of another event Y. In the side-channel context, Y can be considered as the attacker's observations of the given side-channel of interest. Conditional Entropy directly yields Mutual Information (Section 2.3.3) [12, 57, 61].

2.3.2.2 Guessing Entropy

Guessing entropy is a security metric that gives the average number of questions that must be asked to correctly guess a value. In the context of side-channel attacks, it estimates the average number of key candidates that will need to be tested after the attack is complete, thus quantifying the effectiveness of the attack [12, 61, 63].

2.3.3 Mutual Information

Mutual Information is the measure of the amount of information shared between two random variables. In the side-channel context, the variables are usually the true distribution of information and the attacker's observations of that information, hence measuring the amount of information leakage [12, 57]. [14–17, 64–66] all explore Mutual Information Analysis for side-channel analysis in varying contexts. Mutual Information is also used as the basis of the Side-channel Leakage Evaluator and Analysis Kit in [67] and is related to Success Rate (Section 2.3.4) [13].

2.3.4 Success Rate

Generally speaking, Success Rate measures the probability that the adversary is successful by determining the percentage of successes over a large number of attempts [57]. In the side-channel application, Success Rate indicates the efficiency with the adversary can recover the secret key [12, 13]. [68] explores the application of this metric to deep-learning-based side-channel analysis of imbalanced data and finds that it is difficult to embed in DL algorithms, but closely related to Cross Entropy Ratio (Section 2.3.16.2).

2.3.5 Welch's T-Test

Welch's T-test uses hypothesis testing to determine if two separate distributions with unequal variances have equal means. It is the underlying metric used by the Test Vector Leakage Assessment (TVLA) testing methodology initially developed by [69], which focuses on determining the resistance of a cryptographic module to leakage of information through the power side-channel. [70] thoroughly examines and extends the work of [69], with detailed applications of the T-test in higher-order settings. It is identified by [57] as a privacy metric to measure data similarity, and by [71] with gate-level power measurements in conjunction with electronic design automation tools. [72], however, cautions against the use of Welch's T-Test as a standalone pass/fail metric in TVLA to assess whether a cryptographic implementation is safe.

2.3.6 Side-channel Vulnerability Factor (SVF)

SVF is the novel metric developed in [73–75] that measures the information leakage through a side-channel by determining the correlation between the actual cache activity trace ("oracle") and the side-channel observations for memory side-channels.

2.3.7 Spatial Thermal Side-channel Factor (STSF)

The Entropy-based (Section 2.3.2) STSF was developed in [62] as a two-dimensional metric to complement SVF (Section 2.3.6). STSF accounts for the temperature of function blocks as they correlate to secret information.

2.3.8 Cache Side-channel Vulnerability (CSV)

CSV constrains SVF (Section 2.3.6) by limiting its application to caches only and assumes the strongest possible attacker in order to remove the ambiguity of differences between system vulnerabilities and attacker capabilities [76].

2.3.9 Signal Available to the Attacker (SAVAT)

SAVAT is an instruction-level metric that measures the signal made available through the side-channel as a result of a single instruction variation, and is determined by directly analyzing the variation between individual processor instructions [77].

2.3.10 Thermal-Security-in-Multi-Processors (TSMP)

TSMP was introduced by [78] as a metric to quantify the security of multiprocessors against a temperature side-channel attack. TSMP ranges from 0 (not secure) to 1 (more secure).

2.3.11 Maximal Leakage

Maximal Leakage is defined in [79] as the multiplicative increase in the likelihood of correctly guessing a randomized function of X after observing Y, maximized over all such functions. It quantifies the leakage of information from X to Y, by measuring the difference between an informed guess (after observing Y) and a blind guess. Applications of maximal leakage are further explored in [80–82], and as a way to measure information gained from databases or social networks in [57]. [83] argues for the use of maximal leakage over mutual information or channel capacity in evaluating (timing) side-channels.

2.3.11.1 Maximal α-Leakage

When refining guesses, maximal α-leakage gives an adversary the ability to fine-tune the level of confidence of additional guesses, allowing for continuous adjustment between mutual information ($\alpha = 1$, Section 2.3.3) and maximal leakage ($\alpha = \infty$, Section 2.3.11) [84].

2.3.12 Information Leakage Rate

Information leakage rate is a novel metric introduced in [85] to evaluate the amount of information leaked through the electromagnetic side-channel and compare the quality of EM side-channel measurement systems.

2.3.13 Local Differential Privacy

The local differential privacy metric measures how indistinguishable two items of interest are from each other [57], and is considered a useful leakage metric when a system designer is extremely risk averse [81, 82].

2.3.14 Trust Coverage

Trust coverage is a framework that calls for the application of a variable weighted sum of three different coverage metrics (Functional Coverage, Structrual Coverage, Asset Coverage) to quantify the trustworthiness of hardware at the gate-level, as a response to the popularity of hardware trojans [86].

2.3.15 Holistic Assessment Criterion

The Holistic Assessment Criterion focuses on assessing the leakage of power side-channels, and is intended to improve on the TVLA's T-test (Section 2.3.5) by focusing on a null hypothesis built on a well-founded definition of exploitable leakage [87].

2.3.16 Machine Learning Metrics

Recently, several metrics specific to use of machine learning in side channel analysis have been introduced due to discrepancies between the traditional machine learning (ML) metrics of accuracy, precision, and recall and side-channel metrics, with an eye towards designing a metric that better suits ML rather than increasing complexity by attempting to include side-channel metrics in ML algorithms [88].

2.3.16.1 Perceived Information

[89] found that, for balanced data, using the Negative Log Likelihood (aka Cross Entropy) loss function during the training of deep neural nets is the equivalent of maximizing perceived information, which is the lower bound of mutual information (Section 2.3.3) between the observation and the leakage.

2.3.16.2 Cross Entropy Ratio

Cross entropy ratio, which is closely related to both Guessing Entropy (Section 2.3.2.2) and Success Rate (Section 2.3.4), is a useful metric to evaluate the performance of deep learning models for side-channel analysis [68].

Chapter 3: The Temperature Side-channel

3.1 Overview

There exists an array of literature related to the the use of both offensive and defensive temperature methods to address the security of computer systems via the temperature side-channel.

3.2 Characteristics of the Temperature Side-channel

The TSC leakage has a linear relationship to the leakage of the PSC, but with limitations due to the thermal properties of materials. Modeling dynamic temperature changes is accomplished with an RC-equivalent circuit with large (thermal) capacitance [90], and therefore behaves like a low-pass filter (LPF) with a cutoff frequency ($f_c = \frac{1}{2\pi RC}$) in the low kHz range. Since modern processors operate in the GHz range, this low-pass filtering effect attenuates higher-frequency components of system activity, resulting with a delay in temperature compared to power (which changes virtually instantaneously). Temperature changes are slower both to increase and to decrease, which results in previous and current operations being superimposed at the sensor, so leakage has an integrative effect. Advantages and disadvantages of the TSC are listed below [91–94]. Advantages of the TSC include:

1. Linear relationship to power.

2. Ease of access through the use of on-chip sensors, with some applications for infrared imaging.

3. Leakage can be measured internally or externally through temperature variations.

The disadvantages of the TSC are not insignificant:

1. Low bandwidth due to LPF behavior, which attenuates the leakage of high frequency computations, resulting in slow temperature changes.

2. Noisy - since heat generated is superimposed spatially and temporally.

3. Data collection is limited by the response time and resolution of the thermal sensor.

4. Any temperature offset varies over time, as no mechanism exists to control it directly. Conversely, Dynamic Voltage and Frequency Scaling regulates power.

3.3 Sensing the Thermal Channel

Temperature side-channel readings can be obtained through both internal and external means. This section provides an overview of commonly used internal and external sensing options, as well as the context in which these have been used in the literature. The majority of works have used internal on-chip sensors to monitor the temperature side channel, with the Ring Oscillator being a popular internal implementation on FPGAs. External sensors were much less common, with only a small number of works using an external temperature sensor or infrared (IR) camera to capture temperature traces. Across these categories, limited success was found with direct IR image captures, with pre-processing required to adapt temperature readings for analysis through traditional power methods.

3.3.1 Internal Temperature Sensing Methods

Internal temperature readings are either acquired through the internal temperature sensor of each processor core or, when using re-programmable devices, through the use of a specially designed circuit (ring oscillator).

3.3.1.1 Internal Core Sensor

Since maintaining the proper operating temperature range is vital to the performance and lifetime of a chip, cores contain software-readable temperature sensors to monitor temperature and adjust the speed of operations if necessary. These sensors provide an easy source of temperature information *if* one has access to the sensors, either through physical access to the device or the use of a malicious program to provide sensor information. Resolution of this side-channel source is limited by the number of sensors on the device and their placement, as well as by the frequency of temperature readings [95].

The vast majority of existing literature utilizes on-chip sensors, many of which leverage "HotSpot" modeling software [96] to simulate sensor temperatures for a variety of applications. These applications include covert communications [97, 98], physical temperature attacks [99], thermal modeling [62, 93, 100, 101], defense against TSCAs [102] and detection of hardware trojans [103, 104]. Experimental results using on-chip sensors are limited to applications in covert communications [105, 106], and TSCA defense [78, 107, 108].

3.3.1.2 Ring Oscillator

A ring oscillator is a circuit that consists of and odd number of inverters in a feedback loop. When the device is enabled, the signal through the inverters oscillates and generates heat. Each inverter contributes to the delay of the signal, which decreases the frequency of the oscillating signal. Ring oscillators can also be used to detect temperature changes by correlating the output frequency drift to degrees Celsius or Fahrenheit. A number of works have explored the ring oscillator as a temperature sensor on FPGAs [109–117], and are useful in that context due to their ease of implementation, their dynamic nature (they can be added, moved, and removed as desired), their ability to measure junction temperature (in lieu of package temperature, like other on-chip sensors), and that they can be placed as desired (e.g. in an array in order to obtain a thermal map of the die). In the context of temperature side-channels they are used almost exclusively for covert communications applications [112, 118–120].

3.3.2 External Temperature Sensing Methods

External methods to sense temperature include infrared imaging, temperature sensors, and occasionally, fan speeds. Each method is described below.

3.3.2.1 Infrared Camera

Hot objects radiate electromagnetic infrared (IR) frequencies and the intensity of the radiation depends on the object's temperature. A thermal camera uses an array of infrared sensors to measure this electromagnetic radiation and then converts the measurement to an image showing the temperature of different objects as colors. There are several factors that determine the quality

of a thermal image, but the most pertinent to this work are resolution, temperature range, and thermal sensitivity.

Resolution. The resolution of the sensor is the number of sensitive elements ("pixels") that make up the sensor. Since infrared radiation has longer wavelengths than visible light, sensitive elements of infrared cameras are larger than those of traditional cameras, so thermal cameras have fewer pixels and lower resolution overall. Thermal cameras with more pixels produce higher-quality images.

Temperature Range. The temperature range of a given camera is the range of temperatures from the lowest to highest that the camera's thermal sensor (microbolometer) is capable of measuring. Some cameras have multiple temperature range options, which need to be chosen based on the temperature of the object being measured.

Thermal Sensitivity. Thermal sensitivity, also called Noise Equivalent Temperature Difference (NETD), is the smallest temperature difference a microbolometer can detect in the presence of electronic circuit noise. Lower numbers indicate the ease of distinguishing subtle temperature variations.

Several works propose that this would be an "ideal" way to sense temperature because it creates an air-gap between the monitoring equipment and the system being monitored (thus more secure), and does not require access to the on-chip sensors. Additionally, IR imaging has the potential to provide better resolution than a single individual on-chip sensor, as well as two-dimensional thermal/spatial correlation information. This method also eliminates monitoring overhead on the system, which is highly advantageous in resource and power constrained devices [101, 104, 107].

To date, there has only been limited success in applying infrared imaging to the temperature

side-channel, and primarily in processing data for analysis using power methods. Cochran et. al. presented a "thermal-to-power-inversion" method to estimate spatial power using a thermal camera collecting emissions from the back of a silicon die and compensated for challenges associated with the spatial LPF effect of heat diffusion [91]. Meanwhile, Reda et. al. solved the "Blind Power Identification" (BPI) problem, which uses only the chip's total power measurements and (internal or external) thermal sensor measurements to find the thermal model and fine-grain power consumption of the chip, without requiring knowledge of the thermal power model to identify sources [100]. Reda's BPI technique was evaluated using simulation software, real multi-core embedded sensors, and an IR camera, but the camera application was limited to a test chip with a 10x10 grid of microheaters for proof-of-concept validation. Werner et. al. created a thermal modeling framework for accelerator-rich architectures, which estimates the power consumption profile IC by solving the inverse of the heat transfer equation using information gathered from the thermal side-channel, which required images to be pre-processed to account for the spatial LPF effect due to thermal diffusion [101].

3.3.2.2 External Sensor

Although less frequent, some works either placed an external temperature sensor on a depackaged chip [92, 121], while others leveraged sensors embedded in a development board [94, 122].

3.3.2.3 Fan Speeds

Fan speeds, which have a strong correlation to board temperature, were occasionally considered as an acoustic and thermal side-channel. [107] leveraged the CPU fan's acoustic emissions in combination with CPU core temperature for a more robust embedded system monitoring capability. From a convert communications standpoint, [118] posited that since the fan's angular speed is software readable, it could be suitable to indicate temperature, and [105] considered the fan speed and its impact on implementing a bi-directional communications channel.

3.4 Temperature Attacks

The temperature side-channel is an ongoing area of investigation and is increasingly being considered as a vector for many types of temperature attacks; however, not all temperature attacks are side-channel attacks. "Temperature Attacks" in literature broadly fall into three categories: Temperature Side-channel Attacks (TSCA), Covert Channels, and Physical Attacks.

3.4.1 Temperature Side-channel Attacks

Temperature Side-channel Attacks take advantage of thermal leakage accessed through a number of possible sensors to infer system behavior from an active perspective. Typical TSCAs include cold boot [121] and fault injection attacks [92], as well as password cracking [108], and breaking encryption [92,94]. The most comprehensive analysis of TSCAs to date is in [94], which examines how to apply known power side-channel attacks that target asymmetric encryption algorithms to the temperature side-channel, and develops a new technique called "Progressive Correlation Thermal Analysis" (PCTA). Both Thermal and Power SCA techniques are found

to perform well in unprotected scenarios, resulting in 100% key recovery, but TSCAs are less successful when there are countermeasures in place.

3.4.2 Covert Channels

Covert channels leverage the temperature side-channel to transmit information. This type of temperature attack was first introduced in 2009 by Broucier et. al, who used both fan speeds and ring oscillators to create covert channels that leaked data at a very slow rate (1/3 bpm) [118]. These attacks tend to be more sophisticated, as they require a compromised program to modulate data onto a temperature-based carrier signal to achieve even one-way communications. Implementation of bi-directional communication becomes even more complex, as both the source and destination need to be able to transmit and receive the thermal signal. In 2011, [119] implemented temperature-based transmitter and receivers on an FPGA using ring oscillators (Section 3.3.1.2), which enabled covert communications both between separated internal FPGA blocks as well as to nearby FPGAs. Reliable two-way FPGA comms via the TSC were also explored by [112], who considered useful applications (such as IP watermarking and post-fabrication bug fixes) in addition to covert communications, and considered modulation techniques to increase the capacity of the covert channel. Masti used on-core thermal sensors to implement covert channels and explored channel throughput [97], while Guri et. al. demonstrated the "BitWhisper" method, which used built-in sensors for bi-directional covert communications between air-gapped (but adjacent) computers [105]. More recent works focus on increasing channel capacity through (relatively) faster communications schemes [106], using broadband modulation to overcome the interference of low-frequency thermal noise and create multiple channels [98], and implementing

thermal covert channels in FPGA-based cloud computing infrastructures [120].

3.4.3 Physical Attacks

Physical attacks are temperature-based attacks that do not leverage the side-channel at all, but rather aim to damage components sensitive to high temperatures. As early as 2005, Dadvar et. al. considered the potential security risks associated with the use of software to control operating temperature - especially if that program were to become compromised. If malicious code could control cooling fan operations, thermal monitoring sensors, and Dynamic Voltage & Frequency Scaling, the resulting implications to security could be catastrophic to the chip, resulting in the potential for denial of service due to thermal throttling, data integrity issues due to reduction of voltage, and gradual or permanent physical damage, either through accelerated aging or by disabling the thermal failsafe in BIOS [122]. Another example of using temperature to cause physical damage is to target areas of the processor that are further from on-chip sensors, as Kong et. al. demonstrated by using code designed to frequently access the instruction cache while avoiding functional units that are closer to on-chip temperature sensors, in order to drive the processor past its thermal limits and cause gradual damage due to accelerated aging or permanent damage due to thermal runaway [99].

3.5 Temperature Side-channel Defense

Defending against TSCAs can be considered by designing with thermal security in mind or by monitoring a device at run-time for anomalous behavior. These are the two main approaches captured in the literature.

3.5.1 Design-time Defense

Since TSCAs leverage the relationship between temperature leakage and sensitive data, there are two primary ways to defend this side-channel at design-time:

1. Eliminate the thermal leakage.

2. Eliminate the relationship between thermal leakage and sensitive data.

It is impractical to eliminate thermal leakage completely due to the nature of the thermal side-channel, so instead design-time defense focuses on de-correlating temperature from the underlying activity patterns. Gu et. al. introduce a dynamic shielding technique to Protect 3D ICs from TSCAs by using an algorithm implemented in the controller to proactively inject noise in order to conceal critical activities [62], while Knechtel et. al. propose a methodology that models thermal leakage and minimizes its correlation to power during floorplanning [93].

3.5.2 Run-time Defense

Run-time defense methods also aim to reduce thermal leakage to protect against TSCAs, as well as to monitor the thermal side-channel to detect malicious behavior.

As examples of trading performance to reduce thermal leakage, Bao et. al. proposed a secure algorithm for ordering aperiodic tasks with soft deadlines for temp-tracking-based power consumption SCAs, providing more uncertainty for an attacker attempting to infer chip state, while still maintaining a good balance between side-channel leakage and total lateness [102]. Similarly, Dey et. al. introduced a methodology targeted at embedded devices with Dynamic Voltage and Frequency Scaling enabled to reduce thermal gradient by finding the ideal reduced

voltage and frequency to spread work out over time to minimize the impact on performance [78].

The use of the temperature side-channel to detect hardware trojans and malware has been a popular research area. The approach of Forte et. al. [103] utilizes a framework for temperature-based hardware trojan detection at all phases of development, but targets run-time by using local thermal sensors and the global correlation between sensors and the thermal profile to detect trojan activation. Nowroz et. al. also focused on detecting hardware trojans at runtime through the use of spatial thermal maps, inverted power maps, and Principal Component Analysis [104]. To detect malware, Paul-Pena et. al. increased the usefulness of thermal side-channel information by dynamically modifying the process-to-core affinity and then processing information captured with ML algorithms to detect changes in code being executed on embedded sensors [107], while Werner et. al. simulated the detection of malware running on accelerator-rich architectures by using a deep neural network to analyze thermal images [101].

Chapter 4: The Micro-architectural Side-channel

This Chapter provides additional background on the Micro-architectural Side Channel (Section 4.1) as well as its access via Hardware Performance Counters (Section 4.2). Section 4.3 summarizes related background literature that considers the use of HPCs for security applications.

4.1 Micro-architectural Side-channel

Microarchitecture is the specific design of a microprocessor, and consists of all the digital logic, arithmetic, and data path and control circuits required to implement an instruction set in a given processor. The micro-architectural side-channel leverages the distinct hardware features inherent in the particular microprocessor design, particularly with regards to the time it takes to execute instructions, contention inherent in the sharing of hardware resources, and the functionality of the memory subsystem. In contrast to previous side-channels, this side-channel is software-based and does not necessarily require physical proximity to the target device.

Vulnerabilities, attacks, and defenses of the micro-architectural side-channel were recently surveyed in [123, 124]. Micro-architectural side-channels can be further categorized as being based on internal or external timing, as well as whether data (usually in the memory subsystem) is accessed directly, as described in Sections 4.1.1 and 4.1.2 below.

4.1.1 Timing Side-channel

Since most computations do not occur in constant time, information can be inferred about the operation taking place by the amount of time the operation takes, also known as the timing side-channel. The timing side-channel relies on measuring execution time as leaked information. If an observer measures their *own* execution time, it is considered an *internal* timing side-channel measurement, while observations of a *victim* execution time are *external* timing side-channel measurements. The vast majority of micro-architectural side-channels are timing channels [123, 125, 126].

4.1.2 Memory (Access) Side-channel

Memory, cache, and access side-channels are all terms used to reference a popular, specific application of an *internal timing* side-channel that access the memory cache directly [127–129].

4.2 Hardware Performance Counters for the Micro-architectural Side-channel

Hardware Performance Counters (HPCs) are special registers built into the Performance Monitoring Unit of modern processors that measure low-level architectural and micro-architectural events during process execution [130], related to processor and memory system operations (e.g. CPU clock cycles, total instructions retired, memory accesses, cache hits/misses, branches taken, etc.). This section provides information on various software tools used to access HPCs (Section 4.2.1), and additional detail on the Intel®VTune™ Profiler (Section 4.2.2), which was used to collect the HPCs analyzed in Chapters 7 and 8. Specific applications of hardware performance counters to malware and ransomware in literature are described in Sections 5.2.3 and 5.3.1,

respectively.

4.2.1 Profiling Tools

Profiling tools use software to dynamically analyze programs and relate performance metrics and measurements to source code execution. Commonly used profiling tools include:

- **`perf` and `perf_events`**. The `perf` command is the primary interface to the Linux kernel performance monitoring capabilities.
- **`LIKWID`**. `LIKWID` is an open source performance monitoring suite for Linux, and includes the `perfctr` tool to read hardware performance counters.
- **Open Hardware Monitor**. Open Hardware Monitor is open source software that monitors temperature sensors, fan speeds, voltages, load, and clock speeds of a computer.
- **Oracle®Developer Studio: Performance Analyzer**. The Oracle®Developer Studio software suite includes the Performance Analyzer tool to perform statistical profiling of performance data.
- **PAPI**. PAPI stands for Portable Application Programming Interface, and provides high-level interfaces for simple measurements as well as low-lowel event access [131].
- **Intel®VTune™ Profiler**. Intel®VTune™ Profiler is Intel's commercial performance analysis tool suite, which includes a command line interface that leverages Linux `perf`. VTune Profiler is described in more detail in Section 4.2.2.

4.2.2 Intel®VTune™ Profiler Primer

The following sections describe the sampling modes and analysis groups available through VTune Profiler.

4.2.2.1 Sampling Modes

VTune Profiler uses two collection modes, both of which are statistical sampling-based. This means that collected data is not 100% accurate because VTune Profiler collects an event, attributes a current collected event and the full prior sampling interval to the current code context (10k to 2M events). This sampling error is not statistically significant for large numbers of samples, but events that happened for very little time will have very few samples and should be disregarded. The two sampling-based collection modes are:

1. **Hardware Event-Based Sampling (EBS).** Hardware EBS analyzes all processes running on a system at that moment using the counter overflow feature of the PMU to interrupt a process and capture the instruction pointer of the process at the time of interrupt. It requires sampling drivers or Linux Perf and has minimum collection overhead (approx. 2% on 1 ms sampling interval). The number of hardware events that can be collected simultaneously is limited by the CPU, and is usually not more than 4.

2. **User-mode Sampling** - User-mode Sampling collection focuses on applications only and is not capable of gathering system-wide performance data. It does not require sampling drivers and has higher (approx. 5%) overhead than EBS.

4.2.2.2 VTune Analysis Groups

VTune Profiler includes 6 pre-configured VTune Analysis Groups, each of which consists of predefined analysis types with a limited number collection parameters that can be changed. VTune also provides the ability to create custom analyses where any collection options must be enabled manually. The predefined analysis groups and types, reproduced and summarized by Chapter 7 of the Intel®VTune™ Profiler User Guide [132], are:

- **Algorithm.** The VTune Algorithm group allows the programmer to tune software applications for time and efficiency. The VTune Algorithm Group consists of Hotspot, Anomaly Detection, and Memory Consumption analysis types.

  Hotspots. The Hotspots analysis type identifies sections of code that took the most CPU time to execute.

  Anomaly Detection. The Anomaly Detection analysis type identifies anomalies in frequently recurring intervals of code (e.g. loop iterations).

  Memory Consumption.. The Memory Consumption analysis type explores memory consumption over time and identifies which memory objects are allocated and released during an analysis run.

- **Microarchitecture.** The Microarchitecture Analysis Group estimates how effectively code runs on modern hardware.

  Microarchitecture Exploration. The Microarchitecture Exploration analysis type identifies the most significant issues affecting application performance.

Memory Access. The Memory Access analysis type measures a set of metrics to identify memory access related issues.

- **Parallelism.** The Parallelism analysis group introduces analysis types for compute-sensitive applications, such as Threading and HPC Performance Characterization.

  Threading. The Threading analysis type shows how well an application is threaded for the existing number of logical CPU cores on the system.

  HPC Performance Characterization. The HPC performance analysis type evaluates compute-sensitive/throughput applications for floating point operations and memory efficiency.

- **I/O.** The Input and Output Analysis group locates performance bottlenecks at hardware and software levels for I/O-intensive applications.

  Input and Output. The Input and Output performance analysis type determines platform I/O consumption by external PCIe devices, memory bandwith consumption, Intel Ultra Path Interconnect bandwidth consumption, and software data plane utilizations.

- **Accelerators.** The Accelerators analysis group consists of analysis types that monitor CPU, GPU, and FPGA usage for a system or application.

  GPU Offload. The GPU Offload analysis type profiles application that uses a GPU to determine whether an application is GPU or CPU bound.

  GPU Compute/Media Hotspots. The GPU Compute/Media Hotspots analysis type identifies performance issues caused by memory latency or inefficient kernel algorithms for GPU-bound applications.

CPU/FPGA Interaction. The CPU/FPGA Interaction analysis type identifies the most time-consuming FPGA computing tasks.

- **Platform Analyses.** The Platform Analysis group monitors application or system behavior and power usage.

  System Overview. The System Overview analysis type monitors the general behavior of the target system and identifies platform-level factors such as power usage and throttling that limit performance. This is a driverless, event-based sampling analysis.

  Platform Profiler. The Platform Profiler analysis type provides insights into overall system configuration by collecting data over an extended period of time on a deployed system running a full load.

4.3 Considerations When Using Performance Counters for Security Applications

Not all research has reported positive results regarding the use of performance counters for security applications [133]. Dinkarrao et. al. [134] developed attacks on Hardware-assisted Malware Detection by generating adversarial HPC samples that caused reduction in malware detection accuracy from 80% to less than 20%.

Zhou et. al. [135, 136] questioned the assumption that low-level microarchitectural events are causal of high-level software behavior, and identified five classes of drawbacks in related works:

1. Overhead introduced by the use of dynamic binary instrumentation tools, which made some solutions not practical for online protection.

2. Use of Virtual Machines, as challenges virtualizing HPCs have been observed [137].

3. Testing and training data overlap, which creates misleading results since training data cannot possibly include all malware to be encoutered in real-world applications.

4. Lack of cross-validation results.

5. Too few data samples, resulting in either over-generalized or over-interpreted results, as double-decimal precision requires at least 10 thousand experiments (1000 programs in 10-fold cross validation).

Zhou et. al. obtained a best-case F1-score result of 80.78% with 15% false positives when tested 962 malware samples from 35 families and 962 benign-ware samples in a non-virtualized set-up. They used Principal Component Analysis (PCA) to reduce dimensions and identify which (6) HPCs to collect, evaluated five ML models using 1,000 repetitions of 10-fold cross validation to prevent overfitting, and concluded that HPC-based microarchitectural information was NOT reliable to differentiate between benign applications and malware. This result was explained in [138], however, which found that the use of Principal Component Analysis for this application eliminates the correlation between data points in the HPC datastreams and was a major contributor to low performance.

In an analysis of security solutions for Cache/Access-based SCAs, Mushtaq et. al. [139] identified the following challenges in using HPCs for security:

1. Do not guarantee discernible information.

2. Non-deterministic behavior.

3. Multiplexing issues.

4. Performance overhead.

Meanwhile, Das et. al. [130] noted inaccurate measurements resulted from external sources, non-determinism, overcounting, and variations in tool implementations, and that the following issues needed to be considered to obtain accurate HPC measurements:

1. Context Switch Monitoring
2. Interrupt Handling
3. Process Filtering upon Interrupts
4. Minimization of Non-deterministic Event Impact

Even with the aforementioned considerations, a survey by Konstantinou et. al. [140] reviewed research milestones in the use of HPCs to track microarchitectural events and identified several HPC-based security solutions that have successfully transitioned to use in government and industry.

Chapter 5: Malware and Ransomware

This chapter provides brief overviews of malware (Section 5.1) and ransomware (Section 5.1.1), as well as related literature on side-channel based detection methods of each (Sections 5.2 and 5.3, respectively).

5.1 Malware Overview

Malware (*mal*icious soft*ware*) is a hostile application or portion of computer code that is meant to damage a system, its data, the network, or some combination thereof. Malware is the most expensive type of attack and annual costs are increasing rapidly [141]. The most common types of malware [142] are:

- **RANSOMWARE**. Ransomware holds user data hostage by encrypting valuable data or locking access to the system entirely and then demanding a ransom payment to obtain the decryption key. Even if the ransom is paid, there is no guarantee of key receipt or usefulness in regaining access to the system or user data. Ransomware is discussed in more detail in Section 5.1.1.

- **FILELESS MALWARE.** Fileless malware makes changes to an operating system's native files.

- **SPYWARE.** Spyware collects information such as passwords, pins, and payment information without the knowledge of the user.

  KEYLOGGERS. A keylogger is a type of spyware that logs keystroke to steal sensitive information.

- **ADWARE.** Adware is similar to spyware, but monitors user activity to determine which ads to display rather than capturing keystrokes. Adware captures data that can be used to create a profile of the user's internet activity, including where they spend money, who they talk to, and where they've traveled. This can all be accomplished overtly or without user consent, both of which degrade user privacy.

- **TROJANS.** Coined from Homer's Odyssey [143], which told the story of how Greek soldiers were able to sneak into the city of Troy by hiding in a giant wooden horse, this form of malware disguises itself as desirable software that users intentionally download. Once downloaded, the Trojan takes control of the system for malicious purposes.

- **VIRUSES.** A virus is self-replicating form of malware that spreads through networked computers. A virus is a piece of code that is inserted into an application and remains dormant on a host system until activated by a trigger. When the infected application is running the virus can execute and reproduce.

- **WORMS.** A worm is a self-replicating form of malware that spreads through networked computers. It is a standalone program that propagates independently as soon as it infects a system.

- **ROOTKITS.** A rootkit obtains full admin privileges of a victim computer that malicious

actors can use remotely.

- **BOTS and BOTNETS.** A bot is a software application that performs automated tasks on command for malicious purposes. Bots can be used in large numbers over networks called BOTNETS, which can be combined to remotely execute broad-scale attacks (e.g. Distributed Denial of Service (DDoS)).

5.1.1 Ransomware

Ransomware, characterized by demanding payment in exhange for the return of data, comes in two forms: Encrypting (or crypto-) ransomware and locking (non-encrypting) ransomware. Encrypting ransomware converts a users valuable files into ciphertext using various encryption methods, while locking ransomware denies the user access to the device but does not encrypt the files therein.

5.1.1.1 Ransomware Families.

Ransomware is usually identified by "family," with ransomware families sharing similar code signatures, payloads, and malicious commands. Common ransomware families include WannaCry, DarkSide, and REvil/Sodinokibi. The state of the art for current ransomware threats is dynamic, with roughly 200 ransomware families detected by security companies in any given month. However, the vast majority of ransomware attacks are built on less than two dozen ransomware families [144].

5.1.1.2 Ransomware as a Service (RaaS).

RaaS is a business model in which ransomware developers allow the purchase of ransomware to be used by the buyer in an attack. There are numerous forms of ransom profit sharing associated with RaaS, ranging from a one-time fee to a monthly subscription [145] .

5.2 Malware Detection via Side-Channel Analysis

A broad number of works consider Malware detection, analysis, and mitigation techniques [146–154], but relatively few consider side-channel information as an analysis vector. Of these, side-channel approaches for malware detection primarily focus on power measurements, EM measurements, and micro-architectural-based measurements obtained from performance counters.

5.2.1 Malware Detection with the Power Side-channel

Using power side-channel measurements, Clark et. al. developed *WattsUpDoc* to discover untargeted malware on embedded medical devices by monitoring wall outlet power. Meanwhile, Park et. al. proposed the power side-channel to analyze an embedded device in real-time at the instruction-level using ML and templates from a known original device in order to detect malicious activity [155].

5.2.2 Malware Detection with the EM Side-channel

Khan et. al. use EM side-channel emissions to recover individual branch decisions in embedded devices when there is enough training data and signal bandwidth, as well as a high

signal-to-noise ratio [156] and then propose a novel detection system that continuously monitors the EM side-channel for several types of malware behavior, including ransomware [157]. EM emissions were also leveraged in the work of Chawla et. al. to develop a ML methodology to remotely detect malicious activity on an embedded device and classify the activity into a malware family [158].

5.2.3 Micro-architectural Side-Channel for Malware Detection (via Hardware Performance Counters)

Since the micro-architectural side-channel encompasses timing and pipeline considerations of micro-architectural implementations, Hardware Performance Counters are useful in gaining micro-architectural insight. This section summarizes a number of works which have considered detection of Malware via the Microarchitectural Side-channel using Hardware Performance counters, while Section 5.3.1 covers those works which have specifically focused on applying HPCs to Ransomware detection.

The use of HPCs for malware detection in research spans over a decade. In 2011, Malone et. al. focused on static and dynamic integrity-checking of programs to detect malicious modifications [159], while Quinn examined cache timing information and the use of Hardware Performance Counters to detect the presence of rootkits on a system after training a classifier on a known "clean" system state in 2012 [160]. Other early research considered detecting control-flow modifying kernel rootkits in virtual machines [161], detecting common programming errors employed by malware (e.g. buffer overflows and return oriented program [162]), the feasibility of building a malware detector in hardware [163], and the use of unsupervised machine learning

approaches to detect deviations from normal program behavior that could indicate the presence of malware [164].

Since these initial works, numerous approaches have refined various aspects of malware detection via HPCs, from focusing on specific hardware (e.g. Andriod devices [165]) to developing a mathematical framework to find the probability of HPC-based malware detection given a program, sampling rate, and set of HPCs [166]. Wang et. al. reduced overhead associated with HPC-based detection by compressing HPC profiles before sending them to a remote server for analysis [167], while Khraisat et. al. propose a hybrid approach that combines HPCs with a low-level Intrusion Detection System [168]. Recent HPC-based approaches for "Hardware-level Malware Detection" (HMD) use behavioral signatures to instantly detect and disable malicious programs at the beginning of execution with low computational overhead [169]. Another recent HMD approach combines thread-level CPU-core telemetry profiling with global non-core profiling to achieve better performance than current HPC-only-based detectors [170].

Several works have developed HPC-based tools to distinguish the behavior of applications. "Perfextract" was found to accurately differentiate programs in physical *and* virtual environments based on HPC data captured during start up, and could then classify programs into a specific group (e.g. web browser, text editor, image viewer, audio player) based on intrinsic behavior [171]. Ramos et. al. developed an HPC-based tool to model, fingerprint, and compare behavior of applications for later classification [172]. While neither of these tools was directly tested on malware, the application of HPCs to malicious behavior is a logical extension.

5.2.4 HPC-based Machine Learning Classification

Machine learning has become instrumental in quickly detecting malware. Because of the ability to use benign and malware execution samples to train models to quickly learn and identify malicious behavior, many works have focused on the improvement of this aspect of malware detection. The "HPCMalHunter" tool used support vector machines for real-time behavioral malware detection and used singular value decomposition to reduce a collected set of HPCs into a program-specific behavioral signature [173]. Patel et. al analyzed HPC-based classification models in software and hardware to compare performance and design parameters [174], while Sayadi et. al. compared accuracy, robustness, performance, and hardware overhead of general and ensemble HPC-based classifiers identify benign and malware applications [175]. ML-based classification was incorporated in 3 step process for fine-grained malware detection by Kadiyala et. al., through extracting HPCs in each system call of an unknown program and then reducing dimensions by identifying HPCs with maximum variance [176]. Alternately, Elnaggar et al. used an extreme gradient boosting classifier (XGBoost) for improved malware detection soon after launch, and found better results *without* PCA-based dimension reduction [138]. He et. al. applied HPC-based ML classifiers to increase the probability of detecting zero-day malware events by using a two-stage approach that combines DNN and transfer learning [177]. Recent works have focused on explainable ML to order to increase transparency, such as Kuruvila et. al's explainable HPC-based Double Regression ML framework that isolated the most malevolent transient window of an application on two malware and five microarchitectural SCAs [178]. Pan et. al developed a framework using explainable ML that combined HPCs and embedded trace buffers to detect and localize malicious behavior [179].

5.3 Ransomware Detection

With the high costs associated with ransomware, the ability to detect and mitigate its effects is critical. Numerous ransomware detection approaches have been proposed at both hardware and software levels [180]. These detection methods include pattern mining [181], using honey pots [182], examining layered features [183] and "attention" features [184], as well as monitoring:

- file system IO [185–187],
- user data for (non-encryption) file transformation indicators [188],
- API calls [189],
- large integer arithmetic [190],
- file system traversal [191],
- using lock characteristics (specifically to detect Android-based locker ransomware) [192],

Other approaches focus on combining process monitoring with techniques [193], or system and network level activity such as OS-level file system [194] and using of Endpoint Detection and Response tools (EDR) [195].

Just a handful of works apply an HPC-based approach to ransomware detection, and comparison of these approaches is paramount in identifying the best methods for fast, accurate detection, with a low false positive rate and minimal overhead.

5.3.1 Ransomware Detection with Hardware Sensors and Performance Counters

A hardware-based approach, which can significantly reduce software overhead, is a particularly interesting application for ransomware detection because of the time sensitivity of device compromise. Hardware sensors and performance counters are available in all modern processors and provide low-overhead access for monitoring and measuring CPU performance information. There is limited research that examines sensors and performance counters for ransomware detection, all of which have limitations in their approach. Most use data from just a few sensors [196–199], and do not attempt to distinguish ransomware from standard encryption operations [200, 201] or a representative real-world workload [196, 197, 199–202]. These works are detailed below.

In [200] and [201], Taylor et. al. use onboard sensor data through Open Hardware Monitor in lieu of actual HPCs to detect 4 variations of AES encryption as well as XOR encryption to simulate a ransomware attack. Training data was collected on two laptops: an older Windows machine (using 9 sensor values, 8 of which directly measure the CPU) and a 2013 MacBook (using 67 sensor values, only 17 of which measure CPU activity directly), and trained 12 different machine learning algorithms using feature vectors of system sensor data. A test script randomly launched encryption events while sensor data was continuously polled; if no encryption operations were detected the model predicts "no attack," while the presence of encryption operations indicates a simulated ransomware "attack." Prediction accuracy was 94% on Windows and 98% on Mac OSX. Twelve machine learning algorithms were tested and ranked by average time to attack recognition, and 10/12 models recognized the "attack" in less than a second for the many-sensor feature vector (OSX), whereas the 9-sensor feature vector (Windows) took over a minute to

detect attacks in 2/3 of the models. While the side-channel approach is advantageous in that it does not require prior knowledge of ransomware signatures and therefore has the potential to detect previously unknown "zero day" attacks, the experiment itself does not involve real-world ransomware at all.

Alam et. al introduced RATAFIA [196] followed by RAPPER [197], a generalized, two-step unsupervised ransomware detection framework, which used a watchdog program to collect selected perf stat HPC events (instruction, cache-references, cache-misses, branches, and branch misses) every 10 ms, to learn the "normal" behavior of a system under observation. It then used FFTs and an artificial neural network to flag potential ransomware activity and then distinguish whether it is non-malicious or ransomware, at which point ransomware processes are terminated. This approach used a sandbox Linux system and 4 different ransomware samples to establish a detection threshold, and was tested on WannaCry ransomware only, which it was able to detect and then terminate in less than 5 seconds. The major limitations of this work are the use of a limited number of HPCs and evaluation of the detection framework on a single ransomware sample.

"RanStop," was presented by Pundir et. al. to detect crypto-ransomware at runtime [198]. RanStop was developed in a Windows VM running on a Linux OS and is based off a created database of 80 crypto-ransomwares obtained from VirusShare and 76 benign programs consisting of encryptions algorithms and random C programs. It used open source `likwid` to obtain HPC data at 100 μs intervals for 2 ms for all programs in the database and uses pre-processed metric information for feature selection (in lieu of often-noisy raw hardware event counts that require pre-processing) and is run multiple times to collect all HPC metric information for the sixteen event groups in `likwid` (Branch, Clock, Cycle Activity, Data, FLOPS_DP, ICACHE,

L2.DATA, L2.CACHE, L3.DATA, L3.CACHE, TLB.DATA, TLB.INSTR, UOPS, UOPS.EXEC, UOPS.ISSUE, UOPS.RETIRE). A Long Short-Term Memory-based Recurrent Neural Network is then trained to generate a predictive model to distinguish ransomware from "goodware" and found to have high (>95%) accuracy when tested by witholding different percentages of the training data. While this approach uses more robust performance counter information and a greater number of ransomware samples, it relies on virtualized HPCs which have mixed results in representing actual HPC values obtained on system hardware.

Aurangzeb et. al. [202] used Windows running in a virtual machine and collected 13 HPCs events using `perf` and examined the ability to perform binary classification of 160 randomly selected Windows-based malware into "ransomware" and "non-ransomware" using four different classifiers. While this work found that Random Forest and Extreme Gradient Boosting performed best, it is limited by the lack of any benign executables and by collecting only virtualized performance counters.

The MS thesis of Podolonko [199] uses a non-virtualized Windows system with realistic programs and Vtune HPC event data running software that reverts the system to its initial "clean" state at power off. This work utilizes a combination of real ransomware samples, modified ransomware samples, and benign programs and tests detection prior to the 1st encryption operation using 3 classifiers. While it seems to most closely resemble the proposed approach, the limited detail provided on the data collection and evaluation process makes it difficult to discern how HPC data was used and results were obtained.

Most recently, the "DeepWare" [203] ransomware detection model used Linux textttperf to collect five performance counters (instructions, cache references, cache misses, branches, branch misses) every 10 ms on a Windows Guest in virtual machine. The resulting five HPC event time

series were stacked horizontally using a three phase process to create a behavioral image. The resulting dataset included 420,000 behavioral images, consisting of 50% ransomware obtained from 512 ransomware samples from 21 different families. DeepWare uses a deep learning-based Convolutional Neural Network due to its effectiveness in extracting features from images, and demonstrated 98.6% recall against known and unknown ransomware samples.

Chapter 6: Temperature Side-channel Analysis Experiments

This section describes two studies to access the temperature side-channel with a thermal camera for additional analysis. Section 6.1 outlines progress in identifying file operations (read/ write/idle) to a solid state drive (SSD). Section 6.2 describes experiments conducted using different thermal cameras to identify simulated malware on a single board computer. TSCA conclusions drawn from the results of both studies are provided in Section 6.3.

6.1 TSCA Study 1: Temperature Side-channel Analysis to detect File Operations on an SSD with Thermal Imaging

This section describes initial work to examine the correlation between the temperature of SSD components and SSD operations (read, write, and idle) to determine which processing techniques allow us to infer meaningful information from a series of thermal images [204].

6.1.1 Experimental Set-up

The four SSDs in Table 9.2 were examined, with training data collected concurrently with the Section 9.2 study to enable future comparison of side-channel techniques. The thermal data was collected as video using a thermal FLIR A325 camera, that captures 16-bit grayscale images at 30 frames per second. Then the thermal video was processed in the FLIR Research Studio

Suite to obtain individual frames. Figure 6.1 shows the experimental set up on the right, with the inset on the left providing a close up of FLIR camera pointed at an SSD during testing.

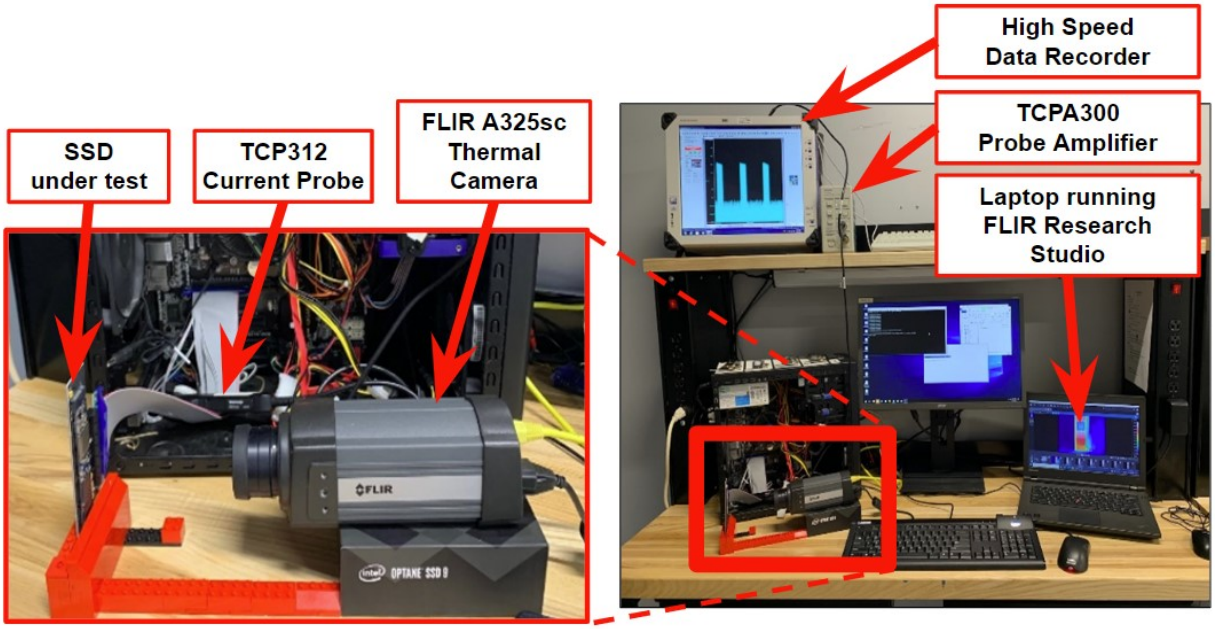


Figure 6.1: Experimental set up to capture thermal images of SSDs with a FLIR A325sc thermal camera.

Data was collected using the FLIR Camera. For each drive, a 30 second thermal video was captured for each operation of reading and writing a 100, 500, or 1000 MB file, as well as during idle stages. Twenty trials, in randomized order, were conducted for idle, read, and write for each file size per SSD. MATLAB was then used to process the images to reveal more defined characteristics. To illustrate the distinguishing features of thermal imaging, Figure 6.2 shows a series of three static images for a Samsung SSD during its idle, read, and write stages. On the write image, key components of the SSD are indicated. In these three images there are distinct differences between the idle, read, and write operations, specifically on the NAND flash memory and the Power Management Integrated Chip (PMIC). The SSD controller is covered by a heat sink, thus making it an indistinctive feature for operation classification purposes. The Double

Data Rate 4 Synchronous Dynamic Random-Access Memory (DDR4 SDRAM) has different heat signatures between the read, write, and idle, clearly differentiable between the write and read or idle; however, the DDR4 SDRAM is not as noticeably different between the idle and read. These differences will be utilized to classify the different operations based on the intensity levels in the thermal images converted to grayscale pixels [204].

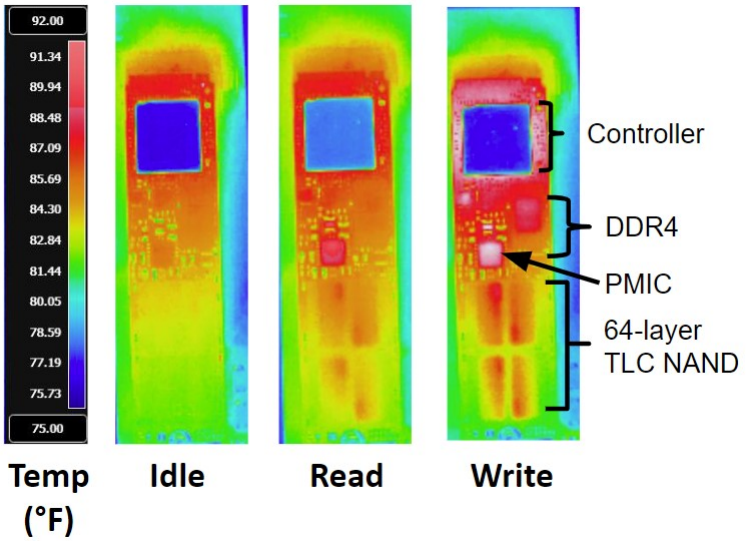


Figure 6.2: Thermal images of a Samsung 970 EVO Plus SSDs during idle, read, write, operations. All images were captured with a FLIR A325sc thermal camera.

6.1.2 Classification Results

In order to better classify the operations, the frames were first cropped using edge detection to uniformly produces frames that were 210x76 pixels. Then the frames were averaged pixel by pixel over 0.5-second, 1-second, 2-second, and 3-second windows. This was attempted based on the current-draw side-channel analysis insights from Section 9.2 and included in Figure 6.3, which showed that the majority of activity occurs in the few seconds after receiving the execution command. Then the MATLAB Classification Learner was used to determine which classifiers

might yield the best results based on cross-validation accuracy [204]. The following there were selected for additional analysis:

1. Tree with 10 branches

2. Cubic Support Vector Machine (SVM)

3. K-nearest neighbor (KNN) with a K value of 1

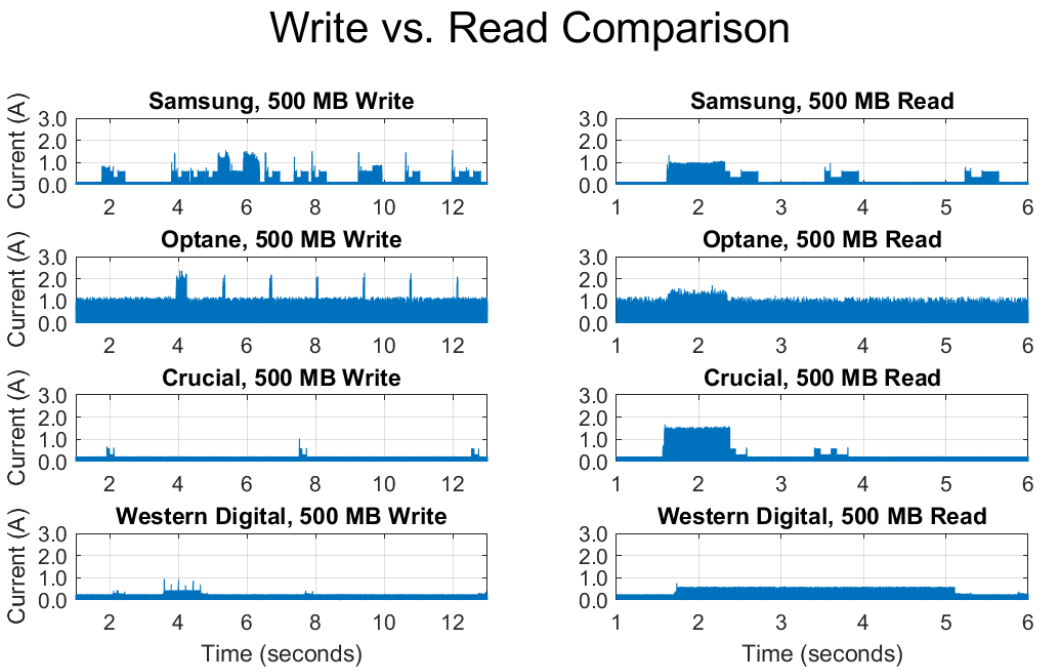


Figure 6.3: Thermal images of a Samsung 970 EVO Plus SSDs during idle, read, write, operations. All images were captured with a FLIR A325sc thermal camera.

6.1.2.1 Classifying Activity

Initial tests focused on the ability to classify activity (read/write) versus nonactivity (idle). The results for all drive, operation, and file size combinations are shown in the representative confusion matrix in Figure 6.4, with the actual classes on the y-axis and the predicted classes on

the x-axis. For non-activity, the classifiers correctly predicted idle 100% of the time across all four SSDs. The classifiers accurately predict that the SSD was in an active state with an accuracy approximately 95% across all four SSDs. The ability to detect activity versus non-activity shows promise as these results indicate it is possible to differentiate activity using the temperature side-channel [204].

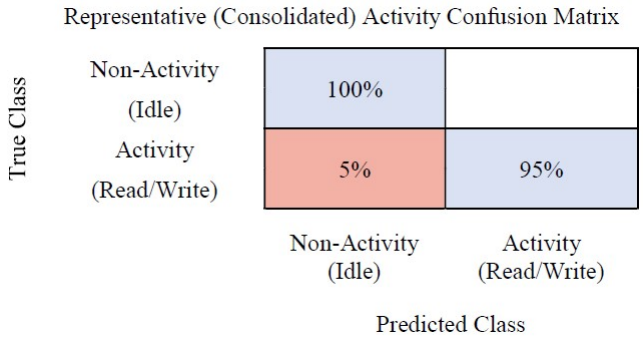


Figure 6.4: Consolidated Confusion Matrix showing the ability to differentiate activity (read/write) from non-activity (idle) for all SSD, operation, and file size combinations for the 3 classifiers trained and tested.

6.1.2.2 Classifying Operations

With promising results using a pixel-by-pixel average over a short period of time, the ability to identify drive operations was examined. Figure 6.5 shows the ten-fold cross-validation results organized by classifier and file size, with SVM results across the top row, KNN results in the middle row, and Tree results in the bottom row. File size increases from left (100 MB) to right (1000 MB) for each row. The x-axis of each graph shows the window size in seconds over which pixels from each frame were averaged, and the y-axis gives the accuracy percentage. The cross-validation data shows promise, as at least three SSDs are above 80% and all four perform better than chance (33.3%).

Initial results to classify drive operations are presented in Figure 6.6, which is organized the

same way as the previous figure, with each row dedicated to an individual classifier and file size increasing from left to right. The accuracy of the test results are below 80% with few exceptions, which is likely due to the classifier overfitting to the training data. This is particularly evident in the tree classifier across the bottom row, where cross-validation results were above 60% but test data accuracy does not surpass the chance threshold of 33.3% (shown in Figures 6.5 and 6.6 as a dashed line. This indicates that a 10-branch tree is not an effective classifier for limited pre-processing, and that follow on work is required to address overfitting across all four SSDs [204].

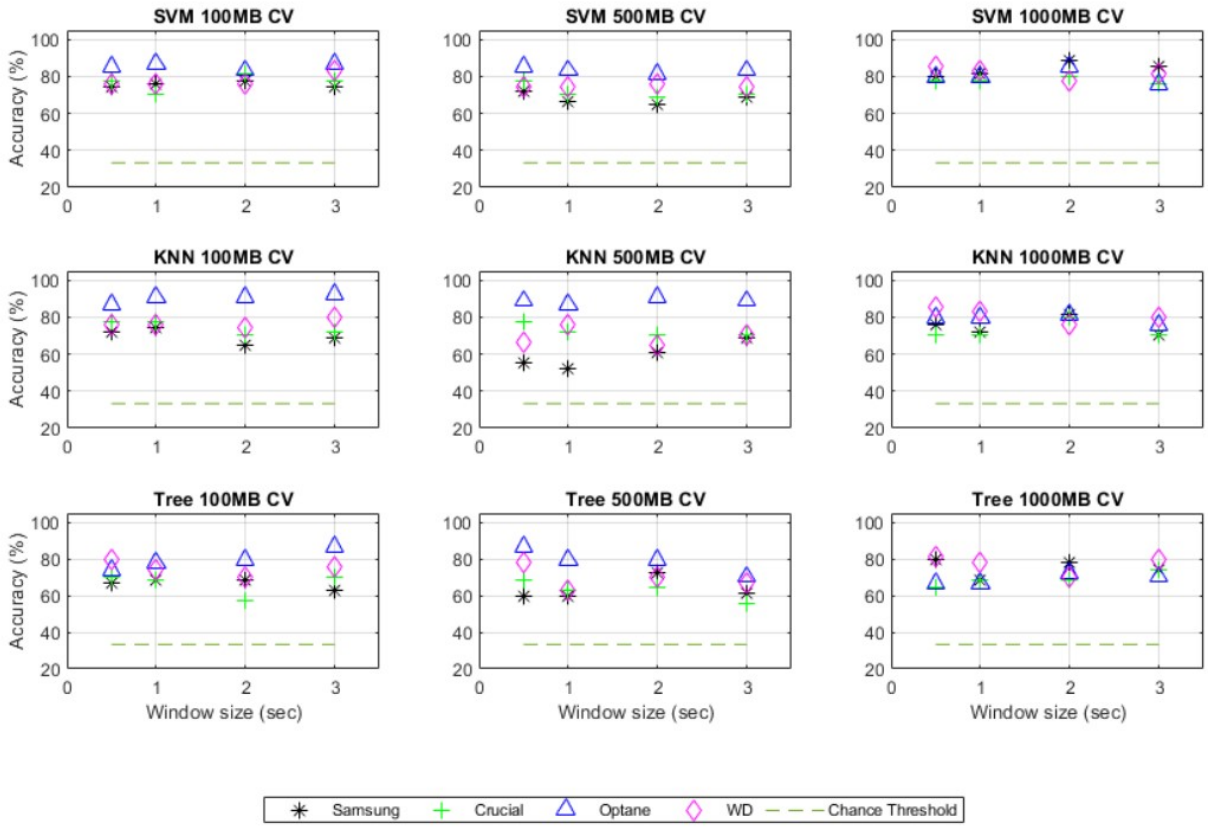


Figure 6.5: 10-fold Cross Validation Results for 3 classifiers trained with read, write, and idle operations at file sizes of 100, 500, and 1000MB. Each row shows results for a single classifier at increasing read/write sizes.

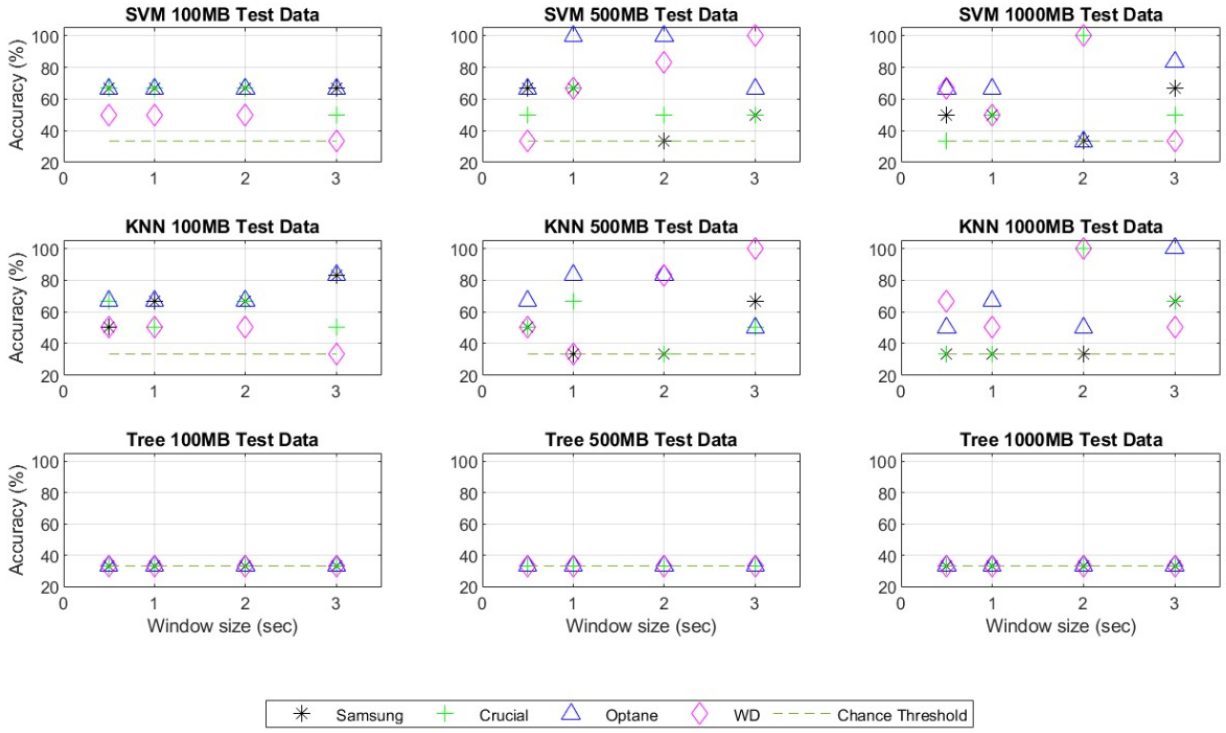


Figure 6.6: Test Data Results for 3 classifiers trained with read, write, and idle operations at file sizes of 100, 500, and 1000MB. Each row shows results for a single classifier at increasing read/write sizes.

6.1.3 Conclusions and Future Work

By developing a preprocessing technique of using single time windows, it is possible to not only identify whether the SSD is active or not, but also classify SSD operations of idle, read, and write via the temperature side-channel. For all four SSDs, the classifiers were able to determine that the SSD was not active 100% of the time, while concluding that the SSD was active at least 94% of the time. Three of the four SSDs (Intel, Samsung, and Western Digital) had accuracies above 80% across the different file size types. While this work is promising, there are many ways to improve the accuracy rates such as smaller window sizes, or moving to spatial variance to crop the frames. This work also confirms the need for preprocessing in order to better understand thermal imaging data. By setting a baseline for how to classify SSD operations using

thermal imaging, further work can be conducted looking at other SSD operations or even security practices.

While initial results are promising, more work is required for consistent SSD operation classification via the temperature side-channel. This should include additional image pre-processing, such as:

1. Inclusion of different window sizes (e.g. 0.25s, 0.75s) since this study showed the best results with a window within the first second of an operation.

2. The use of spatial variance, by comparing which regions within the thermal image have the most variance, and then only classifying based on these regions of the SSD. This approach will likely require further cropping of the frame to only the areas of high temperature variance, thus reducing the total pixels being analyzed as well as eliminating areas that show minimal temperature changes.

6.2 TSCA Study 2: Temperature Side-channel Analysis to detect Simulated Malware on a Single Board Computer

This study explored the ability to use temperature side-channel signatures collected with thermal cameras as a method to detect anomalous computations on a standard embedded architecture, like a BeagleBone Black [1] board running the default version of Ubuntu 18.04 and a Raspberry Pi 3 model B. To get heat readings we use three different thermal cameras: One small, lower-resolution, inexpensive, phone-app based, and two larger, higher resolution, more expensive, with an accompanying software suite. We obtain the dataset by capturing videos from

[1] `https://beagleboard.org/black/`

the cameras' respective applications and show that using thermal features one can detect file writes with high accuracy. The study in this section along was follow-on results was published in [205].

6.2.1 TSCA Study 2 Methodology

This section describes the experimental set-up to evaluate the performance of our manual and automatic file write detectors, as well as our ability to detect malware in simulated IoT devices.

6.2.1.1 Experimental Setup

The components involved in generating, collecting, and analyzing thermal side-channel data are described in the following sections.

Single-Board Computer These experiments were performed on a BeagleBone Black SBC, as described in section 6.2.2. Figure 6.7 shows a diagram of the system set up, in which a laptop is connected to the board with a Secure Shell (SSH) over a USB cable, from where we write, compile and execute our programs.

The SBC is controlled through a series of Python scripts to write files of sizes varying from 1000KB to 25KB to the BeagleBone's flash storage, from a laptop running Linux Ubuntu 18.04.5.

Figure 6.8, shows the actual set up using the smallest of the thermal cameras. The white rectangle highlights the board connected to the laptop, while the thermal camera is shown in the green rectangle. Similar set ups are used for the larger cameras, with the board rotated to the

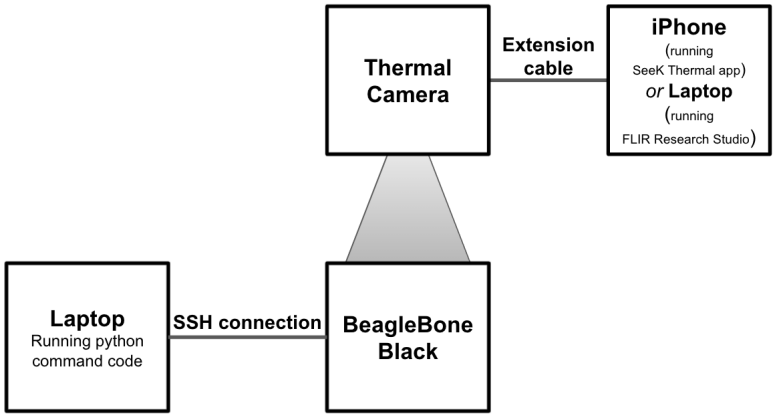


Figure 6.7: System diagram.

vertical plane so the camera can be placed alongside to record.

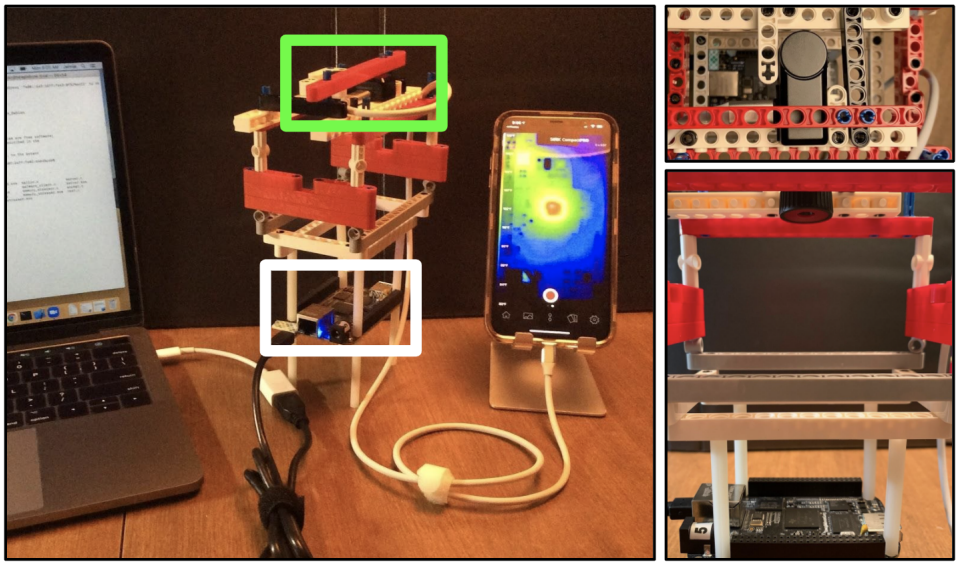


Figure 6.8: Measurement system setup. The green (upper) box in the left image indicates the thermal camera, while the white (lower) box highlights the single board computer. The right images show the top and side views of the camera and board set up.

6.2.1.2 Thermal Cameras

Two of the three thermal cameras, as shown in Figure 6.9, with the characteristics indicated in Table 6.1, were chosen to collect data, due to the different resolution, thermal sensitivity,

Table 6.1: Comparison between the thermal cameras used in experiments.

	FLIR A325sc	**SeeK CompactPRO**	**FLIR T530**
Resolution (pixels)	320x240	320x240	320x240
Temperature Range (°C)	-20 to 350	-40 to 330	-20 to 650
Thermal Sensitivity (mK)	50	<70	<50
Frame Rate (Hz)	30	9	30
Price (USD)	10K	500	11K

and frame rates available at different price points of approximately (a) $10,000, and (b) $500, respectively. To maintain a stable video feed, the thermal camera is placed in a fixed position beside (a) or above (b) the board, so that the majority of the board is within the camera's field of view.

FLIR A325sc. The FLIR A325sc with a 10 mm 45° IR lens is connected via ethernet cable to a laptop running FLIR Research Studio version 1.1.3, which we use to record the file operations generated on the SBC. The minimum and maximum values of the resulting images and videos can be adjusted through the vendor software after recording.

SeeK CompactPRO. The SeeK CompactPRO is connected by a lightening extension cable to an iPhone running the official SeeK Thermal application version 2.2.7.0. We also set a threshold for the min and max values of the heat spectrum to keep the measurements consistent across different experiments at run-time.

FLIR T530. The FLIR T530 is connected via USB cable to a laptop running FLIR Tools, which we use to record the file operations generated on the SBC. The minimum and maximum values of the resulting images and videos can be adjusted through the vendor software after recording. Since the results from the FLIR T530 were indistinguishable from the FLIR A325sc, the FLIR T530 was not used for write event detection or malware analysis.

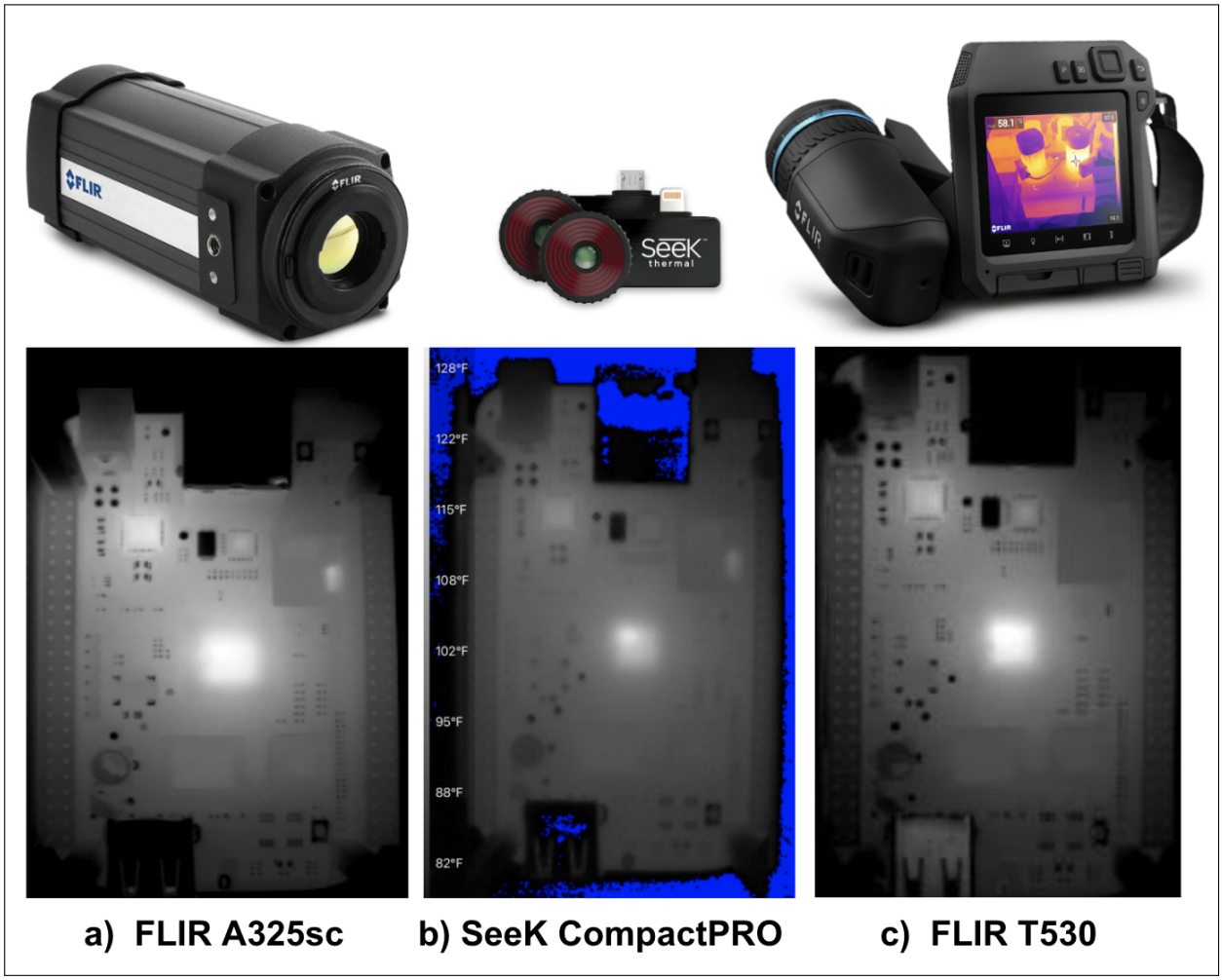


Figure 6.9: Comparison of images captured with (a) FLIR A325sc, (b) Seek Thermal CompactPro, and (c) FLIR T530 cameras. FLIR image temperature scaled to 75-135 °F with FLIR Research Studio software. SeeK CompactPRO scaled to 75-135 °F with SeeK Thermal iPhone app. FLIR T530 scaled from 80-110 °F in FLIR Tools. Blue in SeeK image indicates temperatures $< 75°$.

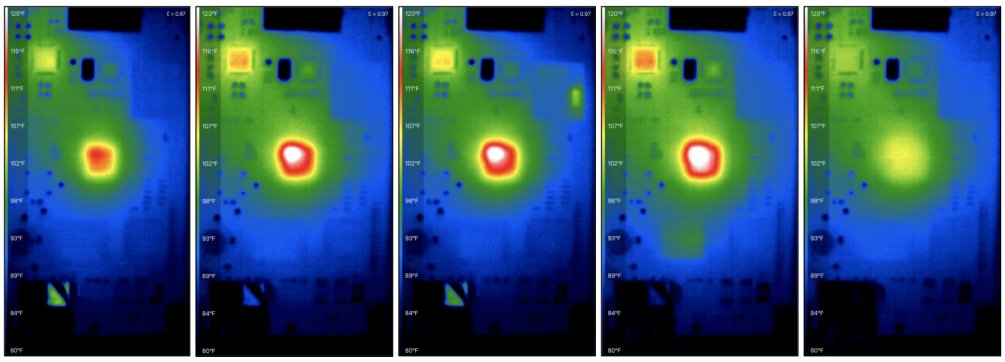


Figure 6.10: File operations performed on a BeagleBone Black, as observed with a SeeK CompactPRO thermal camera.

6.2.2 Thermal Signature of Write Operations

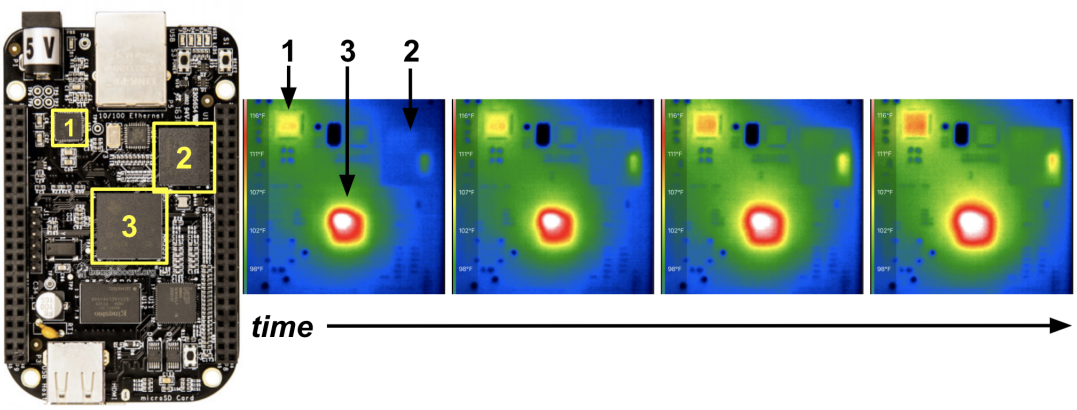


Figure 6.11: Thermal signature of a 100 MB file write over time. The left image shows a BeagleBone Black with three components highlighted: (1) Power Management Integrated Chip, (2) Embedded Multi Media Card, and (3) Processor, all of which produce distinct heat signatures over the course of the write operation.

The SBC contains all the necessary components to operate, namely a power manager (1), 4GB 8-bit embedded Multi Media Card (eMMC) on-board flash storage (2), a Sitara AM3358 ARM processor (3), and 512 MB DDR3 RAM. It also runs the default Ubuntu 18.04 along with the GNU C and C++ compilers and Python 2.7.18. We focus on file writes, which are the most common operation in malware [206]. Figure 6.11 shows the board with these components

indicated, as well as a series of thermal images of the board's heat signature over the course of a file write.

All three chips provide clear visual indications of write activity, as each gets noticeably hotter over milliseconds. The eMMC flash storage is of particular interest as it only heats up during file writes, lending credibility to the hypothesis that the thermal signature could be used to identify the presence of malware.

6.2.2.1 Detecting File Write Operations

File write operations involve writing some number of bytes into the on-board solid state drive (SSD)/eMMC of the BeagleBone. This write activity causes the SSD chip to heat up, as shown previously in Figure 6.11. Therefore, to demonstrate the capability of the thermal detector, we focus on detecting the occurrence of heat variations in the SSD. We consider any file write operation anomalous, for the following reasons:

- If we can detect any file write, we can detect out-of-order malicious file writes with another layer of anomaly detector.
- For certain devices (i.e., temperature sensors), file writes do not occur during the normal operations.

For this experiment, the assumption is that the defender will be able to record normal operation of the device. We also assume that the thermal video is collected from the same location relative to the SBC. While various proposed techniques can perform video aligning, those remain out of the scope.

We perform 20 file writes for files sizes from 1000KB - 100KB in decrements of 100KB,

and then 75, 50, and 25KB. These file write event sizes are chosen to be equal to or less than the average malware file size (in range of hundreds of KB) [207]; in fact, Lua malware are even larger than 1MB. We capture thermal video with both the FLIR A325ac and the SeeK CompactPRO cameras. We then apply both automatic and manual file write detection techniques and measure the false positives (FP), false negatives (FN) and true positives (TP). We use the metrics of accuracy, precision, and recall, to evaluate write detector performance of both automatic and manual detection techniques using both SeeK and FLIR cameras.

Accuracy ((TP + TN) / total samples) is the fraction of correctly labeled samples, and is shown in Figure 6.12.

Precision (TP/(TP + FP)) quantifies the correct positive predictions, which indicates the resistance to false positives. Figure 6.13 shows precision.

Recall (TP/(TP + FN)) is the fraction of correct positive predictions out of all possible positive predictions, and measures resistance to false negatives. Recall is shown in Figure 6.14.

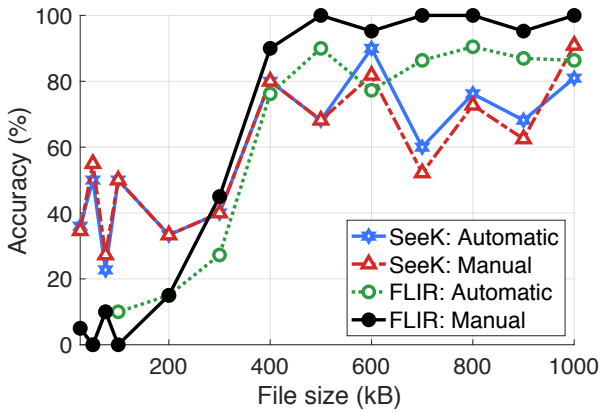


Figure 6.12: Accuracy of SeeK Compact Pro and FLIR A325sc cameras, using both automated and manual detection techniques.

In all three categories, Figures 6.12-6.14 show that manual file write detection using the FLIR thermal camera has the highest performance at file sizes $>$ 300 KB, and is at or above

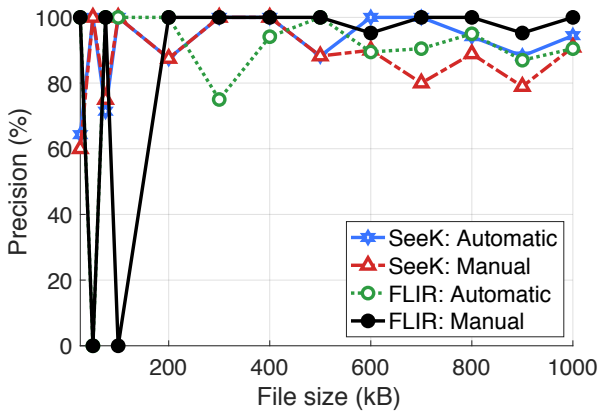


Figure 6.13: Precision of SeeK Compact Pro and FLIR A325sc cameras, using both automated and manual detection techniques.

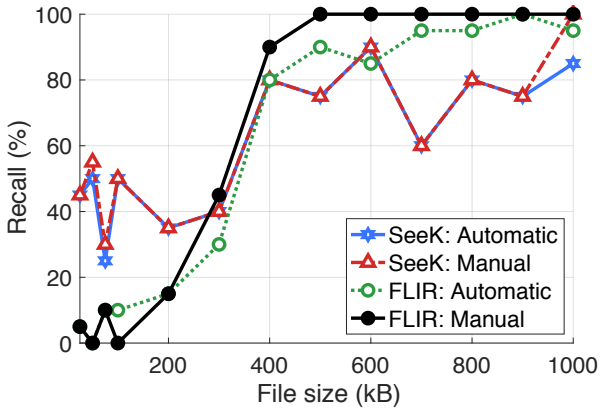


Figure 6.14: Recall of SeeK Compact Pro and FLIR A325sc cameras, using both automated and manual detection techniques.

90% at file sizes of 500 KB and above. The automated file write detection technique with the FLIR camera is just under the manual technique at high file sizes, but suffers from the same poor performance for file writes $\leq$ 300 KB. This result was surprising, as smaller file writes could be visually observed in the FLIR Research Studio software when the temperature was scaled appropriately. This may be due to difficulty in interpreting the files outside of FLIR's specialized software, and requires further evaluation.

Conversely, the SeeK camera, at just a small fraction of the FLIR's price point, averaged

around 70% accuracy at file sizes $> 300KB$, and outperformed the FLIR at detecting file writes $<= 300KB$, with accuracy ranging between roughly 30 and 50%, precision above 60%, and recall comparable to accuracy. While the FLIR software does provide additional processing options that can be explored, there's a clear trade off compared to the much less expensive SeeK, if slightly lower detection performance is tolerable.

Regardless of thermal camera, manual write detection slightly outperformed our automated write detector in both accuracy and recall, although additional evaluation of the automated procedure or alternate ML techniques would likely narrow the margin.

6.3 TSCA Experiment Conclusions and Future Research Directions

In summary, two proof-of-concept experiments that leveraged temperature side-channel measurements collected with a thermal camera were conducted and published [204, 205]. The first focused on detecting and distinguishing (read, write, and idle) file operations on a solid state drive, while the second focused on detecting "malware" (characterized by file write operations) on a single board computer.

Overall, observations of Temperature Side-channel Analysis were consistent with the known advantages and disadvantages of the temperature side-channel described in Section 3. While TSCA provides the ability to gain 2D insight both on and off system into device operations, the time delay associated with heat diffusion and image pre-processing requirements do not make it a viable approach for time-sensitive ransomware detection alone. Temperature could still be viable as a generalized approach to malware detection on devices with overhead constraints, as the benefits of inferring device operations without direct access may be worth the trade off in

associated time delay. Additionally, on- or off-system temperature measurements have not been explored as a supplemental set of side-channel measurements in a multi-channel approach. Both are left as future work.

Chapter 7: Investigating Vtune HPCs for Ransomware Detection

This chapter outlines the series of four initial experiments (two proofs-of-concept and two full studies) to safely collect Vtune HPC data during ransomware execution, automate collection across platforms, incorporate "real world" program samples against which to compare ransomware, and analyze collected data to identify dominant HPCs to aid in reducing the number of HPC features required. The study presented in the following Chapter (8) directly builds on the results presented in this Chapter.

7.1 Proof-of-Concept A: Vtune Data Collection of Ransomware Trials

This section describes the proof-of-concept effort to use Vtune Profiler to collect Hardware Performance Counter measurements of live, real-world ransomware on computer hardware. The parameters of this proof-of-concept are summarized in Figure 7.1 and described in detail in the following sections.

7.1.1 Hardware Setup

The hardware set up consists of a test computer enabled to execute ransomware with the Vtune Profiler performance analysis tool installed to enable hardware performance counter collection. The collection apparatus is shown in Figure 9.19.

# Trials (per operation)	Collection Method	Restore Process
1	MANUAL	Redo Rescue

Trial Parameters		
OS idle (current only)	30	s
Vtune idle (add Vtune)	0	s
Test Operation Duration	90	s

Data Recorder Parameters		
Sampling Rate	200	kS/s
Duration	120	s
Span	2	A
Offset	0	A

Operations Tested (5 total)		
Benign	**Ransomware**	
os	DarkSide	Spring '21
7zip	REvil/Sodinokibi	Spring '21
	WannaCry	Spring '21

Vtune Parameters		
HPCs	108	UA exploration
Sampling Interval	0.05	ms
Duration	90	s
Report "bins"	35,000	
Effective Sampling Rate	~ 0.0026	s

Figure 7.1: Data Collection Summary for Proof of Concept A

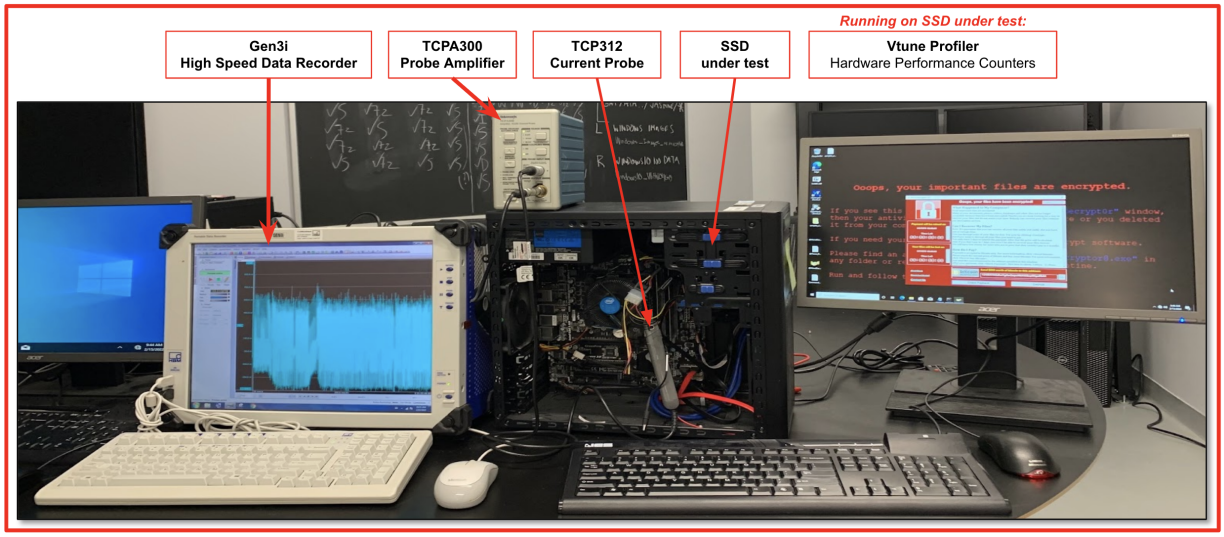


Figure 7.2: Collection apparatus. Includes the data recorder and current probe/amplifier to measure current supplied to the drive, but those capabilities were not incorporated into the proof-of-concept Vtune collection.

7.1.1.1 Testbed Computer

The testbed computer was an Intel Core i5-4590 System with 8 GB DDR3 running Windows 10 Pro 10.0.19043.928 Version 21H1 Experience Pack 120.2212.551.0. The system largely reflected an off-the-shelf configuration, with the exception of Windows Defender being disabled by setting the "DisableAntiSpyware" flag to 1 through the Registry Editor to enable execution of ransomware.

7.1.1.2 Ransomware

Live ransomware samples of WannaCry, DarkSide, and REvil/Sodinokibi obtained from VirusTotal in May 2021 were manually transferred from a zipped, password protected external hard drive, which had to be removed prior to initiating any trials to prevent corruption of the external drive by the ransomware.

7.1.1.3 Vtune HPC Measurements

Vtune Profiler performance analysis tool is run from the command line interface to collect micro-architecture exploration analysis type data (which uses hardware event-based sampling collection) with the following flags set:

1. **Finalization Mode: Full.** Vtune Finalization occurs automatically when collection completes to convert the collected data to a database format. "Fast" mode is default, but "full" gets the most accurate results by performing finalization on unchanged sampling data (which also requires more time and resources).

2. **Sampling Interval (Knob): 0.5.** The Sampling Interval Knob configures the amount of wall-clock time waited before collecting each sample. The default for hardware event-based sampling types is 1ms, but a number between 0.01 and 1000 can be specified.

3. **Analyze System.** The analyze system flag enables analysis of all processes running on the system rather than a specific application (default).

4. **Duration: 150 (seconds).** The duration flag sets the duration of data collection in seconds.

Vtune allows for the creation of custom CSV-formatted time reports sorted by event type, which can be analyzed off-system.

7.1.2 Data Collection Procedure

The proof-of-concept consisted of collecting a single trial of system "idle" (background OS activity only), 7zip encryption of a large (11GB) file, and execution of each type of ransomware identified in Section 7.1.1.2. Encryption and ransomware execution was performed through the Vtune CLI to synchronize HPC collection with the start of the trial. 7zip encryption and 1 type of ransomware (Sodinokibi) were selected for a randomized collection series of 10 trials of each. The Vtune data is converted to a custom CSV-formatted time report sorted by event type and then moved off system for additional analysis using a safe transfer procedure. After ransomware trials, the drive is reformatted and restored to its pre-trial state as described in Section 7.1.3. The data collection procedure is depicted in Figure 7.3.

7.1.3 System Restore Process

Because the execution of ransomware causes the system to become compromised, a safe data transfer process is required to move the VTune `.csv` reports from the corrupted OS SSD to a clean drive for further data analysis after each ransomware trial. The OS SSD is then reformatted and a clean image of the OS and data is restored using Redo Rescue[1].

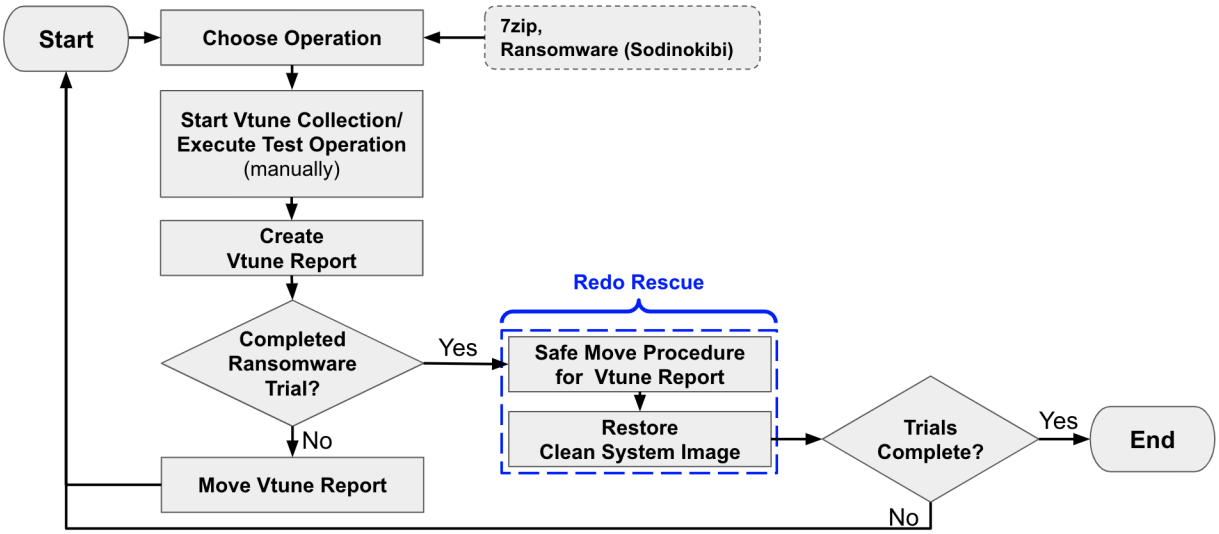


Figure 7.3: Data Collection and System Restore Procedure for manually executed 7zip and Sodinokibi randomized trials.

7.1.4 Initial Observations

Cursory analysis of the mean even rate for performance counters obtained from 7zip encryption (black) compared to REvil/Sodinokibi ransomware (red) shows a clear difference of approximately 1000 events per microsecond (or more) for over a dozen HPCs, as evident in Figure 7.4, which is a promising indicator of potential for classification.

Without Windows Defender disabled, WannaCry would not execute and would be automatically

[1] http://redorescue.com

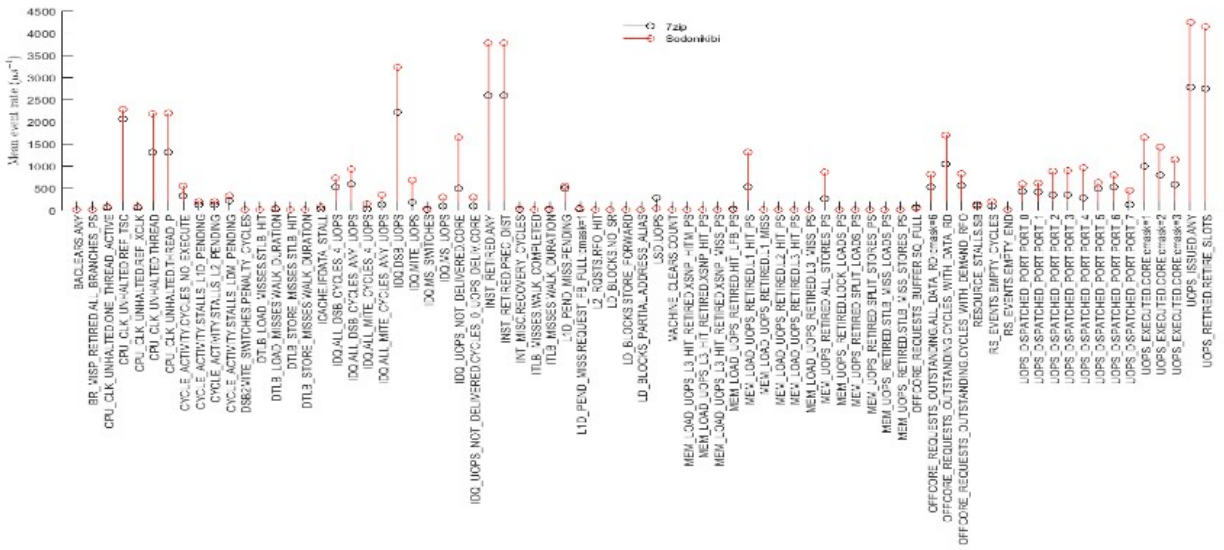


Figure 7.4: Mean Event Rate for a 7zip encryption trial vs. REvil/Sodinokibi ransomware, from analysis of VTune microarchitectural hardware performance counters.

wiped from the system, but DarkSide and Sodinokibi went undetected. It was also noted that VTune would not collect the full 150 second duration of microarchitecture data for DarkSide and WannaCry ransomware when executing from the command line. Additionally, while executing 7z and ransomware trials allowed for synchronization of counter data, there was no way to differentiate VTune start-up up performance metrics from 7z or ransomware functionality. Both issues required resolution before proceeding with future collection.

Aside from ensuring proper security protocols were in place to prevent unintentional executions and propagation of ransomware, the main detractor from using live ransomware on system hardware was the 1.5-2 hour restore process required in between trials.

7.2 Proof-of-Concept B: Simultaneous Current and Vtune Data Collection of ransomware in Non-virtualized and Virtualized Environments

The intent of this study was to compare Vtune results for several real-world ransomware samples obtained from hardware to those obtained from a virtual machine, with the goal of significantly reducing system restore time in between trials. This required duplicate system set ups in order to control as many variables as possible between the non-virtualized and virtualized environments.

This section describes the effort to collect performance counter data from the Vtune profiling application on-system while simultaneously collecting power/current data from an off-system setup, as well the attempt to replicate the hardware performance counter collection process in a virtual machine, and is summarized in Figure 7.5.

# Trials (per operation)	Collection Method	Restore Process	Operations Tested (5 total)		
			Benign	Ransomware	
5	MANUAL	Redo Rescue	os	DarkSide	Spring '21
			7zip	REvil/Sodinokibi	Spring '21
Trial Parameters				WannaCry	Spring '21
OS idle (current only)		30 s			
Vtune idle (add Vtune)		30 s			
Test Operation Duration		120 s			
Data Recorder Parameters			Vtune Parameters		
Sampling Rate	200	kS/s	HPCs	108	UA exploration
Duration	180	s	Sampling Interval	0.05	ms
Span	100	A	Duration	150	s
Offset	0	A	Report "bins"	35,000	
			Effective Sampling Rate	~ 0.0043	s

Figure 7.5: Data Collection Summary for Proof of Concept B

7.2.1 Non-Virtualized Hardware Setup

The hardware set up consists of a test computer enabled to execute ransomware with the Vtune Profiler performance analysis tool installed to enable hardware performance counter collection. A current probe with amplifier and data recorder is utilized to collect side-channel power data. Figure 7.6 diagrams the test setup.

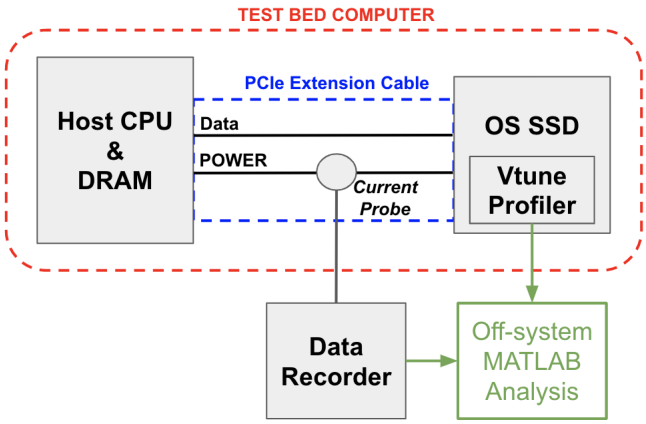


Figure 7.6: Experimental setup for monitoring power supplied to the test OS SSD. Current is measured with a current probe, amplified, and acquired and stored with the data recorder. Vtune microarchitectural hardware performance counters are collected from Vtune Profiler running on-system and then transferred off-system for additional analysis.

7.2.1.1 Testbed Computer

The test bed computer was a Dell Precision 7920 Tower and Intel Xeon Silver 4208 8-core 2.10 GHz System and Cascade Lake micro-architecture with 32 GB DDR4 memory running Windows 10 Pro Build 10.0.19044.1586 Version 21H2 Experience Pack 120.2212.4170.0 on a Samsung 970 EVO Plus M.2 SSD connected via PCIe extension cable over UEFI.

Windows 10 Pro Version 21H2 includes Microsoft Defender, an update to Windows Defender in Version 21H1, which presented some issues when executing mid-2021 versions of ransomware.

Multiple attempts were made to permanently disable Microsoft Defender (including disabling Tamper Protection and Window Security Center), with no success. Even with Microsoft Defender temporarily disabled, the system still detected WannaCry and prevented its execution. The system also detected ransomware when executing Darkside, and while DarkSide was permitted to execute, it did encrypt any important data due to built in ransomware Protection that prevents access of controlled folders. The system did not detect Sodinokibi. Since the goal of this study was to determine what differences exist between non-virtualized and virtualized HPCs, the Windows Pro Version 21H1 (also used in Section 7.1.1.1) was reinstalled on the updated system hardware.

The resulting test bed computer was an Intel Core i5-4570 System with 8 GB DDR3 running Windows 10 Pro Version 10.0.19043 and an Intel Xeon Silver 4208 Processor with a Cascade Lake micro-architecture. Windows Defender was disabled by setting the "DisableAntiSpyware" flag to 1 through the Registry Editor to enable execution of ransomware.

7.2.1.2 Ransomware

Live ransomware samples of WannaCry, DarkSide, and REvil/Sodinokibi obtained from VirusTotal in May 2021 were manually transferred from a zipped, password protected external hard drive, which had to be removed prior to initiating any trials to prevent corruption of the external drive by the ransomware.

7.2.1.3 Vtune HPC Measurements

Vtune Profiler performance analysis tool is run from the command line interface to collect micro-architecture exploration analysis type data using the same flags and values from Section 7.1.1.3.

7.2.1.4 Current Measurements

To collect power side-channel measurements during idle, encryption, and ransomware execution, a Tektronix TCP312 current probe (Tektronix, Inc. Beaverton, Oregon, USA) with the Tektronix TCPA300 current probe amplifier was placed around all five 12 V PCIe lines separated from a PCIe extension cable. All current data were collected by a GEN3i High-Speed Data recorder with a GN412 High-Speed 100 MSps Data Acquisition Card (HBM, Inc. Darmstadt, Germany) at a rate of 200 kSps with 14 bits per sample.

7.2.2 Virtualized System Set-up

The intent of the 2nd part of the study was to compare virtualized HPC results to the HPCs obtained on hardware in part 1 in order to determine the feasibility of using a VM in future experiments to vastly reduce the amount of time required to restore the solid state drive after compromising it with ransomware.

7.2.2.1 Testbed Computer

The testbed computer hardware was identical to that in Section 7.2.1.1. The following steps were performed to create an equivalent virtualized testing network:

1. Set-up equivalent system in VMware ESXi

2. Created Ransomware network

3. Created Ubuntu VM to scp Vtune .csv files after each run

4. Created Windows VM from resulting hardware image described in Section 7.2.1.1

5. Configured network for Windows VM to txr files to Ransomware network

6. Configured VM to Virtualize CPU Performance Counters

7. Set Windows VM to non-persistent state

7.2.2.2 Vtune Collection on Virtual Machines

Since the performance counters collected through Vtune Profiler are Microarchitecture-based, they use Hardware Event-based Sampling, as described in Section 4.2.2. Even with VMware settings configured to virtualize performance counters selected, Vtune cannot collect hardware events. The following error was received: `Hardware Event-based Sampling mode used in this analysis is not supported on Windows OS inside a VMware virtual machine. For full functionality, consider using a bare-metal environment.`

While the VM is still useful for learning some generalized ransomware behavior, it is not a viable path for collecting Hardware Event-based HPC data using Precise-Event Based Sampling, which is desired to ensure accuracy of hardware event counts with as many performance counter events as possible.

7.2.3 Data Collection Procedure

Five trials each of OS background activity ("idle"), 7zip encryption, and 3 real-world ransomware samples (DarkSide, Sodinokibi, and WannaCry) were collected in randomized order. The data recorder was manually started to collect 30 seconds of baseline OS current measurements before manually starting Vtune Profiler to collect 150 seconds of HPC for the entire system at the 30 second mark. The trial operation (OS only idle, 7 zip, or selected ransomware) was manually executed 30 seconds after Vtune HPC collection began (60 seconds from start of trial overall), and all events were terminated after 2 minutes of trial data collection. The safe file transfer and system restore procedure from Section 7.1.3 followed all ransomware trials. The data collection procedure is depicted in Figure 7.7.

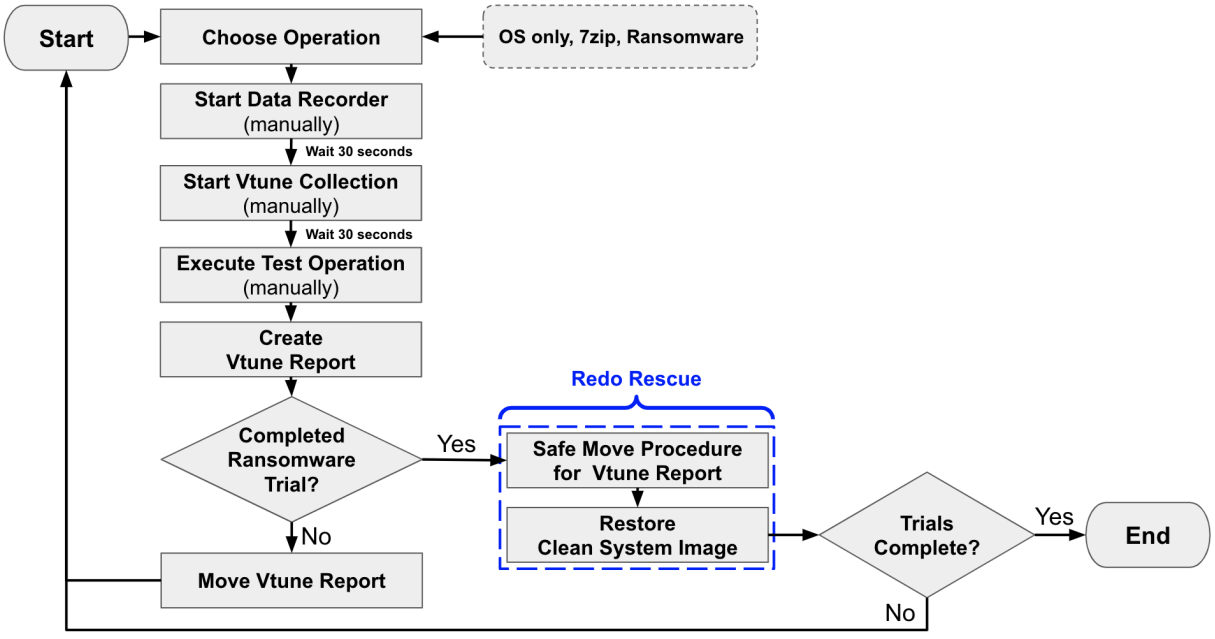


Figure 7.7: Data Collection and System Restore Procedure for manually executed trials.

7.2.4 Vtune Performance Counter Analysis

Appendix A lists HPCs collected across all trials, along with their descriptions. The numbers in the second column pertain to HPCs collected in this experiment. Of these, 95 were present for trials in this section (the remaining 13 were discovered in follow-on experiments), and 69 of these counters appeared in all 25 trial runs. Figure 7.8 lists the 12 HPCs that appeared in all operations other than idle, while Figure 7.9 lists the 7 performance counters that were collected only in the ransomware trials.

Name	Description
EXE_ACTIVITY.BOUND_ON_STORES	Cycles where the Store Buffer was full and no outstanding load
CYCLE_ACTIVITY.STALLS_L3_MISS	Execution stalls while L3 cache miss demand load is outstanding
DTLB_STORE_MISSES.STLB_HIT	Stores that miss the DTLB and hit the STLB
L1D_PEND_MISS.FB_FULL:cmask=1	Number of times a request needed a FL (Fill Buffer) entry but there was no entry available for it. Request includes cacheable/uncacheable demands that are load, store, or SW prefetch instructions
L2_RQSTS.RFO_HIT	Counts the RFO (Read For Ownership) requests that hit L2 cache
LD_BLOCKS.STORE_FORWARD	Counts the number of times where store forwarding was prevented for a load operation. Most common case: Load blocked due to address of memory access (partially) overlapping with a preceding uncompleted store
MEM_INST_RETIRED.SPLIT_LOADS_PS	Retired load instruction that split across a cache line boundary
MEM_INST_RETIRED.SPLIT_STORES_PS	Retires store instructions that split across a cache line boundary
MEM_INST_RETIRED.STLB_MISS_LOADS_PS	Number of retired load instructions that (start a) miss in the STLB
MEM_LOAD_L3_MISS_RETIRED.LOCAL_DRAM_PS	Retired load instruction which data sources missed L3 but serviced from local DRAM
MEM_LOAD_RETIRED.L3_MISS_PS	Retired load instructions missed L3 cache as data sources
OFFCORE_REQUESTS_OUTSTANDING.ALL_DATA_RD:cmask=4	Counts the number of offcore outstanding cacheable Core Data Read transactions in the super queue every cycle. Transaction is considered to be in the Offcore outstanding state between L2 miss and transaction completion sent to requestor (SQ deallocation)

Figure 7.8: Vtune Hardware Performance Counters collected solely in 20 of 20 non-idle trials.

7.2.5 Proof-of-Concept B Observations

This study was successful in developing a process to simultaneously collect off-system (power/current) and on-system (HPC) data, but further analysis of collected data required development of a method to synchronize collection of off-system current and internal micro-architecture measurements. Additionally, the data collection effort required expansion to include other “real

Name	Description
DTLB_LOAD_MISSES.STLB_HIT	Counts loads that miss the DTLB (Data Translation Lookaside Buffer) and hit the STLB (Second-level TLB)
OCR.DEMAND_DATA_RD.L3_HIT.HITM_OTHER_CORE	Counts demand data reads
OCR.DEMAND_RFO.L3_HIT.HITM_OTHER_CORE	Counts all demand data writes (RFOs)
OCR.PF_L2_RFO.L3_HIT.HITM_OTHER_CORE	Counts all prefetch (that bring data to L2) RFOs
ARITH.DIVIDER_ACTIVE	Cycles when divide unit is busy executing divide or square root operations. Accounts for integer and floating-point operations
ROB_MISC_EVENTS.PAUSE_INST	Number of retired PAUSE instructions (that did not end up with a VMExit to the VMM; TSX aborted insts may be counted) - event not supported on 1st SKL and KBL products
UOPS_RETIRED.MACRO_FUSED	Counts the number of macro-fused ups retired

Figure 7.9: Vtune Hardware Performance Counters collected solely in 15 of 15 non-idle trials.

world" workloads that might provide additional insights as to whether the mere presence of the HPCs in Figure 7.9 might be a reliable indicator of ransomware operations. Finally, restoring the drive in between trials took a significant amount of time compared to the time to run the trial itself and was a major obstacle to the ability to collect sufficient data for statistical analysis. Due to the lack of success using Vtune Hardware Event-based Sampling in a virtualized environment, the drive restore process was further examined to reduce restore time between trials, and the improved safe transfer and restore procedure is outlined in Section 7.2.6.

7.2.6 Drive Restore Procedure Using Image for Linux

To increase the ease of safe transfer and the speed of system restore, Image for Linux (IFL) [208] running on a USB3.0 thumb drive replaced Redo Rescue. IFL includes the ability to exit to a command prompt and safely transfer the Vtune `.csv` output files in read-only mode. The restore image was copied to an M.2 NVMe SSD that could be easily removed from the system before trials to prevent ransomware contamination and replaced only during the restore process. These changes reduced average drive restore time from 60 minutes to 10 minutes, a more tolerable period for mass collection.

7.3 Study 1: Automate, Expedite, and Expand Vtune collection

This study included 23 different operations and was designed to incorporate the required improvements identified in Section 7.2.5. It consisted of automating the full scope of each individual trial run through a python script, introducing other operations more representative of a "real world" system workload, and incorporating the streamlined restore process using the results obtained in Section 7.2.6. Study 1 is summarized in Figure 7.10 and is described in the follow sections.

# Trials (per operation)	Collection Method	Restore Process	Operations Tested (23 total)		
			Benign	Ransomware	
20	SCRIPT	Image for Linux	os	DarkSide	Spring '21
			7zip	REvil/Sodinokibi	Spring '21
			18 SPEC Benchmarks	WannaCry	Spring '21

Trial Parameters		
OS idle (current only)	30	s
Vtune idle (add Vtune)	30	s
Test Operation Duration	60	s

Data Recorder Parameters		
Sampling Rate	200	kS/s
Duration	120	s
Span	4	A
Offset	-1.5	A

Vtune Parameters		
HPCs	108	UA exploration
Sampling Interval	1	ms
Duration	90	s
Report "bins"	18,000	
Effective Sampling Rate	0.005	s

Figure 7.10: Data Collection Summary for Study 1

7.3.1 Data Collection Procedure

The following sections describe the improvements made to synchronize off-system current measurements with internal microarchitecture measurements and considerations in simulating a realistic workload against which to compare ransomware HPCs.

7.3.1.1 Hardware Setup

This section uses the same hardware, ransomware, and data (current and HPC) collection setup described in Section 7.2.1 and the restore procedure in Section 7.2.6. The saved system restore image was updated to include ransomware executables saved on the system desktop in a zipped folder to facilitate data collection automation, and the necessary software to simulate a realistic workload, described in the next section.

7.3.1.2 Simulated "Real-World" Workload

To simulate a realistic "real world" workload, SPECworkstation 3.1[2] benchmarks were examined. SPECworkstation 3.1 was released in March 2021 to measure key CPU, GPU, memory, and I/O aspects of workstation performance, and includes workloads for media and entertainment, product development, life sciences, financial services, energy, general operations, and GPU compute capability. It consists of the benchmarks shown in Figure 7.11.

All CPU and GPU workloads were tested for current draw and disk activity. The CPU workloads were generally found to have low disk activity (and as a result low current draw), with the exceptions of 7zip, Octave, and the first 5-10 seconds of Blender, python36, and rodiniaCFD. The GPU workloads generally had higher disk activity, but were discounted since the scope of this research is to consider performance counters integrated within the CPU. The two CPU benchmarks with the behavior most comparable to ransomware were found to be the 7zip and Octave benchmarks. These were added to the randomized test plan for current and HPC side-channel collection first, followed by all available SPECworkstation 3.1 CPU benchmarks.

[2]https://gwpg.spec.org/benchmarks/benchmark/specworkstation-3$_1$/

Test ID	Benchmark	Description
s01	7zip	freeware program that offers good scaling performance between processors to compress and extract a hierarchy of files
s02	octave	computes a variety of math operations using Octave programming language for scientific computing
s03	Blender	measures performance of content-creation workflows
s04	CalculiX	models internal temp of jet engine turbine
s05	Convolution	computes a convolution on a 20k x 20k pixel image with a 100x100 random filter
s06	FFTW	computes discrete Fourier 1D, 2D, & 3D transforms
s07	fsi	models the fluid-structure interaction problem, which examines characteristics of fluid flow around an object
s08	handbrake	based on the open-source media encoder, transcodes 4K mp4 file with an H.265 encoder at 2 resolutions
s09	Kirchhoff	uses the Kirchhoff equation form of the wave equation to calculate the back propagation of the seismic wavefield
s10	lammps	runs the Sandia National Laboratories molecular dynamics simulator, which consists of 5 tests to simulate molecular properties
s11	luxrender	renders a chrome sphere resting on a grid of numbers in a beach scene using LuxMark benchmark
s12	namd	a scalable molecular dynamics simulator which consists of 3 tests that simulate molecular interactions
s13	WPCcfd	uses combustion solver XiFoam to solve compressible premixed/partially-premixed combustion with turbulence modeling and is based on OpenFOAM, a computational fluid dynamics solver
s14	poisson	solves the Poisson equation using OpenMP for parallel execution
s15	python36	uses Python 3.6 to benchmark a variety of math operations
s16	rodiniaLifeSci	a benchmark suite for heterogeneous computing platforms, consists of a collection of parallel programs written in several parallel languages
s17	rodiniaCFD	computational fluid dynamics solver for 3D Euler equations representing compressible flow
s18	srmp	implements Surface-Related Multiples Prediciton (SRMP) algorithm for seismic data processing (to remove multiples from seismic data)

Figure 7.11: SPECworkstation 3.1 Benchmarks incorporated into data collection.

7.3.1.3 Data Collection Automation/Synchronization

To synchronize collection of off-system current and internal micro-architectural measurements, a python script was developed to automate the initiation of current data recording by sending a voltage pulse through a serial USB port to the data recorder, which was configured to run for exactly 180 seconds. After 30 seconds, the script executes Vtune Profiler to collect micro-architectural data on the full system for 150 seconds. The trial operation indicated by the user is initiated after another 30 seconds. All data collection stops automatically at the 3 minute mark, at which point the user terminates any active processes and generates the Vtune report to move off system. The full trial collection procedure is depicted in Figure 7.12.

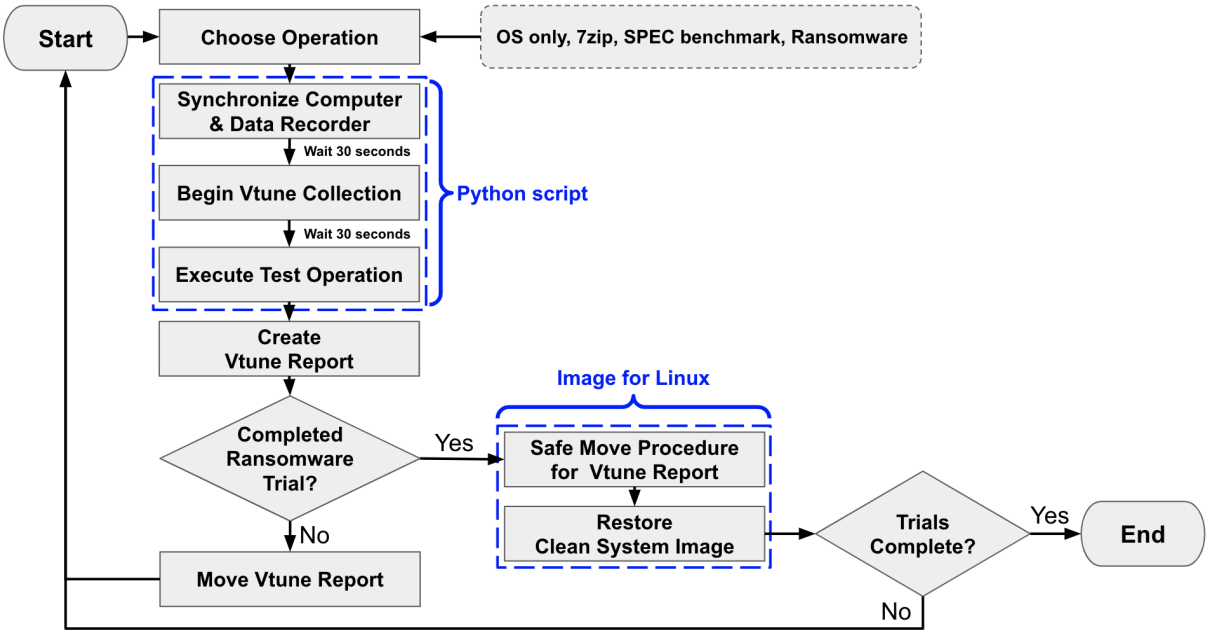


Figure 7.12: Data Collection and System Restore Procedure Study 1.

7.3.2 Experiment Results

The 23-class, 460 trial data set of standard Vtune microarchitectural performance counters was processed alongside the data set obtained in Study 2 (Section 7.4) to determine which performance counters and classification methods were most successful in identifying ransomware operations. The analysis method is described in Section 7.5, and ultimately eliminated the standard Vtune microarchitectural performance counters collection method.

7.4 Study 2: Expanded Vtune HPC Collection with Additional Ransomware Samples

This study doubled the number of HPCs collected and analyzed and added 8 recent ransomware samples to more closely balance the dataset. It is summarized in Figure 7.13 and described in the sections below.

# Trials (per operation)	Collection Method	Restore Process
10	SCRIPT	IFL

Trial Parameters		
OS idle (current only)	0	s
Vtune idle (add Vtune)	10	s
Test Operation Duration	30	s

Data Recorder Parameters		
Sampling Rate	kS/s	
Duration	s	
Span	A	
Offset	A	

Operations Tested (31 total)		
Benign	**Ransomware**	
os	DarkSide_Spr21	Sodinokibi_0722
7zip	Gibberish_0422	Snatch_0422
18 SPEC Benchmarks	HiddenTear_0422	Sodinokibi_0222
	Phobos_0122	WannaCry_Spr21
	Phobos_0522	WannaCry_0622
	REvilSodinokibi_Spr21	

Vtune Parameters		
HPCs	220	custom
Sampling Interval	0.01	ms
Duration	40	s
Report "bins"	8,000	
Effective Sampling Rate	0.005	s

Figure 7.13: Data Collection Summary for Study 2.

7.4.0.1 Data Collection Procedure

The 31-class collection consisted of only the Vtune HPC collection of 220 custom-selected performance counters. The experiment was similar to the 23-class version, but eliminated the current collection, added 8 additional Ransomware samples, and collected 10 seconds of Vtune HPC measurements before the test operation was triggered and collected for 30 seconds. Vtune collected HPCs in sampling mode at the highest possible Vtune sampling rate, which uses the minimum possible sampling interval of 0.01 ms and obtained a multiplexing accuracy of over 99%. All data collection stops automatically at the 40 second mark, at which point the user terminates any active processes. Generation of the Vtune 40 second Vtune database for the trial took significantly longer than for the 23-class collection due to the high sampling rate, a possible indicator that the low sampling interval would not be a viable path forward for on-system detection due to increased system overhead. The entire resulting Vtune database file was saved and moved off system using Image for Linux during the safe-restore procedure. Once off-system, the database was available for Vtune report generation and could be divided into any number of desired bins. To maintain consistency with the 23-class experiment, the 40 seconds of Vtune data were divided into 8k bins to obtain an effective sampling rate of 5 ms. The data collection process is shown in Figure 7.14.

7.4.1 Experiment Results

The 31-class, 310 trial data set of 220 custom selection Vtune performance counters was processed alongside the data set obtained in Study 1 (Section 7.3) to determine which performance counters and classification methods were most successful in identifying ransomware operations.

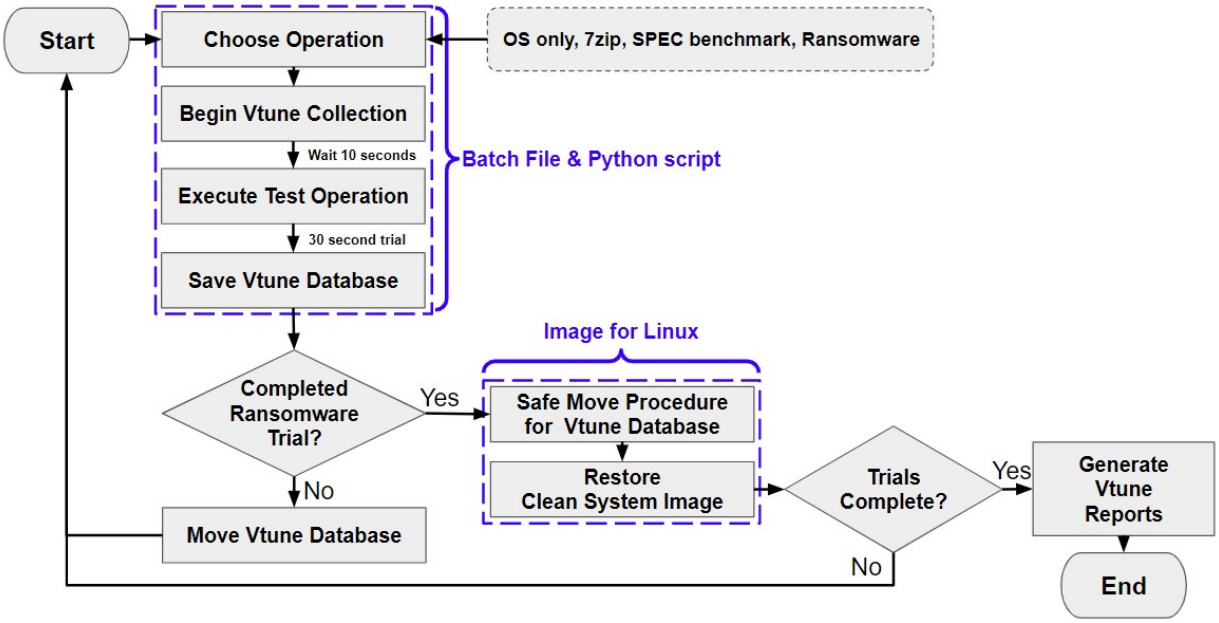


Figure 7.14: Data Collection, System Restore, and Report Generation Procedure for 31-class data set (Study 2).

The analysis method is described in Section 7.5, in which the results ultimately identified a reduced set of 90 features for additional collection and analysis. Additionally, it was noted that the significant processing time required to transfer the entire Vtune database and generate custom time reports generated a need for an more user-friendly data transfer and parallel report generation capability was identified and incorporated in the follow-on study described in Section 8.1.

7.5 Analysis of Performance Counters for Classification of Ransomware Operations for Studies 1 (23-class) and 2 (31-class)

This section describes the method by which the 23-class (Section 7.3) and 31-class (Section 7.4) results were analyzed to identify algorithms with good classification potential so that the best performance counters for ransomware classification/detection could be further refined. MATLAB 2022a Classification Learning App was used for initial identification of classification algorithms

that performed well with the data sets, and then HPCs were tested in the selected algorithms one at a time. Classification statistics (F1) were used to determine which HPCs warranted further analysis.

7.5.1 Feature Generation

The Vtune reports generated for all randomly ordered trials in both studies listed hardware event count over time for each HPC collected. These reports were sorted and then processed in MATLAB to generate features for 3 windows of data at operation start time (30s for the 23-class Vtune UA collection and 10s for the 31-class custom-220-HPC Vtune collection) as well as the window before and after for 4 different window sizes (0.1, 0.5, 1, and 2 seconds). This yielded 12 comparable sets of features for the two experiments, with the start times, window sizes, and stop times indicated in Figures 7.15 and 7.16, respectively.

23-classes (3 Ransomwares, OS, 7zip, 18 SPEC benchmarks)

Start Time	29.9	30	30.1	29.5	30	30.5	29	30	31	28	30	32
Bin Size	0.1	0.1	0.1	0.5	0.5	0.5	1	1	1	2	2	2
Stop Time	30	30.1	30.2	30	30.5	31	30	31	32	30	32	34

Figure 7.15: Start Time - Window Size - Stop Time Combinations for Study 1. Each column indicates a time interval for which HPC feature sets were generated (12 total).

The resulting 24 feature sets (12 for each Study) allowed for exploration of HPC importance for distinct windows of time.

31-classes (11 Ransomwares, OS, 7zip, 18 SPEC benchmarks)

Start Time	9.9	10	10.1	9.5	10	10.5	9	10	11	8	10	12
Bin Size	0.1	0.1	0.1	0.5	0.5	0.5	1	1	1	2	2	2
Stop Time	10	10.1	10.2	10	10.5	11	10	11	12	10	12	14

Figure 7.16: Start Time - Window Size - Stop Time Combinations for Study 2. Each column indicates a time interval for which HPC feature sets were generated (12 total).

7.5.2 Classification Algorithm Identification

This section describes the classification models available in the MATLAB Classification Learner Application as well as the process by which training data was evaluated and a subset of models selected.

7.5.2.1 MATLAB Classification Learner Application

MATLAB's Classification Learner Application provides a graphic user interface though which the performance of different classification models can be explored and algorithm parameters investigated. The Classification Learner consists of nine distinct classifier types, most of which include multiple classification models. The nine classifier types explored through the Classification Learner are:

1. **Decision Trees.** Decision trees are binary trees that predict responses to data by checking the value of one predictor at a time. They are further characterized by the number of splits used to distinguish between classes.

2. **Discriminant Analysis.** Discriminant Analysis models assume each class can be described

by a Gaussian distributions, and the classifier is trained by estimating the parameters of each distribution. They are further characterized by whether the function used to determine class boundaries is linear or non-linear.

3. **Logistic Regression.** Logistic Regression Classifiers model class probabilities as a function of the linear combination of predictors and can be used for binary classification problems only.

4. **Naive Bayes.** Naive Bayes classifiers are based on Bayes' Theorem (conditional probability of an event occurring given that another event occurred) and assumes predictors are conditionally independent. Models are further characterized by the function used to estimate the distribution of data and are useful for multi-class classification problems.

5. **Support Vector Machines.** Support Vector Machine (SVM) models classify data by finding the "best" (largest margin) "hyperplane" (boundary) that separates data points of different classes. The data points closest to the separating hyperplane are called "Support Vectors." Models types are further distinguished based on the Kernel function (Linear, Quadratic, etc.) used to transform input data into the appropriate form for processing. SVM classifiers can be used with 2 or more classes.

6. **Nearest Neighbor.** Nearest neighbor classifiers categorize query points based on the distance to "neighbor" points in the training data. Models are further characterized by the number of neighbors or the metric used to determine distance.

7. **Ensemble.** Ensemble models combine results from weaker learners into a higher quality model. They combine "learner" programs to build models from input training data. The

result is a "Classifier" program that is able to make output predictions based on input test data. Ensembles are further characterized and named by the types of learners used and the resulting classifiers.

8. **Neural Network.** Neural network classification algorithms are designed to recognize patterns, based on the how the human brain functions. All models consist of a first layer connected to the input data, 1 (or more) fully connected layer, and an output layer. Each fully connect layer consists of "neurons" (nodes where computation occurs), which multiplies input data by a weight matrix, adds a bias vector, and is followed by an activation function. The default activation function in the Classification Learner is the Rectified Linear Unit (ReLU). Neural networks are further characterized by the size and number of layers.

9. **Kernel Approximation.** Kernel approximation models perform non-linear classification of data with many observations by mapping predictions from low-dimensional space to high-dimensional space, and then applying a linear classification model in the high-dimensional space. Kernel approximation classifiers are further characterized by the linear model used to fit data in the high-dimensional space.

Figure 7.17, which was adapted from [4], summarizes each model type, description, and distinguishing characteristics. It includes accuracy of model predictions, the time it takes to train a classifier with a given set of input data, and whether or not the classifier uses memory efficiently. It also captures ease of model interpret-ability (whether the user can determine the reasoning behind the classifier predictions), model flexibility (ability to adjust the model parameters), and types of predictors (input variable data types) supported.

CLASSIFIER	DESCRIPTION		ACCURACY	TRAINING SPEED	MEMORY USAGE	INTERPRET-ABILITY	FLEXIBILITY	PREDICTORS SUPPORTED		
								All Numeric	All Categorical	Combination
Decision Trees		**MAX SPLITS**								
COARSE	Binary trees that predict responses to data by checking the value of one predictor at a time	4	can be low; simple trees can prevent overfitting	Fast	Efficient	Easy	Low	☑	☑	☑
MEDIUM		20					Medium			
FINE		100					High			
Discriminant Analysis		**Class Boundaries**								
LINEAR	Models assume each class can be described by a gaussian distribution; good for wide datasets.	linear	Good	Fast	Efficient	Easy	Low	☑	☐	☐
QUADRATIC		non-linear (ellipse, parabola, hyperbola)					Low			
Logistic Regression										
Models class probabilities as a function of the linear combination of predictors; for 2 classes.			can overfit in large datasets	Fast	Efficient	Easy	Low	☑	☑	☑
Naive Bayes										
GAUSSIAN	Based on Bayes' Conditional Probability Theorem, assumes predictors are conditionally independent; useful for multi-class classification.		Good if conditional independence assumption holds	Fast	Efficient	Easy	Low	☑	☑	☑
KERNEL							Medium			
Support Vector Machines										
LINEAR	Models classify data by finding the "best" (largest margin) "hyperplane" (boundary) that separates data points of different classes. The data points closest to the separating hyperplane are called "Support Vectors." Models types are named based on the Kernel function (Linear, Quadratic, etc.) used to transform input data into the appropriate form for processing; can be used with 2 or more classes.		Good	Slow for large datasets	Generally Efficient; but High for large datasets	Easy	Low	☑	☑	☑
QUADRATIC						Hard	Medium			
CUBIC							Medium			
FINE GAUSSIAN							High			
MEDIUM GAUSSIAN							Medium			
COARSE GAUSSIAN							Low			
Nearest Neighbor		**# of Neighbors**								
FINE	Models categorize query points based on the distance to "neighbor" points in the training data. Models are named based on number of neighbors or the metric used to determine distance.	1	Good in low dimen-sions	Fast	High	Hard	High	Euclidian Distance only	Hamming Distance only	☐
MEDIUM		10					Medium			
COARSE		100					Low			
COSINE		10, cosine distance					Medium			
CUBIC		10, cubic distance					Medium			
WEIGHTED		10, distance weight					Medium			
Ensemble		**Classifier / Learner**								
BOOSTED TREES	Ensemble models combine results from weaker learners into a higher quality model. They combine "learner" programs to build models from input training data. The result is a "Classifier" program that is able to make output predictions based on input test data.	AdaBoost / Decision Tree	Good	Slow	High	Hard	Medium-to-high	☐	☑	☑
BAGGED TREES		Random Forest Bag / Decision Tree					High			
SUBSPACE DISCRIMINANT		Subspace / Discriminant					Medium	☑	☐	☐
SUBSPACE KNN		Subspace / Nearest Neighbor					Medium		☑	
RUSBoost TREES		RUSBoost / Decision Tree					Medium			☑
Neural Network		**LAYER SIZES**								
NARROW	Algorithms designed to recognize patterns, based on the human brain. All models consist of a 1st layer connected to input data, 1 (or more) fully connected layers, and an output layer.	10	Good	Slow	High	Hard	increases with size & number of fully connected layers	☑	☑	☑
MEDIUM		25								
WIDE		100								
BILAYERED		[10 10]								
TRILAYERED		[10 10 10]								
Kernel Approximation										
SVM	Performs non-linear classification of data with many observations by mapping predictions from low-dimensional space to high-dimensional space. Named after the linear model used to fit data in the high-dimensional space.		*see **Support Vector Machines***			Hard	Medium	☑	☑	☑
LOGISTIC REGRESSION			*see **Logistic Regression***				Medium			

Figure 7.17: MATLAB Classification Models Evaluated in Classification Learning Application, adapted from [4].

7.5.2.2 Classification Model Selection

To identify which Classification Learner Application models might be most appropriate for a data set of hardware performance counters, each set of features for both studies of multi- (23- or 31-) class and binary (ransomware or benign) versions of labels were loaded individually into the MATLAB Classification Learner Application, where all 32 available classification models were trained using 10-fold cross validation (CV).

The cross validation accuracy for each feature set and every model for Study 1 (23 classes) is shown in Figure 7.18, with the multi-class results on the top half of the table and binary results in the bottom half. Figure 7.19 shows cross validation accuracy for all Study 2 feature sets and classifier combinations. In both Figures, the highest CV Accuracy for each set (row) of features highlighted yellow, while CV Accuracy greater than or equal to 90% is shown in orange font and CV Accuracy greater than or equal to 95% is shown in red font.

CLASSIFICATION ACCURACY				ALL MODELS																														
				1	2	3	4	5		6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
				Tree			Discriminant			Naive Bayes		SVM						KNN						Ensemble					Neural Network					Kernel	
Start Time	Bin Size	Stop Time	# Classes	Fine	Medium	Coarse	Linear	Quadratic	Logistic Regression	Gaussian	Kernel	Linear	Quadratic	Cubic	Fine Gaussian	Medium Gaussian	Coarse Gaussian	Fine	Medium	Coarse	Cosine	Cubic	Weighted	Boosted Trees	Bagged Trees	Subspace Discriminant	Subspace KNN	RUS- Boosted Trees	Narrow	Medium	Wide	Bilayered	Trilayered	SVM	Logistic Regression
29.9	0.1	30	23	57.2	50.7	21.5	f	f	-	f	49.8	60.4	58.7	58.3	23.3	58.7	54.6	56.5	51.1	26.1	53.7	51.3	55.0	60.2	65.7	59.3	58.7	50.7	51.5	56.1	58.0	50.4	52.0	54.8	54.8
30	0.1	30.1	23	60.2	56.1	21.3	f	f	-	f	51.5	58.7	58.3	57.8	16.7	56.5	51.5	56.5	51.3	26.3	51.5	49.3	53.5	60.9	67.8	58.0	57.6	56.1	53.7	56.3	55.4	52.8	48.0	53.5	55.4
30.1	0.1	30.2	23	59.3	55.7	21.7	f	f	-	f	48.9	57.0	59.3	55.4	20.4	56.1	49.8	52.8	51.7	26.1	50.0	49.1	50.2	62.2	64.1	58.9	61.3	54.8	48.9	51.1	52.4	47.8	48.0	59.1	54.8
29.5	0.5	30	23	70.9	64.8	21.5	f	f	-	f	60.4	76.3	74.3	74.1	40.0	75.0	69.8	68.7	69.3	30.9	66.3	66.7	71.1	75.2	76.3	75.0	68.9	64.8	68.0	69.3	69.6	63.9	64.6	65.7	64.3
30	0.5	30.5	23	64.6	63.0	21.5	f	f	-	f	59.1	71.3	72.8	72.8	40.7	72.6	65.0	68.3	67.8	36.7	65.7	63.9	68.9	68.7	73.3	73.3	70.2	62.8	65.9	67.8	68.7	64.8	65.7	70.2	66.5
30.5	0.5	31	23	65.0	61.7	21.3	f	f	-	f	61.1	69.8	68.0	68.9	38.0	68.9	63.3	67.0	65.7	29.3	64.3	63.0	67.0	71.1	73.0	72.0	68.5	61.3	63.9	66.5	69.6	63.3	63.5	65.4	61.7
29	1	30	23	72.8	72.2	21.5	f	f	-	f	62.8	79.8	77.2	78.9	43.5	77.6	75.0	74.8	75.4	31.5	75.0	73.5	75.9	76.5	78.7	79.6	74.6	72.2	75.2	75.4	77.4	70.7	65.0	70.9	66.5
30	1	31	23	72.2	66.3	21.3	f	f	-	f	62.0	75.9	77.8	78.9	43.3	77.2	67.4	73.5	71.1	33.5	70.4	67.2	74.6	74.3	78.5	76.7	71.7	66.1	70.2	74.6	76.5	70.0	68.0	68.7	68.3
31	1	32	23	69.3	67.4	21.5	f	f	-	f	66.3	76.5	78.3	75.9	42.0	77.2	73.5	74.1	73.3	29.8	72.6	70.2	75.9	74.3	78.9	74.6	72.4	67.6	68.9	72.2	73.7	67.4	68.5	71.3	67.4
28	2	30	23	81.5	74.1	21.5	f	f	-	f	67.0	81.7	83.3	83.7	48.5	82.2	75.4	78.9	79.6	38.7	77.6	78.9	82.0	82.2	84.6	82.0	76.5	74.1	81.3	81.7	80.7	77.8	78.0	72.6	70.2
30	2	32	23	75.2	70.0	21.3	f	f	-	f	68.0	81.7	81.5	82.6	50.4	79.6	73.5	78.5	75.9	24.8	76.5	74.1	78.3	74.8	79.6	78.3	74.6	68.3	76.7	76.3	76.5	75.4	70.2	74.6	71.7
32	2	34	23	78.7	75.9	21.7	f	f	-	f	70.2	82.6	83.0	83.0	51.5	80.9	77.4	77.6	76.5	22.8	76.1	74.1	79.3	84.1	84.1	82.4	78.0	75.9	76.1	76.7	78.0	75.7	70.0	76.5	73.5
29.9	0.1	30	2	86.5	87.2	87.8	f	f	81.1	f	61.3	87.4	85.4	83.0	86.7	87.0	87.0	87.2	87.6	87.0	87.6	87.8	88.0	88.7	88.5	88.9	87.2	83.0	84.4	85.7	83.7	81.1	85.4	87.0	87.0
30	0.1	30.1	2	87.0	87.4	90.7	f	f	82.2	f	60.9	88.7	87.8	82.4	87.0	86.7	87.0	85.0	87.4	87.0	87.0	87.2	87.2	86.7	88.7	88.7	87.6	82.8	85.0	84.3	83.5	84.3	86.5	87.0	87.0
30.1	0.1	30.2	2	87.0	87.4	90.0	f	f	82.6	f	61.3	89.1	86.7	84.1	87.0	87.0	87.0	86.5	87.6	87.0	87.6	87.2	88.3	88.0	90.2	90.2	90.4	83.5	87.2	86.7	85.2	85.2	85.2	86.7	87.0
29.5	0.5	30	2	89.1	89.1	90.4	f	f	83.5	f	58.9	92.0	92.6	84.1	89.6	89.6	87.0	86.1	89.8	87.0	88.9	90.0	89.1	87.8	92.0	92.4	88.0	87.6	89.1	88.7	88.9	89.3	87.8	87.2	87.0
30	0.5	30.5	2	88.7	88.9	91.3	f	f	85.4	f	61.5	92.4	90.7	90.0	91.1	91.3	87.0	90.0	92.0	86.5	91.7	90.9	91.7	88.3	91.3	93.0	89.8	89.3	91.3	92.6	92.8	91.1	92.6	87.0	87.0
30.5	0.5	31	2	86.7	86.7	88.5	f	f	83.7	f	62.6	92.0	91.3	89.8	90.0	89.6	87.0	89.3	90.2	87.0	90.2	89.3	90.9	88.9	91.3	92.6	88.9	87.2	90.2	89.8	89.8	91.1	89.6	86.7	87.0
29	1	30	2	91.1	91.1	92.4	f	f	86.5	f	60.9	93.0	93.0	90.4	92.2	89.8	87.0	91.3	91.3	87.0	91.5	91.7	91.5	88.3	93.7	93.9	91.5	93.3	92.0	88.9	92.0	91.5	89.6	87.6	87.0
30	1	31	2	92.8	92.8	88.9	f	f	85.9	f	63.0	94.6	94.8	93.3	94.1	91.1	87.0	95.0	94.3	87.0	94.3	94.1	95.2	91.7	95.7	94.6	91.7	93.0	92.8	93.5	94.3	92.2	93.3	89.3	87.4
31	1	32	2	92.0	92.0	93.3	f	f	85.2	f	64.8	93.9	93.5	93.5	93.0	91.7	87.0	92.6	93.5	87.0	93.5	92.8	94.6	88.0	95.2	95.7	93.0	94.1	92.8	91.5	92.0	91.5	93.5	90.2	88.0
28	2	30	2	93.3	93.3	90.4	f	f	88.9	f	63.5	94.1	94.6	94.8	92.6	89.3	87.0	94.1	93.3	87.0	92.4	93.0	94.6	87.0	95.2	95.7	91.5	96.5	95.7	95.0	94.6	93.9	93.5	89.6	87.0
30	2	32	2	95.0	95.0	94.1	f	f	90.2	f	68.5	97.0	96.3	95.4	94.6	91.5	87.0	96.3	93.7	87.0	93.5	93.3	95.9	87.0	97.0	95.7	94.3	96.1	93.7	94.3	96.3	94.8	94.8	91.3	89.8
32	2	34	2	96.7	96.7	94.1	f	f	95.0	f	72.6	93.9	97.2	96.3	92.6	92.0	87.2	95.2	92.2	87.0	91.7	90.4	95.0	87..0	98.0	96.3	93.9	97.4	95.9	96.5	95.9	95.9	96.5	93.5	89.8

Figure 7.18: MATLAB Classification Learning Application Cross-Validation Accuracy for 23-class data set (Study 1)

CLASSIFICATION ACCURACY			ALL MODELS																														
			1	2	3	4	5		6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
			Tree			Discriminant			Naive Bayes		SVM						KNN						Ensemble					Neural Network					Kernel	
Start Time	Bin Size	Stop Time	# Classes	Fine	Medium	Coarse	Linear	Quadratic	Logistic Regression	Gaussian	Kernel	Linear	Quadratic	Cubic	Fine Gaussian	Medium Gaussian	Coarse Gaussian	Fine	Medium	Coarse	Cosine	Cubic	Weighted	Boosted Trees	Bagged Trees	Subspace Discriminant	Subspace KNN	RUS- Boosted Trees	Narrow	Medium	Wide	Bilayered	Trilayered	SVM	Logistic Regression
9.9	0.1	10	31	66.1	51.3	15.5	f	f	-	f	46.8	67.7	69.7	70.0	39.4	64.8	64.2	74.2	64.5	7.7	66.1	63.5	71.9	69.4	73.5	73.9	59.4	51.0	66.8	70.6	72.6	63.2	60.3	51.3	51.0
10	0.1	10.1	31	66.8	53.2	15.8	f	f	-	f	52.3	70.3	71.0	70.3	33.5	68.4	68.1	72.9	66.5	8.1	67.1	66.8	69.4	71.3	76.5	77.1	61.3	52.6	68.1	74.8	75.2	66.8	63.2	54.8	53.5
10.1	0.1	10.2	31	66.5	52.6	15.8	f	f	-	f	41.3	69.4	70.0	69.4	35.5	61.9	67.7	69.7	64.8	10.6	64.5	64.8	69.4	75.5	73.9	76.8	60.6	52.6	68.4	69.4	71.3	60.6	61.9	53.9	51.9
9.5	0.5	10	31	78.7	62.6	15.8	f	f	-	f	69.0	79.0	78.7	76.5	55.2	76.8	74.8	80.3	77.4	9.4	78.1	75.5	79.0	82.6	82.9	86.5	72.9	62.6	77.4	79.7	80.3	72.9	72.6	68.4	64.2
10	0.5	10.5	31	73.2	58.1	15.8	79.4	f	-	f	66.8	77.1	77.4	79.0	45.8	75.5	74.2	79.4	75.5	13.9	75.2	75.5	81.0	77.4	81.9	86.8	63.5	58.1	78.4	78.1	81.0	73.9	69.4	59.7	59.0
10.5	0.5	11	31	78.4	58.1	15.5	f	f	-	f	68.1	73.2	76.8	76.1	56.5	78.7	72.6	80.0	72.6	6.8	76.5	73.5	78.1	81.3	82.6	85.8	65.8	58.1	77.1	79.0	80.3	74.8	71.6	62.6	59.4
9	1	10	31	80.0	61.6	14.8	f	f	-	f	67.4	82.3	87.4	87.1	62.9	79.7	77.1	80.3	79.7	11.6	81.6	78.1	82.3	84.2	88.4	87.1	77.1	61.6	81.9	81.9	82.3	79.7	72.9	74.5	73.9
10	1	11	31	80.0	61.3	14.8	87.7	f	-	f	70.6	83.9	83.2	81.6	59.7	80.6	76.1	81.3	81.3	10.3	81.0	80.6	84.2	82.9	85.8	91.3	76.1	61.3	83.9	82.6	80.3	78.1	76.1	71.0	69.4
11	1	12	31	81.6	65.2	15.8	f	f	-	f	69.0	82.6	83.9	83.2	66.5	82.9	73.2	85.5	79.4	8.1	77.1	77.4	84.8	85.5	87.1	91.9	78.1	65.2	81.0	85.8	86.1	81.0	78.4	78.1	76.1
8	2	10	31	81.0	60.6	15.8	88.1	f	-	f	69.0	85.5	86.5	87.1	62.9	82.3	81.9	83.5	83.9	10.0	85.2	85.2	86.5	84.8	87.4	91.3	83.9	60.6	83.5	85.2	82.9	78.1	75.2	79.7	77.7
10	2	12	31	85.5	64.5	15.8	91.6	f	-	f	72.3	84.2	86.8	84.2	69.7	82.3	78.1	87.4	84.8	10.0	85.8	85.5	88.1	87.1	90.0	92.6	81.3	64.5	88.1	89.0	87.1	83.9	80.0	82.3	81.0
12	2	14	31	83.9	63.5	16.1	85.5	f	-	f	69.7	88.1	87.7	87.7	67.7	81.9	80.0	86.1	84.5	12.6	84.5	82.6	88.1	84.2	90.0	91.9	83.5	63.5	85.2	86.8	86.8	80.3	78.7	85.2	84.2
9.9	0.1	10	2	88.4	88.4	80.6	f	f	60.6	f	73.9	90.0	87.4	87.1	83.9	88.4	72.6	91.0	86.5	69.7	87.4	82.6	89.4	64.5	90.0	91.9	82.3	89.7	90.0	91.3	89.7	92.3	90.6	70.6	69.7
10	0.1	10.1	2	88.4	88.4	85.2	f	f	61.0	f	74.8	86.1	91.3	87.7	87.7	83.9	73.5	91.0	86.5	68.4	85.8	84.8	88.4	64.5	91.9	90.6	84.5	91.9	91.3	89.4	91.6	88.7	89.4	74.2	73.5
10.1	0.1	10.2	2	88.7	88.7	83.5	f	f	67.1	f	76.1	91.3	92.6	91.9	88.4	80.0	75.5	90.0	88.7	71.9	90.0	87.4	89.7	64.5	93.2	92.9	85.2	92.6	91.3	92.3	91.6	89.7	91.0	76.1	74.2
9.5	0.5	10	2	91.9	91.9	88.4	f	f	58.1	f	74.2	92.6	89.4	90.3	89.4	91.6	73.5	92.6	92.3	70.3	91.9	92.3	92.6	64.5	94.5	91.3	82.9	94.2	93.2	92.9	93.5	93.2	91.9	75.5	72.3
10	0.5	10.5	2	91.6	91.6	90.6	86.8	f	74.8	f	79.4	96.5	95.5	95.5	94.5	91.0	77.4	97.7	91.6	70.6	92.9	91.6	95.5	64.5	94.2	97.4	85.2	91.0	97.4	97.1	96.5	97.4	96.8	81.9	79.4
10.5	0.5	11	2	93.2	93.2	93.5	87.1	f	83.5	f	78.1	96.8	96.8	96.8	97.1	95.2	79.7	97.1	96.5	76.1	97.4	96.8	96.5	64.5	96.1	98.4	91.9	95.2	99.0	98.7	99.0	97.7	99.4	86.5	82.6
9	1	10	2	89.4	89.4	84.8	f	f	65.2	f	78.4	93.5	91.6	90.3	91.0	91.9	73.5	94.5	91.6	69.0	92.6	92.6	93.2	64.5	93.5	91.0	92.6	93.9	92.9	92.9	92.3	94.5	93.9	87.4	77.1
10	1	11	2	92.6	92.6	90.6	88.4	f	87.4	f	79.7	98.4	97.7	98.4	97.4	94.8	80.0	99.0	97.7	73.2	98.1	98.1	97.7	64.5	97.1	98.7	93.5	97.7	100.0	98.7	99.4	98.7	99.4	91.9	83.5
11	1	12	2	95.8	95.8	93.5	f	f	92.3	f	77.1	98.1	99.4	99.4	97.1	96.1	79.0	99.4	97.4	82.6	96.8	95.8	98.7	64.5	97.1	99.4	95.5	96.8	99.7	99.4	99.0	99.7	98.4	94.2	87.1
8	2	10	2	90.3	90.3	88.7	84.8	f	66.8	f	78.1	95.8	94.5	92.3	93.5	93.2	71.6	94.5	94.2	61.3	93.5	94.2	95.2	64.5	94.5	95.5	92.9	93.2	95.8	96.8	93.9	93.9	95.2	90.6	85.5
10	2	12	2	97.4	97.4	94.8	89.7	f	88.1	f	81.6	99.7	99.4	99.0	99.4	96.1	80.6	100.0	100.0	73.2	99.4	100.0	100.0	64.5	98.1	100.0	97.7	97.4	99.4	100.0	100.0	99.7	99.7	94.2	89.0
12	2	14	2	96.1	96.1	94.5	96.1	f	95.5	f	77.4	100.0	99.4	99.0	97.7	95.5	78.7	98.7	99.4	78.4	98.7	99.0	99.7	64.5	97.4	99.0	97.4	96.1	99.7	99.4	99.7	99.0	99.0	96.8	90.0

Figure 7.19: MATLAB Classification Learning Application Cross-Validation Accuracy for 31-class data set (Study 2)

Well performing classification algorithms were identified using binary accuracy results show on the bottom half of Figures 7.18 and 7.19. The resulting algorithms were selected for additional analysis (and are indicated in Figure 7.20.

1. Bagged Tree Ensemble

2. Subspace Discriminant Ensemble

3. Linear Support Vector Machine

4. Fine KNN

5. Narrow Neural Network

23-classes, BINARY only (3 Ransomwares, OS, 7zip, 18 SPEC benchmarks, 108 HPCs)

CLASSIFICATION RESULTS	ALL MODELS																														
	1	2	3	4	5		6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	Tree			Discriminant		Logistic	Naive Bayes		SVM						KNN						Ensemble					Neural Network					Kernel	
Start Time	Bin Size	Stop Time	# Classes	Fine	Medium	Coarse	Linear	Quadratic	Logistic Regressio n	Gaussian	Kernel	Linear	Quadratic	Cubic	Fine Gaussian	Medium Gaussian	Coarse Gaussian	Fine	Medium	Coarse	Cosine	Cubic	Weighted	Boosted Trees	Bagged Trees	Subspace Discrimin ant	Subspace KNN	RUS-Boosted Trees	Narrow	Medium	Wide	Bilayered	Trilayered	SVM	Logistic Regressio n
29.9	0.1	30	2	86.5	87.2	87.8	f	f	81.1	f	61.3	87.4	85.4	83.0	86.7	87.0	87.0	87.2	87.6	87.0	87.6	87.8	88.0	88.7	88.5	88.9	87.2	83.0	84.4	85.7	83.7	81.1	85.4	87.0	87.0
30	0.1	30.1	2	87.0	87.4	90.7	f	f	82.2	f	60.9	88.7	87.8	82.4	87.0	86.7	87.0	85.0	87.4	87.0	87.0	87.2	87.2	86.7	88.7	88.7	87.6	82.8	85.0	84.3	83.5	84.3	86.5	87.0	87.0
30.1	0.1	30.2	2	87.0	87.4	90.0	f	f	82.6	f	61.3	89.1	86.7	84.1	87.0	87.0	87.0	86.5	87.6	87.0	87.6	87.2	88.3	88.0	90.2	90.2	90.4	83.5	87.2	86.7	85.2	85.2	85.2	86.7	87.0
29.5	0.5	30	2	89.1	89.1	90.4	f	f	83.5	f	58.9	92.0	92.6	84.1	89.6	89.6	87.0	86.1	89.8	87.0	88.9	90.0	89.1	87.8	92.0	92.4	88.0	87.6	89.1	88.7	88.9	89.3	87.8	87.2	87.0
30	0.5	30.5	2	88.7	88.9	91.3	f	f	85.4	f	61.5	92.4	90.7	90.0	91.1	91.3	87.0	90.0	92.0	86.5	91.7	90.9	91.7	88.3	91.3	93.0	89.8	89.3	91.3	92.6	92.8	91.1	92.6	87.0	87.0
30.5	0.5	31	2	86.7	86.7	88.5	f	f	83.7	f	62.6	92.0	91.3	89.8	90.0	89.6	87.0	89.3	90.2	87.0	90.2	89.3	90.9	88.9	91.3	92.6	88.9	87.2	90.2	89.8	89.8	91.1	89.6	86.7	87.0
29	1	30	2	91.1	91.1	92.4	f	f	86.5	f	60.9	93.0	93.0	90.4	92.2	89.8	87.0	91.3	91.3	87.0	91.5	91.7	91.5	88.3	93.7	93.9	91.5	93.3	92.0	88.9	92.0	91.5	89.6	87.6	87.0
30	1	31	2	92.8	92.8	88.9	f	f	85.9	f	63.0	94.6	94.8	93.3	94.1	91.1	87.0	95.0	94.3	87.0	94.3	94.1	95.2	91.7	95.7	94.6	91.7	93.0	92.8	93.5	94.3	92.2	93.3	89.3	87.4
31	1	32	2	92.0	92.0	93.3	f	f	85.2	f	64.8	93.9	93.5	93.5	93.0	91.7	87.0	92.6	93.5	87.0	93.5	92.8	94.6	88.0	95.2	95.7	93.0	94.1	92.8	91.5	92.0	91.5	93.5	90.2	88.0
28	2	30	2	93.3	93.3	90.4	f	f	88.9	f	63.5	94.1	94.6	94.8	92.6	89.3	87.0	94.1	93.3	87.0	92.4	93.0	94.6	87.0	95.2	95.7	91.5	96.5	95.7	95.0	94.6	93.9	93.5	89.6	87.0
30	2	32	2	95.0	95.0	94.1	f	f	90.2	f	68.5	97.0	96.3	95.4	94.6	91.5	87.0	96.3	93.7	87.0	93.5	93.3	95.9	87.0	97.0	95.7	94.3	96.1	93.7	94.3	96.3	94.8	94.8	91.3	89.8
32	2	34	2	96.7	96.7	94.1	f	f	95.0	f	72.6	93.9	97.2	96.3	92.6	92.0	87.2	95.2	92.2	87.0	91.7	90.4	95.0	87..0	98.0	96.3	93.9	97.4	95.9	96.5	95.9	95.9	96.5	93.5	89.8

31-classes, BINARY only (11 Ransomwares, OS, 7zip, 18 SPEC benchmarks, 220 HPCs)

CLASSIFICATION RESULTS	ALL MODELS																														
	1	2	3	4	5		6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	Tree			Discriminant			Naive Bayes		SVM						KNN						Ensemble					Neural Network					Kernel	
Start Time	Bin Size	Stop Time	# Classes	Fine	Medium	Coarse	Linear	Quadratic	Logistic Regressio n	Gaussian	Kernel	Linear	Quadratic	Cubic	Fine Gaussian	Medium Gaussian	Coarse Gaussian	Fine	Medium	Coarse	Cosine	Cubic	Weighted	Boosted Trees	Bagged Trees	Subspace Discrimin ant	Subspace KNN	RUS-Boosted Trees	Narrow	Medium	Wide	Bilayered	Trilayered	SVM	Logistic Regressio n
9.9	0.1	10	2	88.4	88.4	80.6	f	f	60.6	f	73.9	90.0	87.4	87.1	83.9	88.4	72.6	91.0	86.5	69.7	87.4	82.6	89.4	64.5	90.0	91.9	82.3	89.7	90.0	91.3	89.7	92.3	90.6	70.6	69.7
10	0.1	10.1	2	88.4	88.4	85.2	f	f	61.0	f	74.8	86.1	91.3	87.7	87.7	83.9	73.5	91.0	86.5	68.4	85.8	84.8	88.4	64.5	91.9	90.6	84.5	91.9	91.3	89.4	91.6	88.7	89.4	74.2	73.5
10.1	0.1	10.2	2	88.7	88.7	83.5	f	f	67.1	f	76.1	91.3	92.6	91.9	88.4	80.0	75.5	90.0	88.7	71.9	90.0	87.4	89.7	64.5	93.2	92.9	85.2	92.6	91.3	92.3	91.6	89.7	91.0	76.1	74.2
9.5	0.5	10	2	91.9	91.9	88.4	f	f	58.1	f	74.2	92.6	89.4	90.3	89.4	91.6	73.5	92.6	92.3	70.3	91.9	92.3	92.6	64.5	94.5	91.3	82.9	94.2	93.2	92.9	93.5	93.2	91.9	75.5	72.3
10	0.5	10.5	2	91.6	91.6	90.6	86.8	f	74.8	f	79.4	96.5	95.5	95.5	94.5	91.0	77.4	97.7	91.6	70.6	92.9	91.6	95.5	64.5	94.2	97.4	85.2	91.0	97.4	97.1	96.5	97.4	96.8	81.9	79.4
10.5	0.5	11	2	93.2	93.2	93.5	87.1	f	83.5	f	78.1	96.8	96.8	96.8	97.1	95.2	79.7	97.1	96.5	76.1	97.4	96.8	96.5	64.5	96.1	98.4	91.9	95.2	99.0	98.7	99.0	97.7	99.4	86.5	82.6
9	1	10	2	89.4	89.4	84.8	f	f	65.2	f	78.4	93.5	91.6	90.3	91.0	91.9	73.5	94.5	91.6	69.0	92.6	92.6	93.2	64.5	93.5	91.0	92.6	93.9	92.9	92.9	92.3	94.5	93.9	87.4	77.1
10	1	11	2	92.6	92.6	90.6	88.4	f	87.4	f	79.7	98.4	97.7	98.4	97.4	94.8	80.0	99.0	97.7	73.2	98.1	98.1	97.7	64.5	97.1	98.7	93.5	97.7	100.0	98.7	99.4	98.7	99.4	91.9	83.5
11	1	12	2	95.8	95.8	93.5	f	f	92.3	f	77.1	98.1	99.4	99.4	97.1	96.1	79.0	99.4	97.4	82.6	96.8	95.8	98.7	64.5	97.1	99.4	95.5	96.8	99.7	99.4	99.0	99.7	98.4	94.2	87.1
8	2	10	2	90.3	90.3	88.7	84.8	f	66.8	f	78.1	95.8	94.5	92.3	93.5	93.2	71.6	94.5	94.2	61.3	93.5	94.2	95.2	64.5	94.5	95.5	92.9	93.2	95.8	96.8	93.9	93.9	95.2	90.6	85.5
10	2	12	2	97.4	97.4	94.8	89.7	f	88.1	f	81.6	99.7	99.4	99.0	99.4	96.1	80.6	100.0	100.0	73.2	99.4	100.0	100.0	64.5	98.1	100.0	97.7	97.4	99.4	100.0	100.0	99.7	99.7	94.2	89.0
12	2	14	2	96.1	96.1	94.5	96.1	f	95.5	f	77.4	100.0	99.4	99.0	97.7	95.5	78.7	98.7	99.4	78.4	98.7	99.0	99.7	64.5	97.4	99.0	97.4	96.1	99.7	99.4	99.7	99.0	99.0	96.8	90.0

Figure 7.20: Five Classification Algorithms were selected based on their performance across both data sets.

7.5.3 HPC Feature Selection

In order to reduce the dimensions of the data sets, multiple methods of feature selection were explored. First, the feature ranking algorithms included in MATLAB's Classification Learning app were examined for both the multi-class and binary versions of the 23- and 31-class data sets, as discussed in the section below. Since both data sets are unbalanced, the second approach examined cross validation accuracy, precision, recall, and F1 scores for individual features as an alternative to MATLAB's Classification Learner results.

7.5.3.1 Feature Selection in MATLAB Classification Learner App

In addition to recording cross-validation accuracy results, the full sets of HPC importance scores from the MRMR, Chi2, ReliefF, ANOVA, and Kruskal Wallis algorithms were recorded. Additional information on each feature ranking algorithm, summarized from [209] is provided

below:

1. **Minimum Redundancy Maximum Relevance (MRMR)** - finds optimal set of features that is mutually and maximally dissimilar using the MRMR algorithm, which quantifies redundancy and relevance in terms of mutual information.

2. **Chi2** - uses individual Chi-squared tests to determine if each predictor variable is independent of the response variable and then ranks features using p-values (the probability that a statistical measure will be greater than or equal to the observed results).

3. **ReliefF** - uses ReliefF algorithm, which ranks predictors for multiclass variables.

4. **ANOVA** - performs one-way Analysis of Variance (ANOVA), which tests whether two or more normally distributed populations have equal means for each predictor variable, and then ranks features using p-values.

5. **Kruskal Wallis** - uses Kruskal-Wallis Test, which compares medians of groups of data to determine if samples come from the same populations, and then ranks features using p-values. Kruskal-Wallis is a nonparametric version of ANOVA, but assumes continuous distributions and that populations are mutually independent, rather than normally distributed.

In order to provide a weighted ranking of HPC importance across all sets of start/stop times and windows for both sets of data collection, the importance score for each HPC in each set of features was summed across all HPCs and all sets of features for each class to obtain an overall importance score for each HPC and each ranking algorithm.

An example of weighting importance scores for the 31-class and binary features for the MRMR algorithm is shown in Figure 7.21. Sets of features are color coded by window size,

consistent with the Figures 7.15 and 7.16, where start/stop time combos for the window sizes of 0.1 seconds is white, while the start/stop time combos for window sizes of 0.5 seconds, 1 second, and 2 seconds are increasingly darker shades of yellow. In the color coded section of the 31-class feature set, HPC rankings and importance scores are listed in ranking order for the top and bottom 10 ranked HPCs. On the right hand side, importance scores are summed across every HPC and ranking for the first 5 and last 10 HPCs for the 31-class classification rankings and for the first 10 rankings only for the binary classification rankings.

For the binary feature sets, the top ranked HPC for the 0.5 second window from 9.5 to 10 seconds is in red bold font (HPC 1, Importance score = 0.217). Meanwhile, HPC 212 is the top ranked feature in 3 sets, with an importance score of 0.3174 for the 0.5 second window from 10.5 to 11 seconds, the 1 second window from 10 to 11 seconds (Importance score = 0.2829), and the 2 second window from 8 to 10 seconds (Importance score = 0.2829), all of which are indicated in purple bold font. Those scores are summed across the first row, with the rounded sum of 0.883 (0.3174 + 0.2829 + 0.2829 = 0.8832) also shown in purple bold font under the 212 column on the right. This is performed for all 220 feature rankings for all 220 HPCs and then summed to obtain a weighted average for each counter. The section outlined in orange shows the summed importance score across the top 10 rows, which is also performed for the remaining 210 rows to obtain the weighted value of 1.264.

The weighted importance scores for each HPC were sorted from high to low for each feature ranking algorithm. The top 10 HPCs of the 31-class experiment for all 5 ranking algorithms are shown in Figure 7.22. With no clear convergence in dominant HPCs, this approach was abandoned in favor of the statistics-based ranking methods described in the following sections.

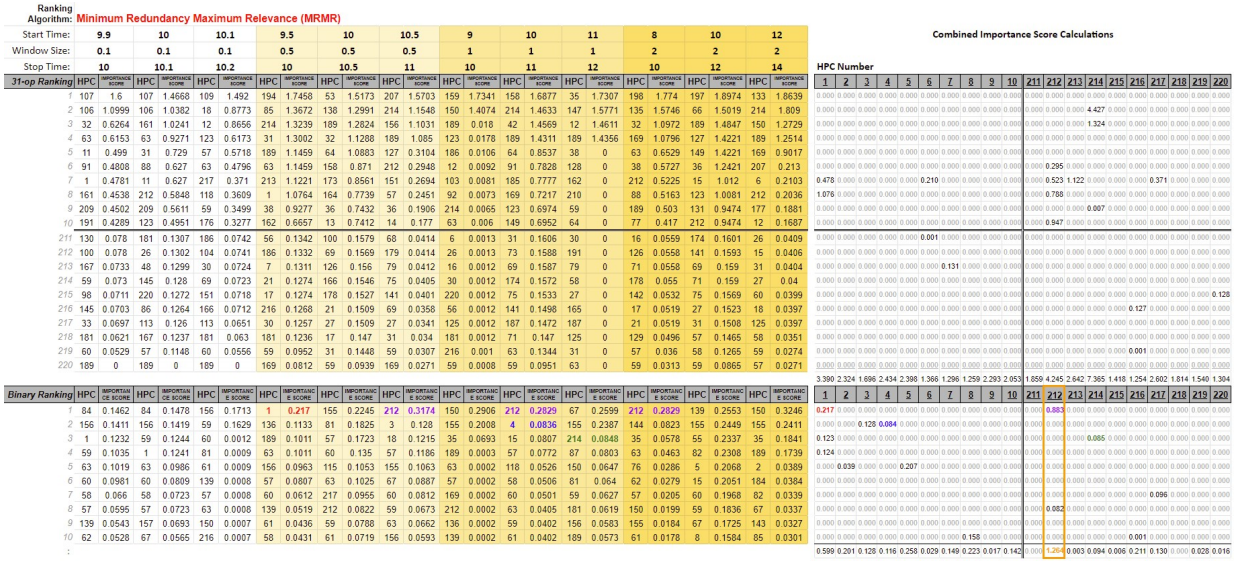


Figure 7.21: Weighted importance score calculation example for the 31-class (220 HPC) data set and the MRMR algorithm. Feature ranking rows 11-210 are hidden for the 31-operation classification ranking results, while feature rows 11-220 are hidden for the binary classification ranking results. On the right hand side, weighted importance scores are summed across each row for each HPC to get the total importance score for each HPC and ranking. Importance scores are then summed by column to get a weighted importance score for each HPC for all rankings and data sets.

Feature Ranking:		MRMR			Chi2			ReliefF			ANOVA			Kruskal Wallis		
31	23	HPC	Importance	Description	HPC	Importance	Description	HPC	Importance	Description	HPC	Importance	Description	HPC	Importance	Description
1	1	189	9.637	SQ_MISC.SPLIT_LOCK	193	6130.2	UOPS_DISPATCHED_PORT.PORT_1	19	3.213	CPU_CLK_UNHALTED.REF_TSC	1	INF	ARITH.DIVIDER_ACTIVE	158	1168.4	MEM_INST_RETIRED.ALL_LOADS_PS
2	2	214	7.365	OPS_ISSUED.SLOW_LEA	171	5993.5	MEM_LOAD_RETIRED.L1_HIT_PS	20	3.210	CPU_CLK_UNHALTED.REF_XCLK	22	INF	CPU_CLK_UNHALTED.RING0_TRANS	171	1162.1	MEM_LOAD_RETIRED.L1_HIT_PS
3	3	12	5.780	BR_MISP_RETIRED.CONDITIONAL_PS	92	5954.1	IDQ.DSB_UOPS	16	3.210	CPU_CLK_THREAD_UNHALTED.REF_XCLK	35	INF	DSB2MITE_SWITCHES.COUNT	194	1158.8	UOPS_DISPATCHED_PORT.PORT_2
4	X	107	4.846	INST_RETIRED.ANY	87	5910.2	IDQ.ALL_DSB_CYCLES_4_UOPS	23	3.171	CPU_CLK_UNHALTED.THREAD_ANY	36	INF	DSB2MITE_SWITCHES.PENALTY_CYCLES	195	1157.6	UOPS_DISPATCHED_PORT.PORT_3
5	X	123	4.757	L1D_PEND_MISS.FB_FULL	206	5890.6	UOPS_EXECUTED.CYCLES_GE_1_UOP_EXEC	24	3.169	CPU_CLK_UNHALTED.THREAD_P	38	INF	DTLB_LOAD_MISSES.STLB_HIT	107	1154.5	INST_RETIRED.ANY
6	X	150	4.575	LD_BLOCKS.STORE_FORWARD	158	5878.0	MEM_INST_RETIRED.ALL_LOADS_PS	29	2.966	CYCLE_ACTIVITY.CYCLES_MEM_ANY	44	INF	DTLB_STORE_MISSES.MISS_CAUSES_A_WALK	108	1153.4	INST_RETIRED.ANY_P
7	X	159	4.521	MEM_INST_RETIRED.ALL_STORES_PS	194	5865.2	UOPS_DISPATCHED_PORT.PORT_2	220	2.935	UOPS_RETIRED.TOTAL_CYCLES	45	INF	DTLB_STORE_MISSES.STLB_HIT	109	1152.9	INST_RETIRED.PREC_DIST
8	X	63	4.448	FP_ASSIST.ANY	53	5861.3	EXE_ACTIVITY.3_PORTS_UTIL	25	2.931	CPU_CLK_UNHALTED.THREAD_P_ANY	46	INF	DTLB_STORE_MISSES.WALK_ACTIVE	29	1152.3	CYCLE_ACTIVITY.CYCLES_MEM_ANY
9	X	212	4.245	UOPS_EXECUTED.X87	192	5840.1	UOPS_DISPATCHED_PORT.PORT_0	110	2.912	INST_RETIRED.TOTAL_CYCLES_PS	47	INF	DTLB_STORE_MISSES.WALK_COMPLETED	170	1152.2	MEM_LOAD_RETIRED.FB_HIT_PS
10	X	32	4.151	CYCLE_ACTIVITY.STALLS_L3_MISS	88	5835.3	IDQ.ALL_DSB_CYCLES_ANY_UOPS	17	2.893	CPU_CLK_THREAD_UNHALTED.REF_XCLK_ANY	49	INF	DTLB_STORE_MISSES.WALK_COMPLETED_4K	53	1145.3	EXE_ACTIVITY.3_PORTS_UTIL

Figure 7.22: Weighted importance scores by algorithm for the to 10 HPCs in the 31-class experiment, ranked from highest to lowest importance score value.

7.5.3.2 Single-HPC Accuracy-based Feature Selection

Cross Validation Accuracy was explored in more depth for the highest performing classifiers for both the multi-class and binary versions of both the 23- and 31-class data sets. To accomplish this, MATLAB code was developed to train the Bagged Tree and Subspace Discriminant Ensemble Classifiers individually, one HPC feature at a time, for each start-window-stop time combination. HPC features were ranked by accuracy for each classifier, with the top 20 single HPC features that produced the highest accuracy values for the Bagged Tree Ensemble Classifier in Figure 7.23, and the corresponding rankings for the Subspace Discriminant Ensemble Classifier in Figure 7.24.

BAGGED TREE ENSEMBLE CROSS-VALIDATION ACCURACY

	23-class - Sorted High-to-Low				31-class - Sorted High-to-Low				23-class BINARY - Sorted				31-class BINARY - Sorted			
	HPC		Description	Mean	HPC		Description	Mean	HPC		Description	Mean	HPC		Description	Mean
1	52	107	INST_RETIRED.ANY'	0.5109	107	52	INST_RETIRED.ANY	0.4976	69	161	MEM_INST_RETIRED.SPLIT_LOADS_PS'	0.8779	107	52	INST_RETIRED.ANY	0.7876
2	8	23	CPU_CLK_UNHALTED.THREAD'	0.4268	194	93	UOPS_DISPATCHED_PORT.PORT_2	0.4876	71	163	MEM_INST_RETIRED.STLB_MISS_LOADS_PS'	0.8741	207	X	UOPS_EXECUTED.CYCLES_GE_2_UOPS_EXEC	0.7796
3	6	19	CPU_CLK_UNHALTED.REF_TSC'	0.3875	207	X	UOPS_EXECUTED.CYCLES_GE_2_UOPS_EXEC	0.4841	33	67	FRONTEND_RETIRED.L2_MISS_PS'	0.8732	212	104	UOPS_EXECUTED.X87	0.7780
4	77	171	MEM_LOAD_RETIRED.L1_HIT_PS'	0.3846	88	42	IDQ.ALL_DSB_CYCLES_ANY_UOPS	0.4839	1	1	ARITH.DIVIDER_ACTIVE'	0.8728	195	94	UOPS_DISPATCHED_PORT.PORT_3	0.7774
5	92	193	UOPS_DISPATCHED_PORT.PORT_1'	0.3786	197	96	UOPS_DISPATCHED_PORT.PORT_5	0.4796	80	175	MEM_LOAD_RETIRED.L3_HIT_PS'	0.8716	70	34	FRONTEND_RETIRED.LATENCY_GE_16_PS	0.7758
6	91	192	UOPS_DISPATCHED_PORT.PORT_0'	0.3728	171	77	MEM_LOAD_RETIRED.L1_HIT_PS	0.4793	14	32	CYCLE_ACTIVITY.STALLS_L3_MISS'	0.8707	194	93	UOPS_DISPATCHED_PORT.PORT_2	0.7718
7	93	194	UOPS_DISPATCHED_PORT.PORT_2'	0.3688	218	108	UOPS_RETIRED.RETIRE_SLOTS	0.4793	37	81	FRONTEND_RETIRED.STLB_MISS_PS'	0.8707	53	X	EXE_ACTIVITY.3_PORTS_UTIL	0.7718
8	45	92	IDQ.DSB_UOPS'	0.3643	92	45	IDQ.DSB_UOPS	0.4761	56	112	INT_MISC.RECOVERY_CYCLES'	0.8701	171	77	MEM_LOAD_RETIRED.L1_HIT_PS	0.7680
9	108	218	UOPS_RETIRED.RETIRE_SLOTS'	0.3614	192	91	UOPS_DISPATCHED_PORT.PORT_0	0.4747	39	X	ICACHE_16B.IFDATA_STALL:cmask=1:e=yes'	0.8701	211	103	UOPS_EXECUTED.THREAD	0.7656
10	22	51	EXE_ACTIVITY.1_PORTS_UTIL'	0.3591	206	X	UOPS_EXECUTED.CYCLES_GE_1_UOP_EXEC	0.4742	4	11	BR_MISP_RETIRED.ALL_BRANCHES_PS'	0.8697	7	X	BR_INST_RETIRED.NEAR_CALL_PS	0.7653
11	23	52	EXE_ACTIVITY.2_PORTS_UTIL'	0.3580	53	X	EXE_ACTIVITY.3_PORTS_UTIL	0.4715	51	106	ILD_STALL.LCP'	0.8697	92	45	IDQ.DSB_UOPS	0.7626
12	96	197	UOPS_DISPATCHED_PORT.PORT_5'	0.3543	91	X	IDQ.DSB_CYCLES	0.4715	18	38	DTLB_LOAD_MISSES.STLB_HIT:cmask=1'	0.8696	8	X	BR_INST_RETIRED.NEAR_RETURN_PS	0.7626
13	15	33	CYCLE_ACTIVITY.STALLS_MEM_ANY'	0.3522	195	94	UOPS_DISPATCHED_PORT.PORT_3	0.4712	24	55	EXE_ACTIVITY.BOUND_ON_STORES'	0.8696	150	62	LD_BLOCKS.STORE_FORWARD	0.7575
14	67	X	MEM_INST_RETIRED.ANY'	0.3516	158	X	MEM_INST_RETIRED.ALL_LOADS_PS	0.4696	25	57	FP_ARITH_INST_RETIRED.128B_PACKED_DOUBLE'	0.8696	218	108	UOPS_RETIRED.RETIRE_SLOTS	0.7559
15	3	3	BR_INST_RETIRED.ALL_BRANCHES'	0.3495	87	41	IDQ.ALL_DSB_CYCLES_4_UOPS	0.4683	26	58	FP_ARITH_INST_RETIRED.128B_PACKED_SINGLE'	0.8696	158	X	MEM_INST_RETIRED.ALL_LOADS_PS	0.7551
16	54	109	INST_RETIRED.PREC_DIST'	0.3487	51	22	EXE_ACTIVITY.1_PORTS_UTIL	0.4683	27	59	FP_ARITH_INST_RETIRED.256B_PACKED_DOUBLE'	0.8696	159	66	MEM_INST_RETIRED.ALL_STORES_PS	0.7546
17	94	195	UOPS_DISPATCHED_PORT.PORT_3'	0.3462	213	105	UOPS_ISSUED.ANY	0.4645	28	60	FP_ARITH_INST_RETIRED.256B_PACKED_SINGLE'	0.8696	33	15	CYCLE_ACTIVITY.STALLS_MEM_ANY	0.7540
18	97	198	UOPS_DISPATCHED_PORT.PORT_6'	0.3433	198	97	UOPS_DISPATCHED_PORT.PORT_6	0.4634	29	61	FP_ARITH_INST_RETIRED.SCALAR_DOUBLE'	0.8696	213	105	UOPS_ISSUED.ANY	0.7538
19	95	196	UOPS_DISPATCHED_PORT.PORT_4'	0.3420	193	92	UOPS_DISPATCHED_PORT.PORT_1	0.4581	30	62	FP_ARITH_INST_RETIRED.SCALAR_SINGLE'	0.8696	1	1	ARITH.DIVIDER_ACTIVE	0.7535
20	105	213	UOPS_ISSUED.ANY'	0.3409	159	66	MEM_INST_RETIRED.ALL_STORES_PS	0.4562	53	X	INST_RETIRED.NOP'	0.8696	206	X	UOPS_EXECUTED.CYCLES_GE_1_UOP_EXEC	0.7530

Figure 7.23: Single-HPC Feature Classification Accuracy for top 20 Features using Bagged Tree Ensemble Classification Algorithm for multi-class and binary versions of Studies 1 and 2

SUBSPACE DISCRIMINANT ENSEMBLE CROSS-VALIDATION ACCURACY

	23-class - Sorted High-to-Low				31-class - Sorted High-to-Low				23-class BINARY - Sorted				31-class BINARY - Sorted			
	HPC		Description	Mean	HPC		Description	Mean	HPC		Description	Mean	HPC		Description	Mean
1	52	107	INST_RETIRED.ANY'	0.4536	92	45	IDQ.DSB_UOPS	0.4632	37	81	FRONTEND_RETIRED.STLB_MISS_PS'	0.8701	67	33	FRONTEND_RETIRED.L2_MISS_PS	0.7737
2	8	23	CPU_CLK_UNHALTED.THREAD'	0.3837	197	96	UOPS_DISPATCHED_PORT.PORT_5	0.4578	33	67	FRONTEND_RETIRED.L2_MISS_PS'	0.8696	66	X	FRONTEND_RETIRED.L1I_MISS_PS	0.7642
3	77	171	MEM_LOAD_RETIRED.L1_HIT_PS'	0.3676	192	91	UOPS_DISPATCHED_PORT.PORT_0	0.4573	1	1	ARITH.DIVIDER_ACTIVE'	0.8696	139	X	L2_RQSTS.CODE_RD_MISS	0.7511
4	108	218	UOPS_RETIRED.RETIRE_SLOTS'	0.3574	218	108	UOPS_RETIRED.RETIRE_SLOTS	0.4505	3	3	BR_INST_RETIRED.ALL_BRANCHES'	0.8696	85	X	ICACHE_64B.IFTAG_MISS	0.7495
5	3	3	BR_INST_RETIRED.ALL_BRANCHES'	0.3572	193	92	UOPS_DISPATCHED_PORT.PORT_1	0.4503	4	11	BR_MISP_RETIRED.ALL_BRANCHES_PS'	0.8696	132	X	L2_RQSTS.ALL_CODE_RD	0.7476
6	49	100	IDQ_UOPS_NOT_DELIVERED.CORE'	0.3560	87	41	IDQ.ALL_DSB_CYCLES_4_UOPS	0.4500	6	19	CPU_CLK_UNHALTED.REF_TSC'	0.8696	138	X	L2_RQSTS.CODE_RD_HIT	0.7392
7	92	193	UOPS_DISPATCHED_PORT.PORT_1'	0.3560	107	52	INST_RETIRED.ANY	0.4484	7	20	CPU_CLK_UNHALTED.REF_XCLK'	0.8696	76	X	FRONTEND_RETIRED.LATENCY_GE_32_PS	0.7210
8	91	192	UOPS_DISPATCHED_PORT.PORT_0'	0.3496	88	42	IDQ.ALL_DSB_CYCLES_ANY_UOPS	0.4457	8	23	CPU_CLK_UNHALTED.THREAD'	0.8696	83	38	ICACHE_16B.IFDATA_STALL	0.7164
9	105	213	UOPS_ISSUED.ANY'	0.3496	194	93	UOPS_DISPATCHED_PORT.PORT_2	0.4454	9	24	CPU_CLK_UNHALTED.THREAD_P'	0.8696	2	2	BACLEARS.ANY	0.7116
10	15	33	CYCLE_ACTIVITY.STALLS_MEM_ANY'	0.3487	195	94	UOPS_DISPATCHED_PORT.PORT_3	0.4446	10	26	CYCLE_ACTIVITY.CYCLES_L1D_MISS'	0.8696	80	X	FRONTEND_RETIRED.LATENCY_GE_8_PS	0.7097
11	93	194	UOPS_DISPATCHED_PORT.PORT_2'	0.3482	53	X	EXE_ACTIVITY.3_PORTS_UTIL	0.4441	11	29	CYCLE_ACTIVITY.CYCLES_MEM_ANY'	0.8696	18	5	CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE	0.7024
12	45	92	IDQ.DSB_UOPS'	0.3462	207	X	UOPS_EXECUTED.CYCLES_GE_2_UOPS_EXEC	0.4403	12	30	CYCLE_ACTIVITY.STALLS_L1D_MISS'	0.8696	70	34	FRONTEND_RETIRED.LATENCY_GE_16_PS	0.7011
13	54	109	INST_RETIRED.PREC_DIST'	0.3429	91	X	IDQ.DSB_CYCLES	0.4395	13	31	CYCLE_ACTIVITY.STALLS_L2_MISS'	0.8696	79	X	FRONTEND_RETIRED.LATENCY_GE_64_PS	0.7005
14	22	51	EXE_ACTIVITY.1_PORTS_UTIL'	0.3426	213	105	UOPS_ISSUED.ANY	0.4374	14	32	CYCLE_ACTIVITY.STALLS_L3_MISS'	0.8696	15	X	CPU_CLK_THREAD_UNHALTED.ONE_THREAD_ACTIVE	0.6997
15	16	34	CYCLE_ACTIVITY.STALLS_TOTAL'	0.3417	206	X	UOPS_EXECUTED.CYCLES_GE_1_UOP_EXEC	0.4358	15	33	CYCLE_ACTIVITY.STALLS_MEM_ANY'	0.8696	156	X	MACHINE_CLEARS.MEMORY_ORDERING	0.6995
16	31	X	FRONTEND_RETIRED.ANY_DSB_MISS'	0.3395	198	97	UOPS_DISPATCHED_PORT.PORT_6	0.4347	16	34	CYCLE_ACTIVITY.STALLS_TOTAL'	0.8696	116	X	ITLB_MISSES.STLB_HIT	0.6992
17	97	198	UOPS_DISPATCHED_PORT.PORT_6'	0.3377	4	X	BR_INST_RETIRED.CONDITIONAL_PS	0.4325	17	36	DSB2MITE_SWITCHES.PENALTY_CYCLES'	0.8696	167	X	MEM_LOAD_L3_HIT_RETIRED.XSNP_MISS_PS	0.6855
18	67	X	MEM_INST_RETIRED.ANY'	0.3333	33	15	CYCLE_ACTIVITY.STALLS_MEM_ANY	0.4317	18	38	DTLB_LOAD_MISSES.STLB_HIT:cmask=1'	0.8696	65	X	FRONTEND_RETIRED.ITLB_MISS_PS	0.6817
19	23	52	EXE_ACTIVITY.2_PORTS_UTIL'	0.3299	171	77	MEM_LOAD_RETIRED.L1_HIT_PS	0.4315	19	39	DTLB_LOAD_MISSES.WALK_ACTIVE'	0.8696	160	68	MEM_INST_RETIRED.LOCK_LOADS_PS	0.6774
20	103	211	UOPS_EXECUTED.THREAD'	0.3297	159	66	MEM_INST_RETIRED.ALL_STORES_PS	0.4306	20	45	DTLB_STORE_MISSES.STLB_HIT:cmask=1'	0.8696	81	37	FRONTEND_RETIRED.STLB_MISS_PS	0.6745

Figure 7.24: Single-HPC Feature Classification Accuracy for top 20 Features using Subspace Discriminant Ensemble Classification Algorithm for multi-class and binary versions of Studies 1 and 2

Both classifiers were then trained by incrementing the number of features by 1 to obtain a new accuracy value based on the number of features used. These accuracy values for each data set were plotted against the number of features required to obtain the accuracy value. Figures 7.25 and 7.26 show the cumulative cross validation accuracy results for the Bagged Tree Ensemble and Subspace Discriminant Ensemble Classifiers, respectively. In both figures, the multi-class and binary versions of the 23-class data set are in the two plots, while the comparable versions of the 31-class data sets are in the bottom two plots.

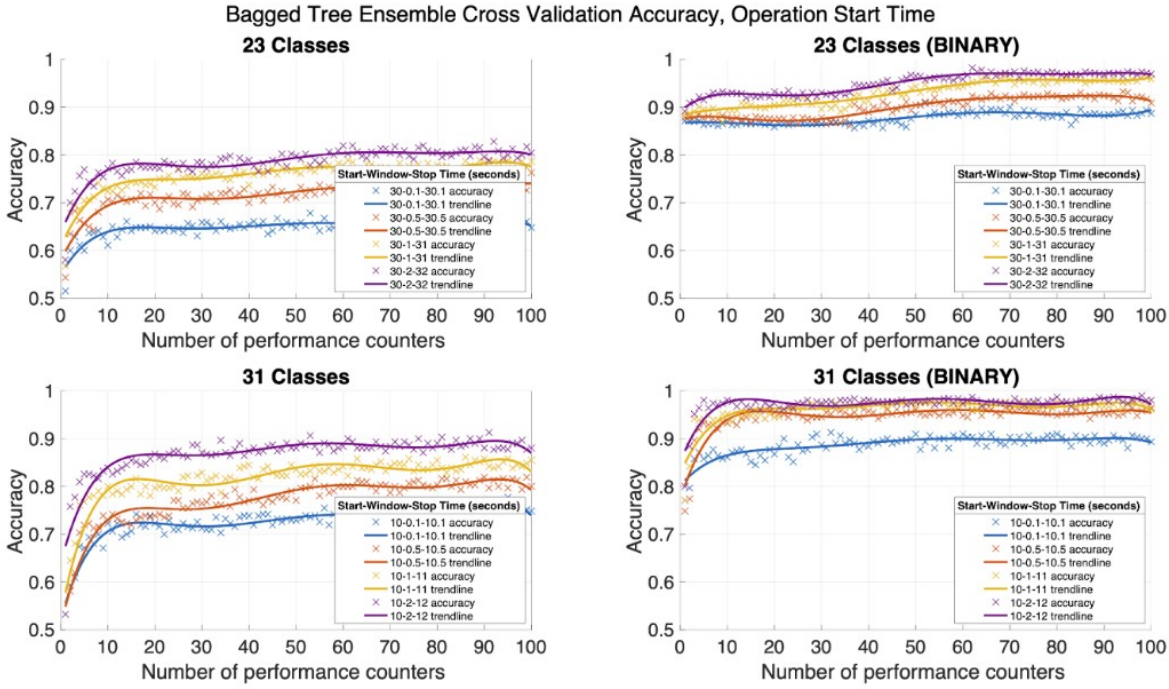


Figure 7.25: Cross Validation Accuracy for incremental performance counters in order from highest to lowest individual accuracy values for Bagged Tree Ensemble Classifier. 23-class data sets are shown in the top two plots, with 31-class data sets in the bottom two plots. The multi-class version of each data set is on the left, while the binary version of each data set is on the right.

The comparison of cumulative cross validation accuracy results for each start time, window size, stop time combination of features for Study 1 (23-class) and Study 2 (31-class) are shown in Figures 7.27(multi-class data set) and 7.28 (binary data set). For the multi-class data sets

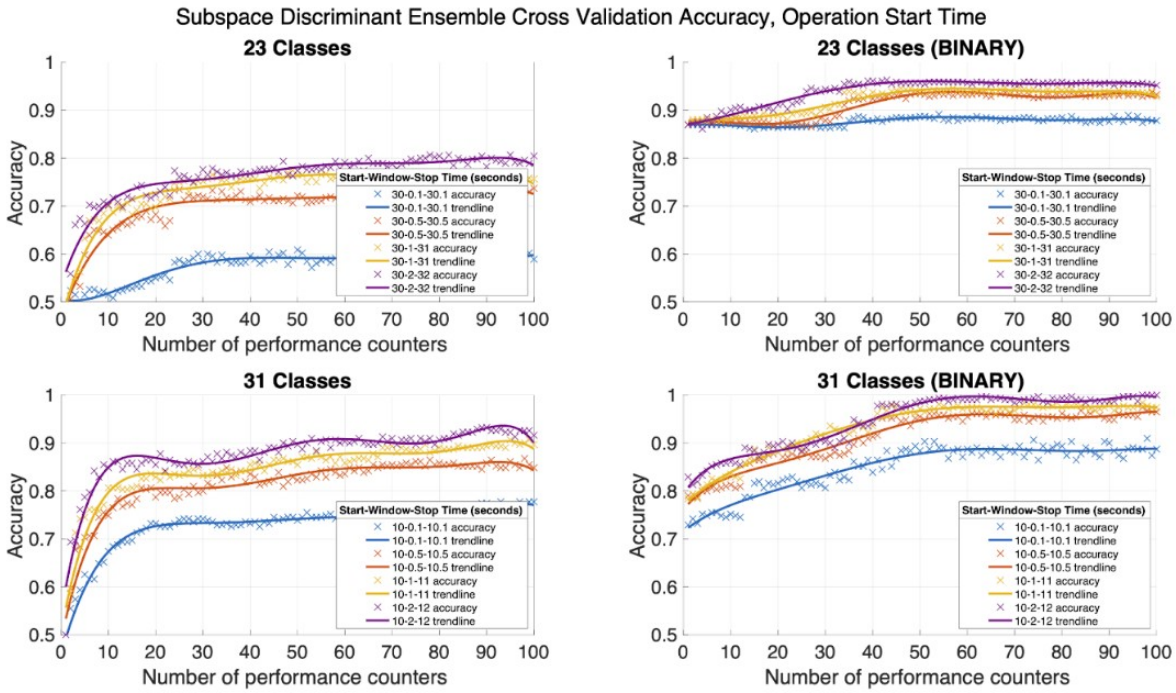


Figure 7.26: Cross Validation Accuracy for incremental performance counters in order from highest to lowest individual accuracy values for Subspace Discriminant Ensemble Classifier. 23-class data sets are shown in the top two plots, with 31-class data sets in the bottom two plots. The multi-class version of each data set is on the left, while the binary version of each data set is on the right.

(above), both data sets demonstrated relatively steady accuracy between 10 and 20 counters, and showed higher accuracy values as the window size increased from 0.1 s to 2 seconds. The 31-class accuracy values ranged from approximately 0.7 for 0.1 s of data to 0.9 for 2 seconds of data, while 23-class accuracy values settled at approximately 0.6 for 0.1 second of data to 0.8 for 2 seconds of data.

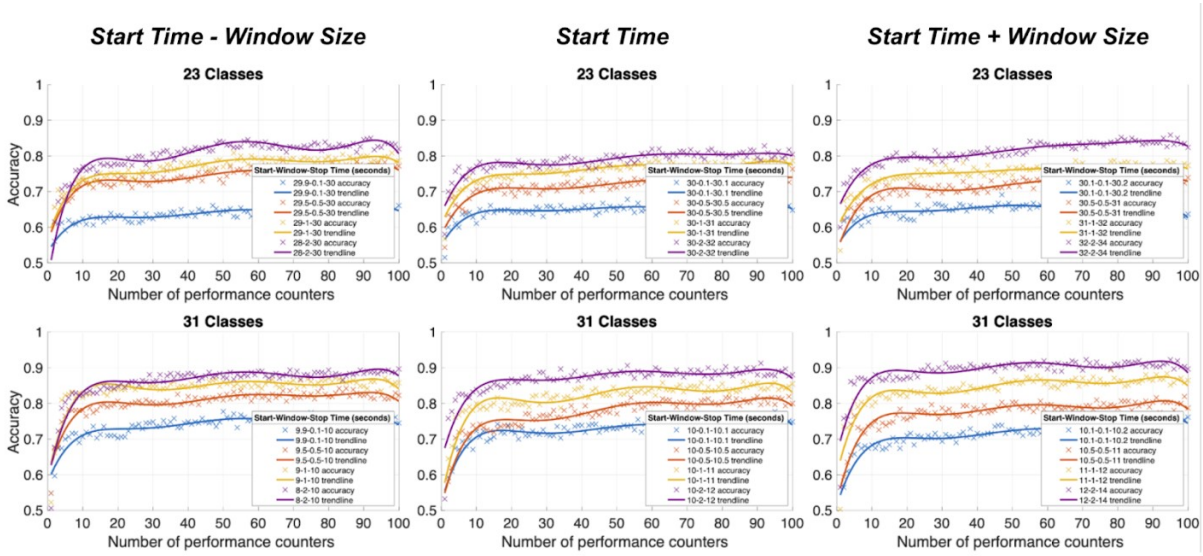


Figure 7.27: Comparison of Bagged Tree Classifier Cross Validation Accuracy for incremental performance counters in order from highest to lowest individual accuracy values for windows of 0.1 s, 0.5 s, 1 s, and 2 s before, at, and after operation start time. 23-class multi-class data sets are shown across the top, with 31-class multi-class data sets across the bottom.

As expected, higher accuracy was achieved with binary data sets (below), as the classifier was only making a determination between benign operation or ransomware. Additionally, the HPC features collected in the 31-class data set appear to be more indicative of ransomware. However, as both data sets were unbalanced, with 20 benign operations vs. 3 ransomware operations for the 23-class data set and 20 benign operations vs. 11 ransomware operations for the 31-class data set, accuracy alone was a misleading metric. For example, any classifier could classify every single operation in the 23-class data set as benign and obtain $\sim$87% accuracy

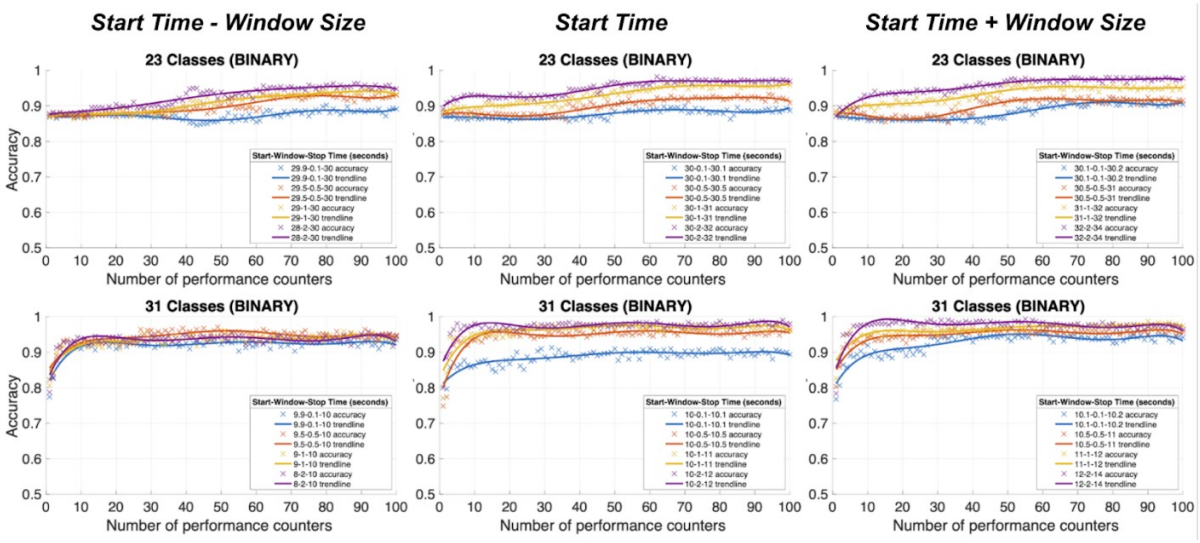


Figure 7.28: Comparison of Bagged Tree Classifier Cross Validation Accuracy for incremental performance counters in order from highest to lowest individual accuracy values for windows of 0.1 s, 0.5 s, 1 s, and 2 s before, at, and after operation start time. 23-class BINARY data sets are shown across the top, with 31-class BINARY data sets across the bottom.

(20/23) while classifying 100% of the ransomware operations incorrectly. This same error in the 31-class data set produces ∼64.5% accuracy (20/31), another indicator that the HPC features in Study 2 were better suited than the Study 1 HPC features to capture ransomware. Regardless, for unbalanced data sets, cross validation accuracy alone is an insufficient metric to narrow the selection of HPC features from 220 to a more managable number.

7.5.3.3 Single-HPC F1-based Feature Selection

The method of feeding a single-HPC feature at a time to the selected classification algorithms was extended to calculate precision, recall, and F1 for the 5 top-performing classifiers from Section 7.5.2, in addition to accuracy. Since both data sets are unbalanced (20 benign operations vs. 3 ransomware operations for the 23-class data set and 20 benign operations vs. 11 ransomware operations for the 31-class data set), it is essential to consider metrics that balance False Positives

and False Negatives. The following terms and performance metrics are commonly used for these purposes:

Performance Metrics

- Confusion Matrix. A confusion matrix (Figure 7.29) is standard way to visualize supervised machine learning outcomes. It depicts the actual classes of input data on the y-axis, the classifier output predictions on the x-axis, and presents results in terms of numbers or percentages of True Positives, True Negatives, False Positives, and False Negatives, as described below.

- True Positive (TP) - A TP occurs when the classification model predicts the positive class correctly (when the actual value and the predicted value are both positive). True Positives are shown in Quadrant II of Figure 7.29.

- True Negative (TN) - A TN occurs when the classification model correctly predicts the negative class (when the actual value and the predicted value are both negative). True Negatives are shown in Quadrant IV of Figure 7.29.

- False Positive (FP) - A FP occurs when the model predicts the positive class but the actual class is negative. This is considered a "Type I error." False Positives are shown in Quadrant III of Figure 7.29.

- False Negative (FN) - A FN occurs when the negative class if predicted by the actual class is positive. This is considered a "Type II error." False Negatives are shown in Quadrant I of Figure 7.29.

- Accuracy ($Accuracy = \frac{TP+TN}{TP+FN+FP+TN}$) - is the number of correct predictions over the

total number of predictions.

- Precision ($Precision = \frac{TP}{TP+FP}$) - Precision indicates what proportion of predicted positive classes were truly positive, or the percentage of correct positive predictions made.

- Recall ($Recall = \frac{TP}{TP+FN}$, aka Sensitivity) - Recall gives the proportion of positive predictions made out of all positive samples in the data set, or what proportion of actual positive predictions are correctly classified.

- F1 Score (Equation 7.1) - The F1 Score combines Precision and Recall into a single metric by taking the harmonic mean. Since it accounts for both FP and FN, it is a better metric for imbalance data sets.

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{7.1}$$

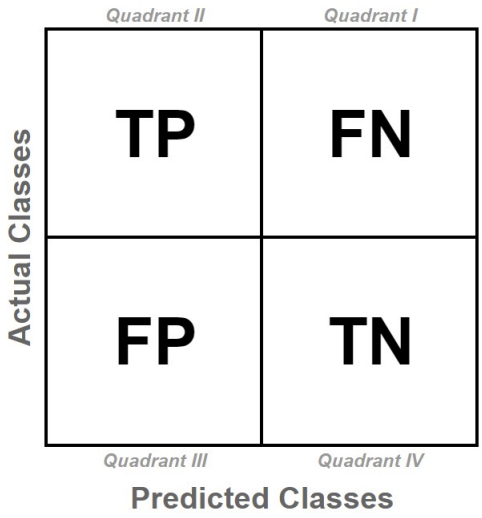


Figure 7.29: A Confusion Matrix is a standard way to visualize supervised machine learning outcomes. False Negatives (FN) are placed in Quadrant I, True Positives (TP) are placed in Quadrant II, False Positives (FP) are placed in Quadrant III, and True Negatives (TN) are in Quadrant IV

For ransomware detection, it is imperative that ransomware operations are not missed, while at the same time not falsely identifying benign operations as ransomware. To accomplish this, precision, recall, and F1 scores were calculated in addition to cross validation accuracy for every possible HPC/feature, one HPC at a time for the five classification algorithms identified in section 7.5.2.

The individual Cross Validation Confusion Matrix was generated for every possible combination of performance counter, classifier, and feature start-stop time (12 feature sets x 108 HPCs = 1296 Confusion Matrices for the 23-class data set and 12 feature sets x 220 HPCs = 2640 Confusion Matrices for the 31-class data set) per classifier. Of the 3,936 Confusion Matrices generated for each of the 5 classifiers, roughly a quarter produced a worst case Cross Validation Confusion Matrix like the one shown below, where every single operation was identified as benign.

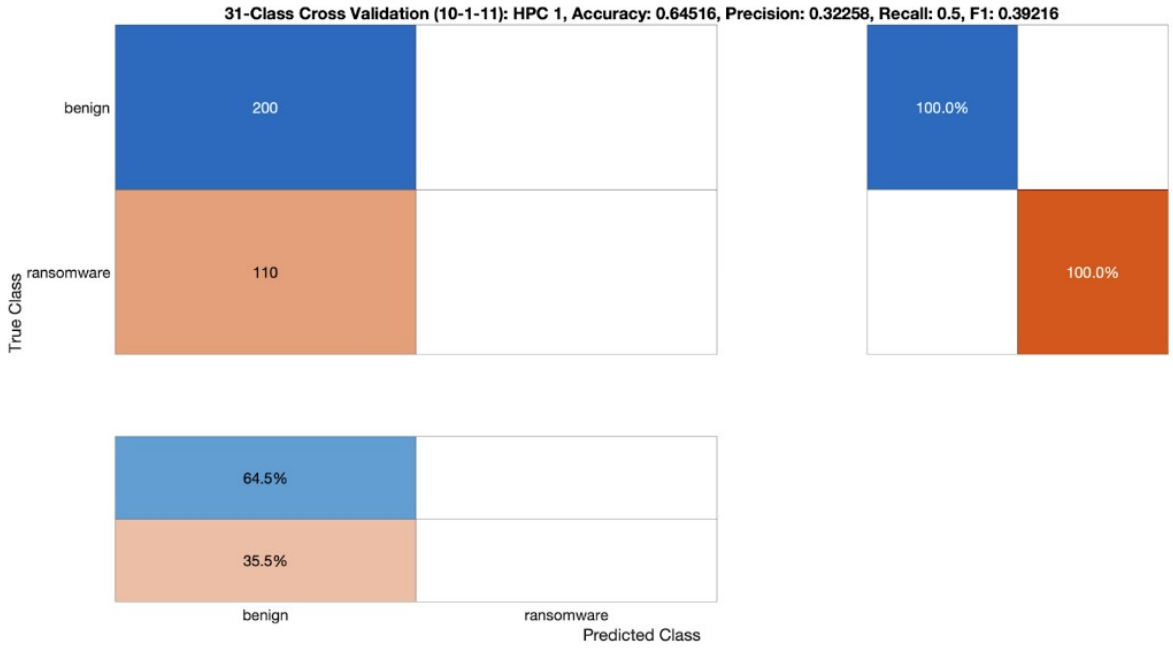


Figure 7.30: Example “worst case” Confusion Matrix for single-HPC classification tests.

The single-feature best-case Confusion Matrix (Figure 7.31) was produced by the Fine

KNN Classifier with a 2 second feature set using HPC feature 8 (BR_INST_RETIRED.NEAR_RETURN_PS), which correctly identified 179 of 200 benign operations and 91 of 110 ransomware operations using just the single feature, for an overall accuracy of 87.1%, overall precision of 85.5%, overall recall of 86.1%, and combined F1-score of 0.8597.

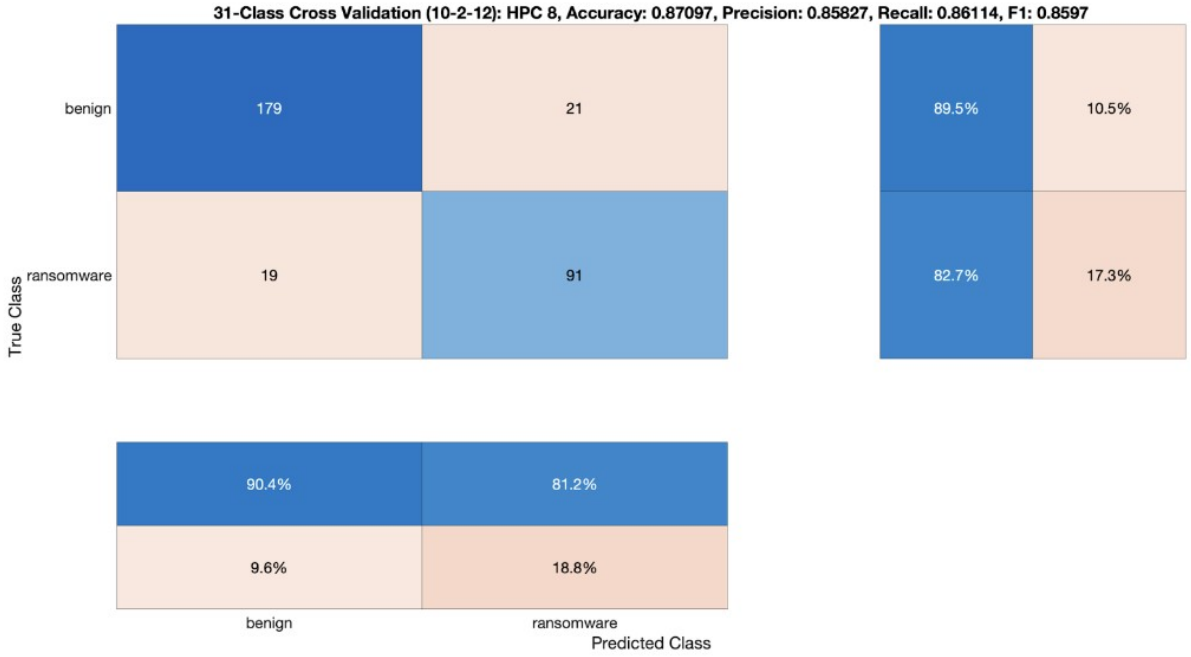


Figure 7.31: Best single-HPC Confusion Matrix resulting from training 5 classification algorithms with a single HPC feature at a time. This Confusion Matrix shows the cross validation results of a Fine KNN classifier trained with 2 seconds of data and the single HPC feature BR_INST_RETIRED.NEAR_RETURN_PS.

In order to reduce the number of features to collect, the F1 score was selected as a metric with which to rank HPC features, as it better accounts for unbalanced data, by balancing the percent of correct "positive" predicitons with what proportion of the "positive" predictions were correct. The F1 score for each HPC feature was averaged over the 12 start time - stop time feature sets for each of the 5 classifiers. An example showing the individual F1 scores for each start/stop time window and the mean F1 score for the 1st 24 HPCs of the Fine KNN classifier is shown in Figure 7.32.

Fine KNN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
31-CLASSES (BINARY)	ARITH.DIVIDER_ACTIVE	BACLEARS.ANY	BR_INST_RETIRED.ALL_BRANCHES_PS	BR_INST_RETIRED.CONDITIONAL_PS	BR_INST_RETIRED.COND_NTAKEN	BR_INST_RETIRED.FAR_BRANCH_PS	BR_INST_RETIRED.NEAR_CALL_PS	BR_INST_RETIRED.NEAR_RETURN_PS	BR_INST_RETIRED.NEAR_TAKEN_PS	BR_INST_RETIRED.NOT_TAKEN	BR_MISP_RETIRED.ALL_BRANCHES_PS	BR_MISP_RETIRED.CONDITIONAL_PS	BR_MISP_RETIRED.NEAR_CALL_PS	BR_MISP_RETIRED.NEAR_TAKEN_PS	CPU_CLK_THREAD_UNHALTED.ONE_THREAD_ACTIV	CPU_CLK_THREAD_UNHALTED.REF_XCLK	CPU_CLK_THREAD_UNHALTED.REF_XCLK_ANY	CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE	CPU_CLK_UNHALTED.REF_TSC	CPU_CLK_UNHALTED.REF_XCLK	CPU_CLK_UNHALTED.REF_XCLK_ANY	CPU_CLK_UNHALTED.RING0_TRANS	CPU_CLK_UNHALTED.THREAD_ANY	CPU_CLK_UNHALTED.THREAD_P
F1																								
9.9-0.1-10:	0.6775	0.6119	0.5964	0.6248	0.6279	0.6564	0.6200	0.6647	0.5894	0.6492	0.5535	0.5394	0.5364	0.5362	0.5583	0.6292	0.6121	0.5429	0.7082	0.6321	0.5951	0.6307	0.6521	0.6168
10-0.1-10.1:	0.6900	0.5843	0.6477	0.6942	0.5826	0.6828	0.6339	0.6134	0.6846	0.6379	0.4952	0.5192	0.5895	0.5942	0.5055	0.5640	0.5499	0.4808	0.6223	0.5714	0.5748	0.5898	0.6208	0.6292
10.1-0.1-10.2:	0.6689	0.5659	0.6562	0.6882	0.6647	0.6505	0.6893	0.6814	0.6666	0.6449	0.5642	0.5799	0.5955	0.6607	0.5207	0.6520	0.6364	0.5390	0.6382	0.6106	0.6264	0.6488	0.6843	0.6632
9.5-0.5-10:	0.7666	0.6856	0.6632	0.6905	0.6548	0.6913	0.7383	0.7508	0.7590	0.6929	0.5924	0.6208	0.6134	0.6406	0.7177	0.5930	0.5955	0.5811	0.6674	0.6440	0.5912	0.6213	0.6420	0.6662
10-0.5-10.5:	0.7152	0.7066	0.7141	0.7456	0.6422	0.7231	0.7657	0.7223	0.6548	0.6534	0.5642	0.5144	0.6146	0.6206	0.6259	0.6619	0.5400	0.5423	0.6367	0.6082	0.5652	0.7241	0.6632	0.6548
10.5-0.5-11:	0.7329	0.7029	0.6988	0.7475	0.6449	0.7177	0.8083	0.7736	0.6772	0.6562	0.6119	0.5465	0.6394	0.5702	0.7521	0.6590	0.6278	0.7182	0.6843	0.7489	0.6520	0.7515	0.6720	0.6750
9-1-10:	0.7537	0.6479	0.7316	0.6535	0.7323	0.6935	0.7182	0.7427	0.6814	0.7241	0.7190	0.5031	0.5836	0.6479	0.6636	0.6814	0.5379	0.6929	0.7022	0.6871	0.5551	0.6967	0.6292	0.6109
10-1-11:	0.7605	0.7210	0.7824	0.6321	0.6649	0.7755	0.7816	0.8094	0.7434	0.6717	0.5935	0.6050	0.6730	0.6632	0.6792	0.6562	0.6351	0.6988	0.6746	0.6578	0.6425	0.7988	0.7211	0.7192
11-1-12:	0.6887	0.7434	0.7037	0.6654	0.6492	0.7305	0.7485	0.7494	0.7300	0.5834	0.6437	0.6939	0.6177	0.6137	0.7537	0.6995	0.6505	0.7363	0.7024	0.7231	0.6579	0.7383	0.7100	0.7223
8-2-10:	0.7665	0.6875	0.7259	0.6720	0.7047	0.8187	0.7856	0.7515	0.7607	0.6967	0.6307	0.6678	0.5959	0.6564	0.6887	0.7070	0.6125	0.7235	0.7082	0.7354	0.6114	0.7149	0.6675	0.6863
10-2-12:	0.7847	0.7261	0.7342	0.7415	0.7108	0.8293	0.8591	0.8597	0.8324	0.7546	0.6717	0.6619	0.6856	0.6435	0.7159	0.7363	0.6817	0.7293	0.7329	0.7377	0.6491	0.7935	0.6958	0.7445
12-2-14:	0.7067	0.7534	0.7312	0.7120	0.7119	0.7383	0.7975	0.7626	0.7697	0.6534	0.7393	0.6954	0.6350	0.6407	0.7053	0.7363	0.6858	0.7526	0.7231	0.7577	0.6856	0.7264	0.7293	0.7475
MEAN	0.7260	0.6781	0.6988	0.6889	0.6659	0.7256	0.7455	0.7401	0.7124	0.6682	0.6149	0.5956	0.6150	0.6240	0.6572	0.6647	0.6138	0.6448	0.6834	0.6762	0.6172	0.7029	0.6740	0.6780

Figure 7.32: F1 scores for each HPC feature and start/stop time window were calculated and then averaged across all start/stop time windows to obtain an average F1 score for HPC feature ranking purposes.

Features were then sorted from highest to lowest F1 score for both the 23- and 31-class data sets for each of the 5 classifiers. Each classifier was then re-trained by incrementing the number of HPCs by 1 each time, and plots of Cross Validation Accuracy, Precision, Recall, and F1 for all sets of features for both data sets were generated for each classifier. Plots of cross validation statistics (Accuracy, Precision, Recall, and F1) for each window size at operation start time for both data sets are shown for each classifier in Figures 7.33-7.37.

The top 20 ranked features from highest to lowest F1 score for both the 23- and 31-class data sets for each of the 5 classifiers are shown in the Figure 7.38. 80 individual HPCs generated F1 scores of 0.7 or better alone, and 40 of those occurred in the top 20 of any particular classifier (highlighted yellow). Since only 2 distinct HPCs from the 23-class data set generated F1 scores of 0.7 or better (6: `CPU_CLK_UNHALTED.REF_TSC` and 8: `CPU_CLK_UNHALTED.THREAD_ANY`), and both correlate to HPCs collected in the 31-class data set (19 and 23, respectively), the 23-class (Study 1) data set was dropped from consideration and all further analysis was conducted on the 220 HPCs collected in the 31-class data set.

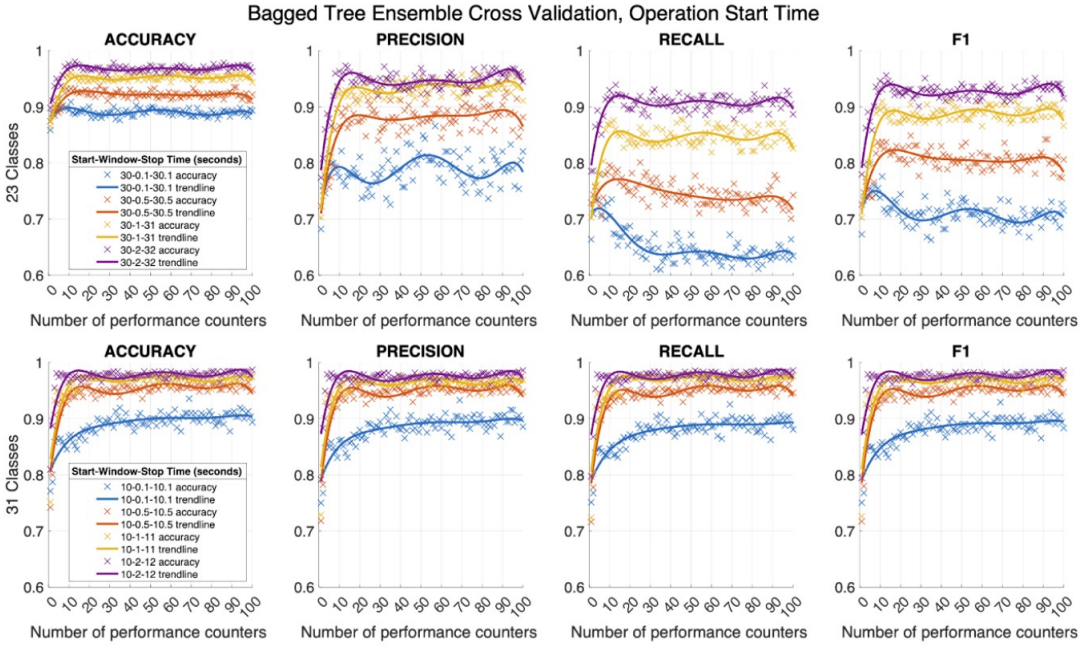


Figure 7.33: Cross validation accuracy, precision, recall, and F1 plots for binary 23-class (Study 1) and 31-class (Study 2) data sets trained with Bagged Tree Ensemble classifier and incrementing the number of HPCs used for training. HPCs were ranked from high to low by F1 score averaged over start/stop time window sizes.

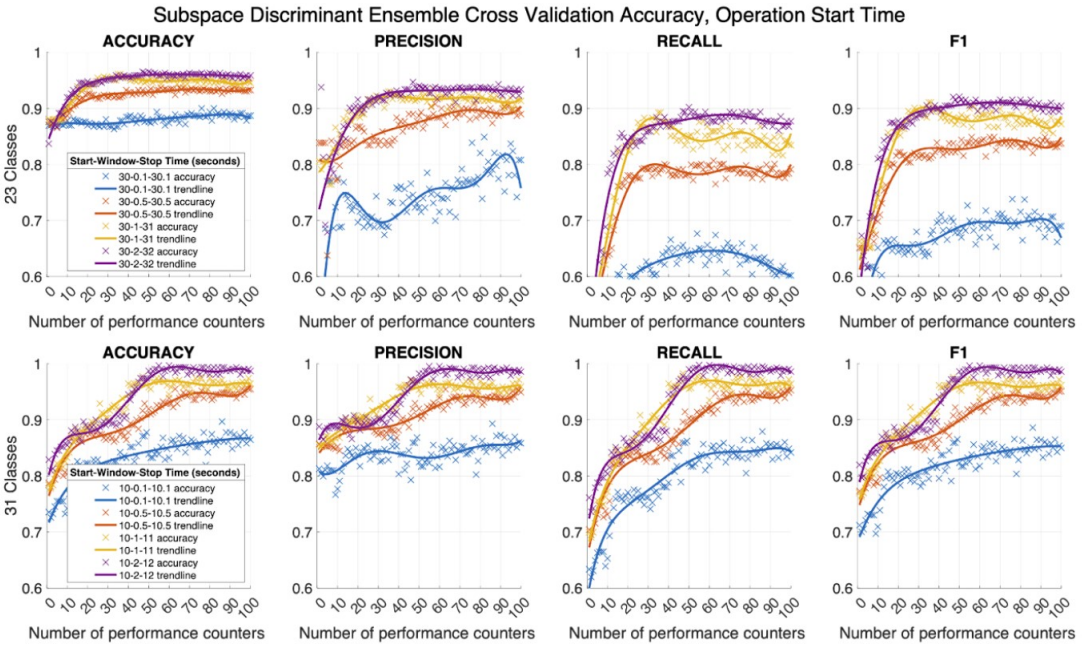


Figure 7.34: Cross validation accuracy, precision, recall, and F1 plots for binary 23-class (Study 1) and 31-class (Study 2) data sets trained with Subspace Discriminant Ensemble classifier and incrementing the number of HPCs used for training. HPCs were ranked from high to low by F1 score averaged over start/stop time window sizes.

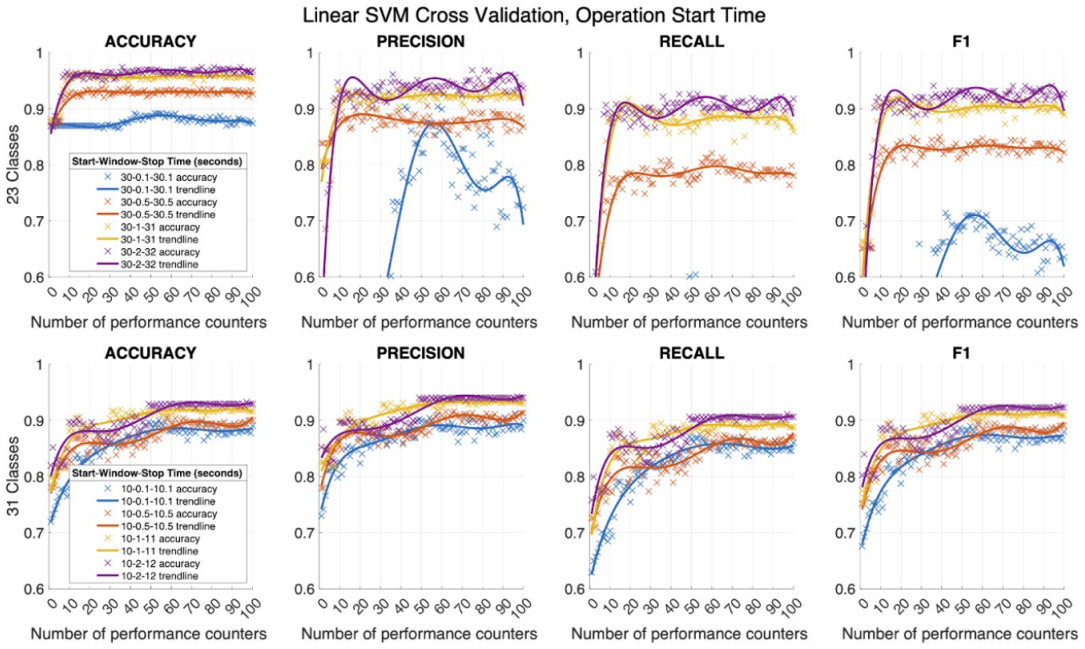


Figure 7.35: Cross validation accuracy, precision, recall, and F1 plots for binary 23-class (Study 1) and 31-class (Study 2) data sets trained with Linear Support Vector Machine classifier and incrementing the number of HPCs used for training. HPCs were ranked from high to low by F1 score averaged over start/stop time window sizes.

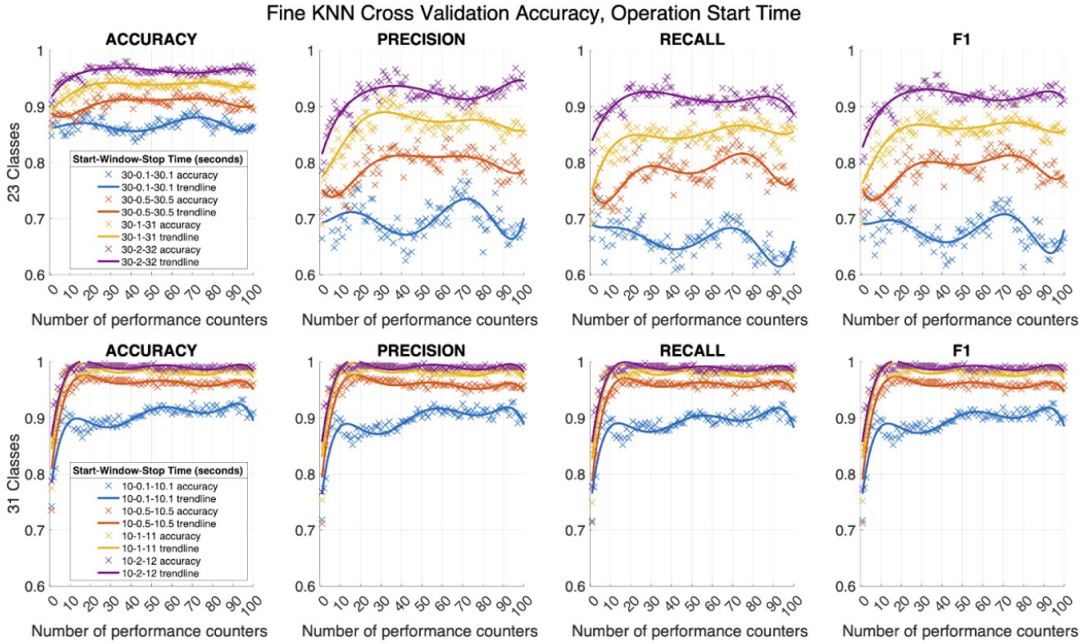


Figure 7.36: Cross validation accuracy, precision, recall, and F1 plots for binary 23-class (Study 1) and 31-class (Study 2) data sets trained with Fine K-Nearest Neighbor classifier and incrementing the number of HPCs used for training. HPCs were ranked from high to low by F1 score averaged over start/stop time window sizes.

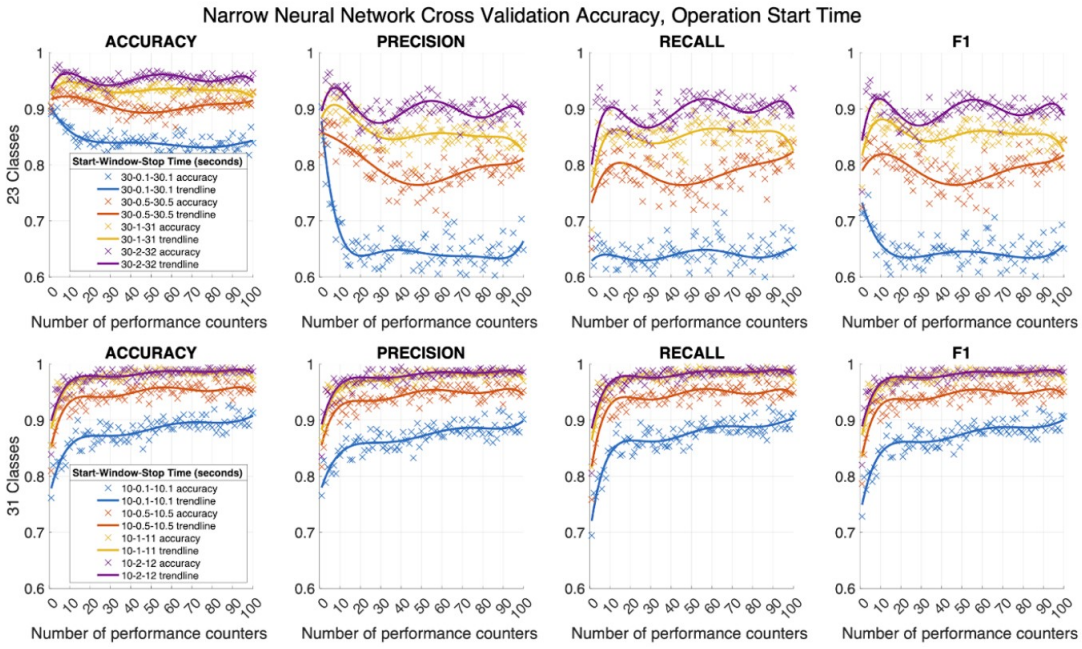


Figure 7.37: Cross validation accuracy, precision, recall, and F1 plots for binary 23-class (Study 1) and 31-class (Study 2) data sets trained with Narrow Neural Network classifier and incrementing the number of HPCs used for training. HPCs were ranked from high to low by F1 score averaged over start/stop time window sizes.

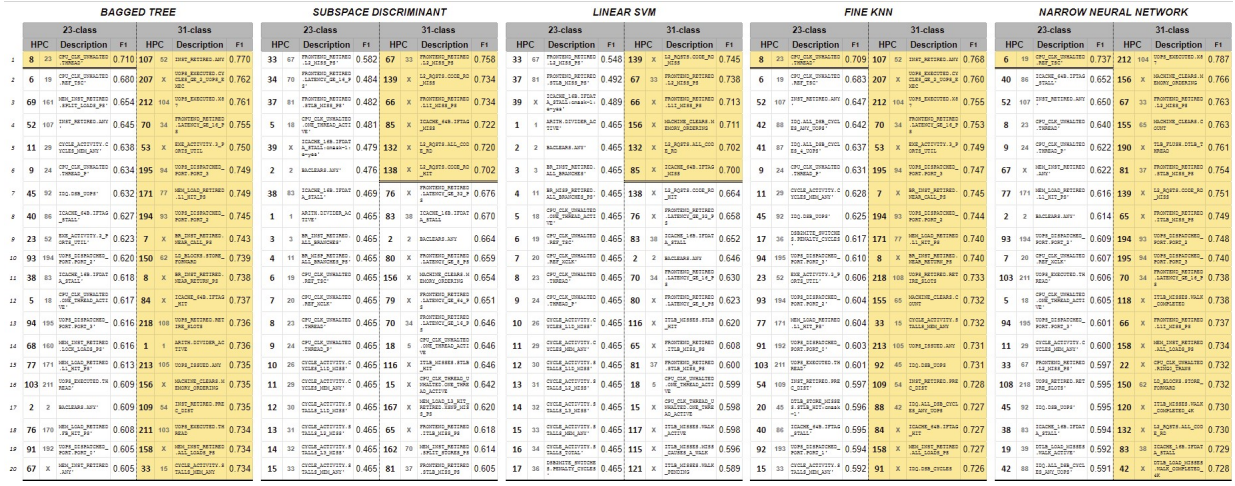


Figure 7.38: Top 20 F1-ranked HPCs for each classification algorithm. Study 1 (23-class) HPC features are on the left column and Study 2 (31-class) HPC features are on the right. Corresponding feature numbers from the opposing study are listed in gray next to the HPC number for each study.

7.5.4 Single-HPC F1-based Feature Reduction

In order to reduce the feature set for another round of data collection with additional ransomware samples, additional analysis of the Study 2 (31-class) data set was performed to determine the highest attainable F1 using combinations of features. This involved the following steps:

1. Fix the top performing HPC for each classifier based on F1 rank (using average F1-score across all 12 start/window/stop time feature sets). Based on the results from the previous section, these HPCs are shown on the right-hand side in the top row of Figure 7.38, and reproduced in Figure 7.39 for the Study 2 (31-class) data set only.

2. For each of the 5 classifiers, iterate through all (220 minus the number of previously selected, "fixed" HPCs) remaining features, calculating accuracy, precision, recall, F1 and generating the confusion matrix for each classifier/fixed HPC combination.

3. Average F1 scores across each start/window/stop time feature set.

4. Rank remaining (220 minus the number of previously selected, "fixed" HPCs) features from highest to lowest combined F1.

5. Repeat steps, fixing the HPC determined in step 4 in addition to any HPC(s) already fixed from Step 1.

After 13 iterations, the top HPCs based on averaged-F1 ranked order are shown in Figure 7.40. The light yellow shading indicates any of the 80 HPCs that had individual F1-scores over 0.7, while those shaded in dark yellow were in the top 20 of any respective classification

	BAGGED TREE			SUBSPACE DISCRIMINANT			LINEAR SVM			FINE KNN			NARROW NEURAL NETWORK		
	HPC	Description	Total F1	HPC	Description	Total F1	HPC	Description	Total F1	HPC	Description	Total F1	HPC	Description	Total F1
1	107	INST_RETIRED.ANY	0.7700	67	FRONTEND_RETIRED.L2_MISS_PS	0.7581	139	L2_RQSTS.CODE_RD_MISS	0.7446	107	INST_RETIRED.ANY	0.7684	212	UOPS_EXECUTED.X87	0.7873

Figure 7.39: Top F1-ranked HPC for each classification algorithm for Study 2 (31-class data set) only. These were used as a starting point to identify the fewest number of HPC features to generate the highest possible F1 score.

algorithm.

	BAGGED TREE			SUBSPACE DISCRIMINANT			LINEAR SVM			FINE KNN			NARROW NEURAL NETWORK		
	HPC	Description	Total F1	HPC	Description	Total F1	HPC	Description	Total F1	HPC	Description	Total F1	HPC	Description	Total F1
1	107	INST_RETIRED.ANY	0.7700	67	FRONTEND_RETIRED.L2_MISS_PS	0.7581	139	L2_RQSTS.CODE_RD_MISS	0.7446	107	INST_RETIRED.ANY	0.7684	212	UOPS_EXECUTED.X87	0.7873
2	118	ITLB_MISSES.WALK_COMPLETED	0.9064	66	FRONTEND_RETIRED.L1I_MISS_PS	0.7579	12	BR_MISP_RETIRED.CONDITIONAL_PS	0.8224	97	IDQ.MS_MITE_UOPS	0.9122	156	MACHINE_CLEARS.MEMORY_ORDERING	0.8463
3	156	MACHINE_CLEARS.MEMORY_ORDERING	0.9306	12	BR_MISP_RETIRED.CONDITIONAL_PS	0.7952	156	MACHINE_CLEARS.MEMORY_ORDERING	0.8722	14	BR_MISP_RETIRED.NEAR_TAKEN_PS	0.9457	36	DSB2MITE_SWITCHES.PENALTY_CYCLES	0.9183
4	173	MEM_LOAD_RETIRED.L2_HIT_PS	0.9403	162	MEM_INST_RETIRED.SPLIT_STORES_PS	0.8172	127	L2_LINES_IN.ALL	0.9228	46	DTLB_STORE_MISSES.WALK_ACTIVE	0.9545	81	FRONTEND_RETIRED.STLB_MISS_PS	0.9360
5	61	FP_ARITH_INST_RETIRED.SCALAR_DOUBLE	0.9462	124	L1D_PEND_MISS.PENDING	0.8273	61	FP_ARITH_INST_RETIRED.SCALAR_DOUBLE	0.9339	89	IDQ.ALL_MITE_CYCLES_4_UOPS	0.9567	162	MEM_INST_RETIRED.SPLIT_STORES_PS	0.9406
6	112	INT_MISC.RECOVERY_CYCLES	0.9480	85	ICACHE_64B.IFTAG_MISS	0.8294	60	FP_ARITH_INST_RETIRED.256B_PACKED_SINGLE	0.9378	171	MEM_LOAD_RETIRED.L1_HIT_PS	0.9590	26	CYCLE_ACTIVITY.CYCLES_L1D_MISS	0.9445
7	212	UOPS_EXECUTED.X87	0.9509	148	L2_TRANS.L2_WB	0.8306	213	UOPS_ISSUED.ANY	0.9382	214	OPS_ISSUED.SLOW_LEA	0.9612	6	BR_INST_RETIRED.FAR_BRANCH_PS	0.9462
8	71	FRONTEND_RETIRED.LATENCY_GE_2	0.9516	57	FP_ARITH_INST_RETIRED.128B_PACKED_DOUBLE	0.8365	73	FRONTEND_RETIRED.LATENCY_GE_2_BUBBLES_GE_1_PS	0.9432	40	DTLB_LOAD_MISSES.WALK_COMPLETED	0.9636	89	IDQ.ALL_MITE_CYCLES_4_UOPS	0.9534
9	142	L2_RQSTS.MISS	0.9537	156	MACHINE_CLEARS.MEMORY_ORDERING	0.8408	122	L1D.REPLACEMENT	0.9456	147	L2_RQSTS.RFO_MISS	0.9670	90	IDQ.ALL_MITE_CYCLES_ANY_UOPS	0.9542
10	153	LONGEST_LAT_CACHE.MISS	0.9546	84	ICACHE_64B.IFTAG_HIT	0.8694	117	ITLB_MISSES.WALK_ACTIVE	0.9477	145	L2_RQSTS.REFERENCES	0.9662	192	UOPS_DISPATCHED_PORT.PORT_0	0.9558
11	174	MEM_LOAD_RETIRED.L2_MISS_PS	0.9560	61	FP_ARITH_INST_RETIRED.SCALAR_DOUBLE	0.8730	41	DTLB_LOAD_MISSES.WALK_COMPLETED_2M_4M	0.9501	6	BR_INST_RETIRED.FAR_BRANCH_PS	0.9663	143	L2_RQSTS.PF_HIT	0.9573
12	7	BR_INST_RETIRED.NEAR_CALL_PS	0.9542	157	MACHINE_CLEARS.SMC	0.8968	198	UOPS_DISPATCHED_PORT.PORT_6	0.9517	31	CYCLE_ACTIVITY.STALLS_L2_MISS	0.9682	209	UOPS_EXECUTED.CYCLES_GE_4_UOPS_EXEC	0.9586
13	117	ITLB_MISSES.WALK_ACTIVE	0.9540	104	IDQ_UOPS_NOT_DELIVERED.CYCLES_LE_2_UOP_DELIV.CORE	0.9187	128	L2_LINES_OUT.NON_SILENT	0.9559	128	L2_LINES_OUT.NON_SILENT	0.9700	59	FP_ARITH_INST_RETIRED.256B_PACKED_DOUBLE	0.9598

Figure 7.40: HPC Features selected for each classifier to achieve the highest possible F1 score with the fewest number of features, ranked by F1 score.

Figure 7.41 shows the cross validation F1 score for the cumulative number of trained features in the ranked order shown in Figure 7.40 for each classifier for the 0.1s, 0.5 s, 1s, and 2s windows at Start Time - Window, at Start Time, and at Start Time + Window.

7.5.4.1 F1-ranked HPC Feature Repeatability

Since the averaged F1 scores for HPC combinations were within a thousandth of each other in some cases, the top 3 F1 scores for each algorithm were also considered, as shown in Figure 7.42. The light yellow shading still indicates the 80 HPCs that had individual F1-scores over 0.7, while those dark yellow were in the top 20 of any respective classification algorithm.

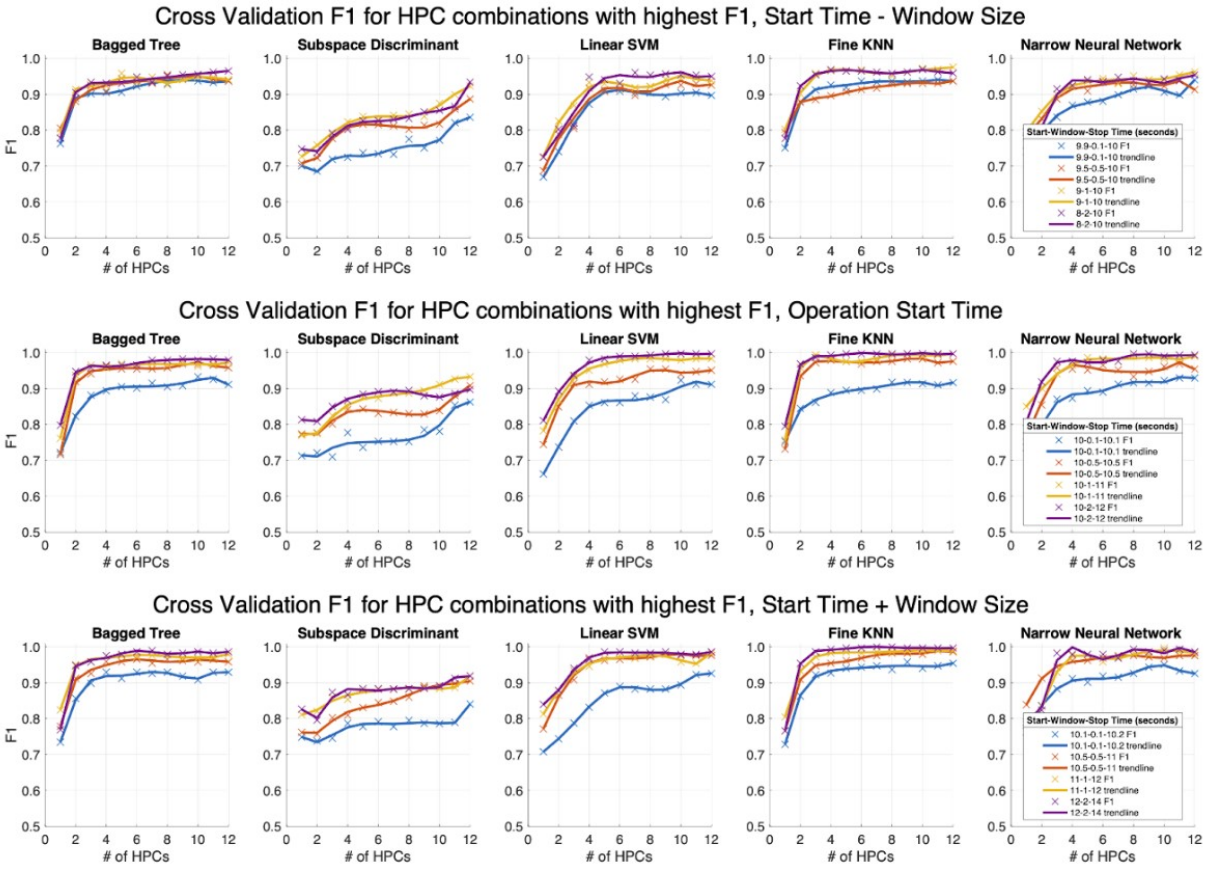


Figure 7.41: Cross Validation F1 for each classifier, with HPC features selected to achieve the highest possible F1 score with the fewest number of features. The HPC Features selected for training each classifier are shown in Figure 7.40.

	BAGGED TREE			SUBSPACE DISCRIMINANT			LINEAR SVM			FINE KNN			NARROW NEURAL NETWORK		
	HPC	Description	Total F1	HPC	Description	Total F1	HPC	Description	Total F1	HPC	Description	Total F1	HPC	Description	Total F1
1	107	INST_RETIRED.ANY	0.7700	67	FRONTEND_RETIRED.L2_MISS_PS	0.7581	139	L2_RQSTS.CODE_RD_MISS	0.7446	107	INST_RETIRED.ANY	0.7684	212	UOPS_EXECUTED.X87	0.7873
	207	UOPS_EXECUTED.CYCLES_GE_2_UOPS_EXE	0.7623	139	L2_RQSTS.CODE_RD_MISS	0.7343	67	FRONTEND_RETIRED.L2_MISS_PS	0.7378	207	UOPS_EXECUTED.CYCLES_GE_2_UOPS_EXEC	0.7603	156	MACHINE_CLEARS.MEMORY_ORDERING	0.7661
	212	UOPS_EXECUTED.X87	0.7614	66	FRONTEND_RETIRED.L1I_MISS_PS	0.7341	66	FRONTEND_RETIRED.L1I_MISS_PS	0.7133	212	UOPS_EXECUTED.X87	0.7554	67	FRONTEND_RETIRED.L2_MISS_PS	0.7632
2	118	ITLB_MISSES.WALK_COMPLETED	0.9064	66	FRONTEND_RETIRED.L1I_MISS_PS	0.7579	12	BR_MISP_RETIRED.CONDITIONAL_PS	0.8224	97	IDQ.MS_MITE_UOPS	0.9122	156	MACHINE_CLEARS.MEMORY_ORDERING	0.8463
	115	ITLB_MISSES.MISS_CAUSES_A_WALK	0.9043	139	L2_RQSTS.CODE_RD_MISS	0.7494	11	BR_MISP_RETIRED.ALL_BRANCHES_PS	0.8148	99	IDQ.MS_UOPS	0.9113	158	MEM_INST_RETIRED.ALL_LOADS_PS	0.8376
	97	IDQ.MS_MITE_UOPS	0.9024	70	FRONTEND_RETIRED.LATENCY_GE_16_PS	0.7462	14	BR_MISP_RETIRED.NEAR_TAKEN_PS	0.8062	6	BR_INST_RETIRED.FAR_BRANCH_PS	0.9094	171	MEM_LOAD_RETIRED.L1_HIT_PS	0.8314
3	156	MACHINE_CLEARS.MEMORY_ORDERING	0.9306	12	BR_MISP_RETIRED.CONDITIONAL_PS	0.7952	156	MACHINE_CLEARS.MEMORY_ORDERING	0.8722	14	BR_MISP_RETIRED.NEAR_TAKEN_PS	0.9457	36	DSB2MITE_SWITCHES.PENALTY_CYCLES	0.9183
	84	ICACHE_64B.IFTAG_HIT	0.9294	11	BR_MISP_RETIRED.ALL_BRANCHES_PS	0.7897	143	L2_RQSTS.PF_HIT	0.8461	11	BR_MISP_RETIRED.ALL_BRANCHES_PS	0.9456	35	DSB2MITE_SWITCHES.COUNT	0.9094
	136	L2_RQSTS.ALL_PF	0.9289	112	INT_MISC.RECOVERY_CYCLES	0.7895	122	L1D.REPLACEMENT	0.8421	12	BR_MISP_RETIRED.CONDITIONAL_PS	0.9455	84	ICACHE_64B.IFTAG_HIT	0.9057
4	173	MEM_LOAD_RETIRED.L2_HIT_PS	0.9403	162	MEM_INST_RETIRED.SPLIT_STORES_PS	0.8172	127	L2_LINES_IN.ALL	0.9228	46	DTLB_STORE_MISSES.WALK_ACTIVE	0.9545	81	FRONTEND_RETIRED.STLB_MISS_PS	0.9360
	140	L2_RQSTS.DEMAND_DATA_RD_HIT	0.9398	161	MEM_INST_RETIRED.SPLIT_LOADS_PS	0.8158	136	L2_RQSTS.ALL_PF	0.9208	144	L2_RQSTS.PF_MISS	0.9539	70	FRONTEND_RETIRED.LATENCY_GE_16_PS	0.9345
	143	L2_RQSTS.PF_HIT	0.9390	199	UOPS_DISPATCHED_PORT.PORT_7	0.8123	154	LONGEST_LAT_CACHE.REFERENCE	0.9190	50	DTLB_STORE_MISSES.WALK_PENDING	0.9538	121	ITLB_MISSES.WALK_PENDING	0.9342
5	61	FP_ARITH_INST_RETIRED.SCALAR_DOUBL	0.9462	124	L1D_PEND_MISS.PENDING	0.8273	61	FP_ARITH_INST_RETIRED.SCALAR_DOUBLE	0.9339	89	IDQ.ALL_MITE_CYCLES_4_UOPS	0.9567	162	MEM_INST_RETIRED.SPLIT_STORES_PS	0.9406
	84	ICACHE_64B.IFTAG_HIT	0.9445	123	L1D_PEND_MISS.FB_FULL	0.8265	81	FRONTEND_RETIRED.STLB_MISS_PS	0.9302	40	DTLB_LOAD_MISSES.WALK_COMPLETED	0.9564	61	FP_ARITH_INST_RETIRED.SCALAR_DOUBLE	0.9401
	102	IDQ_UOPS_NOT_DELIVERED.CYCLES_FE_W	0.9438	47	DTLB_STORE_MISSES.WALK_COMPLETED	0.8255	18	CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE	0.9291	128	L2_LINES_OUT.NON_SILENT	0.9561	128	L2_LINES_OUT.NON_SILENT	0.9397

Figure 7.42: Top 3 Cross Validation F1 scores for each classifier and F1 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features.

The F1-ranked Feature Reduction steps (Section 7.5.4) were then repeated for the 1st 4 counters from Figure 7.40 to check the repeatability of the F1 rank results. The results for the five classifiers (Bagged Tree, Subspace Discriminant, Linear SVM, Fine KNN, Narrow Neural Network) are shown in Figures 7.43-7.47, respectively. In each figure, the top 3 counters identified in the 1st run are in the left column for each classifier, while the top 3 counters of the 2nd run are in the right column for each classifier. For 4 of 5 classifiers (all but Linear SVM), the top F1-ranked HPC is identical in both runs. For the Linear SVM classifier (Figure 7.45), the 1 and 2 ranks switched from run 1 to run 2, but the top 3 ranked HPCs for the 1st fixed feature were the same across runs. Additionally, for 4 of 5 classifiers (all but Narrow Neural Network), the top 3 ranked HPCs were the same for the 1st fixed feature. For the Narrow Neural Network (Figure 7.47, the 3rd and 4th ranked HPCs swapped between runs 1 and 2, so the top 4 ranked HPCs for the 1st fixed feature were the same across runs.

For the 2nd fixed feature, the top-ranked combined F1 score was the same for 4 of 5

BAGGED TREE

	1st Run			2nd Run		
	HPC	Description	Total F1	HPC	Description	Total F1
1	107	INST_RETIRED.ANY	0.7700	107	INST_RETIRED.ANY	0.7660
	207	UOPS_EXECUTED.CYCLES_GE_2_UOPS_EXEC	0.7623	207	UOPS_EXECUTED.CYCLES_GE_2_UOPS_EXEC	0.7610
	212	UOPS_EXECUTED.X87	0.7614	212	UOPS_EXECUTED.X87	0.7590
2	118	ITLB_MISSES.WALK_COMPLETED	0.9064	118	ITLB_MISSES.WALK_COMPLETED	0.9076
	115	ITLB_MISSES.MISS_CAUSES_A_WALK	0.9043	66	FRONTEND_RETIRED.L1I_MISS_PS	0.9053
	97	IDQ.MS_MITE_UOPS	0.9024	120	ITLB_MISSES.WALK_COMPLETED_4K	0.9052
3	156	MACHINE_CLEARS.MEMORY_ORDERING	0.9306	143	L2_RQSTS.PF_HIT	0.9351
	84	ICACHE_64B.IFTAG_HIT	0.9294	4	BR_INST_RETIRED.CONDITIONAL_PS	0.9312
	136	L2_RQSTS.ALL_PF	0.9289	84	ICACHE_64B.IFTAG_HIT	0.9306
4	173	MEM_LOAD_RETIRED.L2_HIT_PS	0.9403	84	ICACHE_64B.IFTAG_HIT	0.9420
	140	L2_RQSTS.DEMAND_DATA_RD_HIT	0.9398	61	FP_ARITH_INST_RETIRED.SCALAR_DOUBLE	0.9420
	143	L2_RQSTS.PF_HIT	0.9390	122	L1D.REPLACEMENT	0.9400

Figure 7.43: Comparison of HPC features with three highest F1 cross validation scores for 1st 4 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features using Bagged Tree Ensemble Classifier.

SUBSPACE DISCRIMINANT

	1st Run			2nd Run		
	HPC	Description	Total F1	HPC	Description	Total F1
1	67	FRONTEND_RETIRED.L2_MISS_PS	0.7581	67	FRONTEND_RETIRED.L2_MISS_PS	0.7550
	139	L2_RQSTS.CODE_RD_MISS	0.7343	66	FRONTEND_RETIRED.L1I_MISS_PS	0.7360
	66	FRONTEND_RETIRED.L1I_MISS_PS	0.7341	139	L2_RQSTS.CODE_RD_MISS	0.7340
2	66	FRONTEND_RETIRED.L1I_MISS_PS	0.7579	66	FRONTEND_RETIRED.L1I_MISS_PS	0.7565
	139	L2_RQSTS.CODE_RD_MISS	0.7494	139	L2_RQSTS.CODE_RD_MISS	0.7479
	70	FRONTEND_RETIRED.LATENCY_GE_16_PS	0.7462	70	FRONTEND_RETIRED.LATENCY_GE_16_PS	0.7444
3	12	BR_MISP_RETIRED.CONDITIONAL_PS	0.7952	12	BR_MISP_RETIRED.CONDITIONAL_PS	0.7981
	11	BR_MISP_RETIRED.ALL_BRANCHES_PS	0.7897	11	BR_MISP_RETIRED.ALL_BRANCHES_PS	0.7902
	112	INT_MISC.RECOVERY_CYCLES	0.7895	112	INT_MISC.RECOVERY_CYCLES	0.7889
4	162	MEM_INST_RETIRED.SPLIT_STORES_PS	0.8172	162	MEM_INST_RETIRED.SPLIT_STORES_PS	0.8200
	161	MEM_INST_RETIRED.SPLIT_LOADS_PS	0.8158	161	MEM_INST_RETIRED.SPLIT_LOADS_PS	0.8150
	199	UOPS_DISPATCHED_PORT.PORT_7	0.8123	199	UOPS_DISPATCHED_PORT.PORT_7	0.8110

Figure 7.44: Comparison of HPC features with three highest F1 cross validation scores for 1st 4 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features using Subspace Discriminant Ensemble Classifier.

LINEAR SVM

	1st Run			2nd Run		
	HPC	Description	Total F1	HPC	Description	Total F1
1	139	L2_RQSTS.CODE_RD_MISS	0.7446	67	FRONTEND_RETIRED.L2_MISS_PS	0.7520
	67	FRONTEND_RETIRED.L2_MISS_PS	0.7378	139	L2_RQSTS.CODE_RD_MISS	0.7360
	66	FRONTEND_RETIRED.L1I_MISS_PS	0.7133	66	FRONTEND_RETIRED.L1I_MISS_PS	0.7050
2	12	BR_MISP_RETIRED.CONDITIONAL_PS	0.8224	12	BR_MISP_RETIRED.CONDITIONAL_PS	0.8218
	11	BR_MISP_RETIRED.ALL_BRANCHES_PS	0.8148	11	BR_MISP_RETIRED.ALL_BRANCHES_PS	0.8130
	14	BR_MISP_RETIRED.NEAR_TAKEN_PS	0.8062	14	BR_MISP_RETIRED.NEAR_TAKEN_PS	0.8048
3	156	MACHINE_CLEARS.MEMORY_ORDERING	0.8722	156	MACHINE_CLEARS.MEMORY_ORDERING	0.8722
	143	L2_RQSTS.PF_HIT	0.8461	143	L2_RQSTS.PF_HIT	0.8438
	122	L1D.REPLACEMENT	0.8421	168	MEM_LOAD_L3_HIT_RETIRED.XSNP_NONE_PS	0.8426
4	127	L2_LINES_IN.ALL	0.9228	127	L2_LINES_IN.ALL	0.9220
	136	L2_RQSTS.ALL_PF	0.9208	136	L2_RQSTS.ALL_PF	0.9210
	154	LONGEST_LAT_CACHE.REFERENCE	0.9190	142	L2_RQSTS.MISS	0.9210

Figure 7.45: Comparison of HPC features with three highest F1 cross validation scores for 1st 4 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features using Linear Support Vector Machine Classifier.

FINE KNN

	1st Run			2nd Run		
	HPC	Description	Total F1	HPC	Description	Total F1
1	107	INST_RETIRED.ANY	0.7684	107	INST_RETIRED.ANY	0.7670
	207	UOPS_EXECUTED.CYCLES_GE_2_UOPS_EXEC	0.7603	207	UOPS_EXECUTED.CYCLES_GE_2_UOPS_EXEC	0.7600
	212	UOPS_EXECUTED.X87	0.7554	212	UOPS_EXECUTED.X87	0.7560
2	97	IDQ.MS_MITE_UOPS	0.9122	99	IDQ.MS_UOPS	0.9110
	99	IDQ.MS_UOPS	0.9113	95	IDQ.MS_CYCLES	0.9095
	6	BR_INST_RETIRED.FAR_BRANCH_PS	0.9094	97	IDQ.MS_MITE_UOPS	0.9087
3	14	BR_MISP_RETIRED.NEAR_TAKEN_PS	0.9457	112	INT_MISC.RECOVERY_CYCLES	0.9456
	11	BR_MISP_RETIRED.ALL_BRANCHES_PS	0.9456	12	BR_MISP_RETIRED.CONDITIONAL_PS	0.9441
	12	BR_MISP_RETIRED.CONDITIONAL_PS	0.9455	14	BR_MISP_RETIRED.NEAR_TAKEN_PS	0.9438
4	46	DTLB_STORE_MISSES.WALK_ACTIVE	0.9545	128	L2_LINES_OUT.NON_SILENT	0.9550
	144	L2_RQSTS.PF_MISS	0.9539	127	L2_LINES_IN.ALL	0.9540
	50	DTLB_STORE_MISSES.WALK_PENDING	0.9538	40	DTLB_LOAD_MISSES.WALK_COMPLETED	0.9540

Figure 7.46: Comparison of HPC features with three highest F1 cross validation scores for 1st 4 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features using Fine K-Nearest Neighbor Classifier.

NARROW NEURAL NETWORK

	1st Run			2nd Run		
	HPC	Description	Total F1	HPC	Description	Total F1
1	212	UOPS_EXECUTED.X87	0.7873	212	UOPS_EXECUTED.X87	0.7920
	156	MACHINE_CLEARS.MEMORY_ORDERING	0.7661	156	MACHINE_CLEARS.MEMORY_ORDERING	0.7700
	67	FRONTEND_RETIRED.L2_MISS_PS	0.7632	155	MACHINE_CLEARS.COUNT	0.7670
2	156	MACHINE_CLEARS.MEMORY_ORDERING	0.8463	156	MACHINE_CLEARS.MEMORY_ORDERING	0.8442
	158	MEM_INST_RETIRED.ALL_LOADS_PS	0.8376	51	EXE_ACTIVITY.1_PORTS_UTIL	0.8387
	171	MEM_LOAD_RETIRED.L1_HIT_PS	0.8314	194	UOPS_DISPATCHED_PORT.PORT_2	0.8315
3	36	DSB2MITE_SWITCHES.PENALTY_CYCLES	0.9183	84	ICACHE_64B.IFTAG_HIT	0.9157
	35	DSB2MITE_SWITCHES.COUNT	0.9094	36	DSB2MITE_SWITCHES.PENALTY_CYCLES	0.9144
	84	ICACHE_64B.IFTAG_HIT	0.9057	35	DSB2MITE_SWITCHES.COUNT	0.9064
4	81	FRONTEND_RETIRED.STLB_MISS_PS	0.9360	43	DTLB_LOAD_MISSES.WALK_PENDING	0.9380
	70	FRONTEND_RETIRED.LATENCY_GE_16_PS	0.9345	40	DTLB_LOAD_MISSES.WALK_COMPLETED	0.9360
	121	ITLB_MISSES.WALK_PENDING	0.9342	184	RESOURCE_STALLS.ANY	0.9340

Figure 7.47: Comparison of HPC features with three highest F1 cross validation scores for 1st 4 ranks, with HPC features selected to achieve the highest possible F1 score with the fewest number of features using Narrow Neural Network Classifier.

classifiers (all but Fine KNN), and for Fine KNN the top-ranked 2nd HPC of the 2nd run was the 2nd-ranked HPC of the 1st run. The results for the 3rd fixed HPC were less consistent, with only 2 of 5 (Subspace Discriminant and Linear SVM) classifiers yielding the same top-ranked HPC between runs.

Since the top HPC was not repeatable at each rank but the top several HPCs at each rank were similar, the set of HPC features to be collected was reduced from 220 to 90 by collecting only HPCs that were ranked in the top 3 features for each of the the 1st 10 iterations of fixed features. The resulting 90 HPCs selected as features, along with their descriptions, are included in Appendix A; these were used for Study 3 (Chapter 8).

7.6 Summary

The experiments in this chapter resulted in a streamlined ransomware data collection and safe transfer process on compromised hardware, as well as the identification of five classification

algorithms and 90 performance counters for additional evaluation. Additionally, VTune standard micro-architectural collection and several feature ranking algorithms were ruled out of further consideration, in favor of performance counters and ranking algorithms that generated more consistent, explainable results.

Chapter 8: Selecting Vtune HPCs for Ransomware Detection

This chapter describes the refined side-channel analysis procedure in detail, which incorporates the lessons learned in the studies in Chapter 7. The experimental set up and general data collection procedure is provided in Section 8.1. Section 8.2 describes the process of validating collected training data and reducing/selecting features, along with classifier training and cross-validation results. Classifiers are evaluated with a separately collected set of test data, with classifier performance and test data results included in Section 8.3. Additional data analysis improvements are discussed in Sections 8.4 (start-time detection) and 8.5 (top feature comparison). This chapter concludes with an experiment summary and future work discussion in Section 8.6.

8.1 Study 3: 35-class, 90 HPC Feature Collection with Networked Database transfer and Parallel Report Generation

To further streamline the data collection process and continue towards a balanced data set, four more recent samples of DarkSide, Phobos, and Snatch were added to the Ransomware training set, along with a new Babuk ransomware sample. The full ransomware set consisted of 15 distinct samples, as shown in Figure 8.1.

Vtune HPCs for the reduced 90-feature set were collected in conjunction with current data, according to the parameters in Figure 8.2. Twenty trials each of the 35 operations were collected

Ransomware Sample	Sample Date
DarkSide	May 2021
REvil/Sodinokibi	May 2021
WannaCry	May 2021
Phobos	January 2022
REvil/Sodinokibi	February 2022
Gibberish	April 2022
HiddenTear	April 2022
Snatch	April 2022
Phobos	May 2022
WannaCry	June 2022
REvil/Sodinokibi	July 2022
Babuk	October 2022
DarkSide	May 2021
Phobos	September 2022
Snatch	May 2021

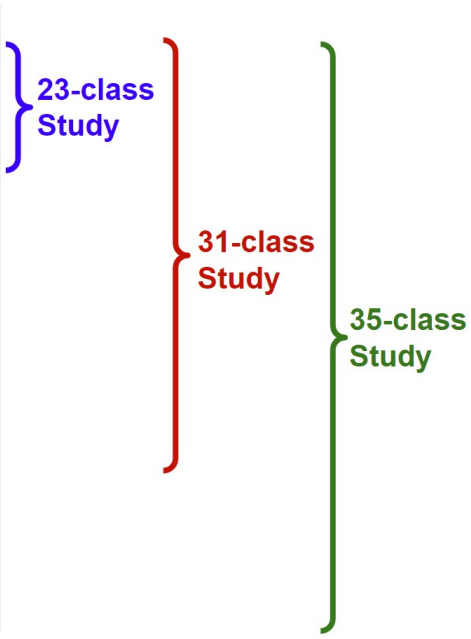


Figure 8.1: Study 3 included all ransomware samples used in Studies 1 and 2, plus 4 new ransomware versions.

in randomized order.

The data collection process was further streamlined by the incorporation of a networked storage server so that Vtune performance counter results could be `scp`'d to the server for report generation, in order to simplify the safe transfer process and allow Vtune reports to be processed in parallel. The resulting data flow is shown in Figure 8.3.

8.1.1 Training Data Verification

The Vtune reports generated for each of the randomized trials was again processed in MATLAB. First, the trials were sorted by operation and the HPC count from the Vtune report was averaged every 0.5s and plotted for the full 30s trial. A summary plot of HPC 48, `INST_RETIRED.ANY`, which counts the number of instructions retired from execution in the Vtune sampling window, shown in Figure 8.4. This process was repeated with the HPC count from the Vtune

# Trials (per operation)	Collection Method	Restore Process
20	SCRIPT	IFL

Trial Parameters		
OS idle (current only)	15	s
Vtune idle (add Vtune)	15	s
Test Operation Duration	15	s

Data Recorder Parameters		
Sampling Rate	200	kS/s
Duration	45	s
Span	4	A
Offset	-1.5	V

Operations Tested (35 total)		
Benign		
os, 7zip, 18 SPEC Workstation 3.1 CPU Benchmarks		
Ransomware		
Babuk_1022	Phobos_0122	Snatch_0422
DarkSide_Spr21	Phobos_0522	Snatch_0521
DarkSide_0521	Phobos_0922	Sodinokibi_0222
Gibberish_0422	Sodinokibi_Spr21	WannaCry_Spr22
HiddenTear_0422	Sodinokibi_0722	WannaCry_0622

Vtune Parameters		
HPCs	90	top 3 F1 at ea lvl
Sampling Interval	0.5	ms
Duration	30	s
Report "bins"	6,000	
Effective Sampling Rate	0.005	s

Figure 8.2: Data Collection Summary for Study 3.

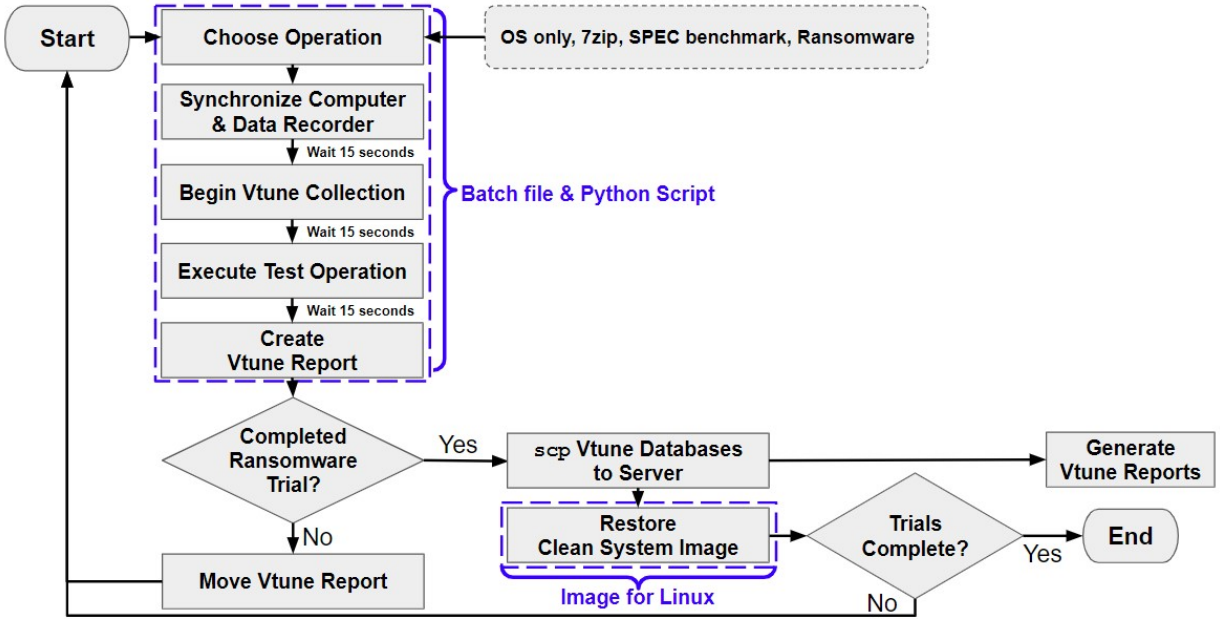


Figure 8.3: Data Collection, System Restore, and Networked Report Transfer/Generation Procedure for 35-class data set (Study 3).

report, but averaged every 0.1s and plotted from 10-20s, to provide additional granularity around operation trigger time (15s). Note that because the automated Python script launched Vtune, waited 15s, and then executed the test operation, the actual test start time varied slightly based on actual Vtune start up time.

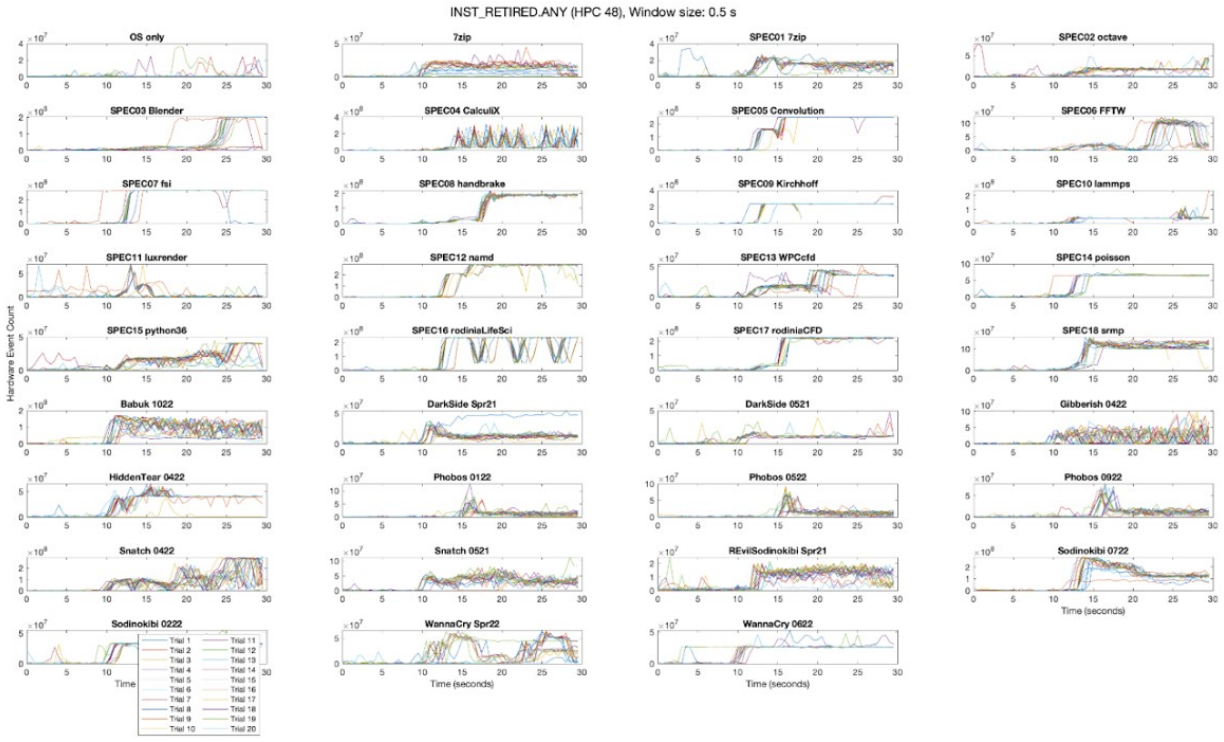


Figure 8.4: Summary of 20 Training Data Trials for each of 35 classes in Study 3 (HPC 48) `INST_RETIRED.ANY`.

Each of the 20 trials per operation was plotted individually for each operation and HPC. HPC 48, `INST_RETIRED.ANY`, was used to baseline training data and identify any clearly anomalous trials, which were repeated. An example for a Sodinokibi Ransomware sample is shown in Figure 8.5, before trial 151 was repeated.

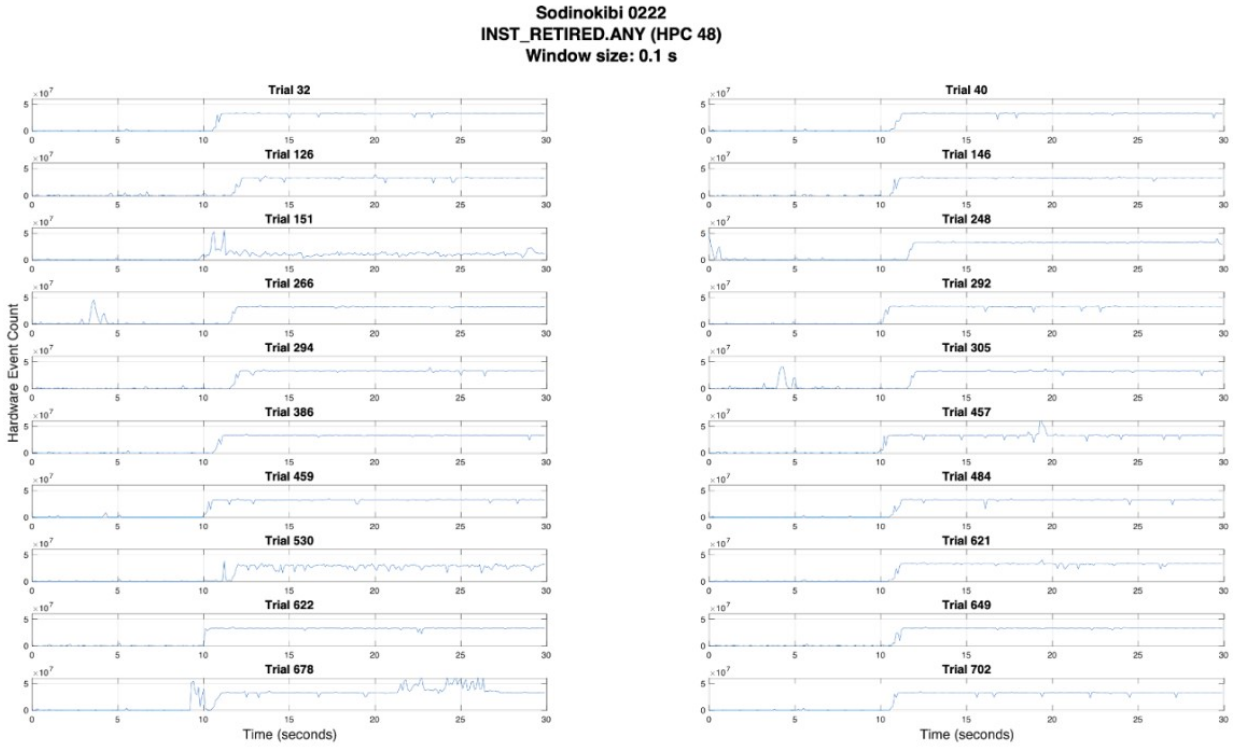


Figure 8.5: Example Plot of HPC 48 (`INST_RETIRED.ANY`) Individual Sodinokibi Trials for Training Data Verification Trial 151 was identified as anomalous and repeated.

8.2 Training Data Validation

Again, Vtune reports were processed in MATLAB to generate features for 3 windows of data at operation trigger time (15s) as well as the window before and after for 4 different window sizes (0.1, 0.5, 1, and 2 seconds), as described in Section 7.5.1. This yielded 12 sets of features, in order to capture the initial stages of a sliding window approach to ransomware detection. The start times, window sizes, and stop times are indicated in each column of the table below.

To optimize which of the reduced set of 90 HPCs would be used as training features, the same five classifiers identified in Section 7.5.2 were trained with a single HPC at a time, using a similar process to the one outlined in Section 7.5.4, with each of the 90 HPCs collected the in 35-class data set.

35-classes (15 Ransomwares, OS, 7zip, 18 SPEC benchmarks)

Start Time	14.9	15	15.1	14.5	15	15.5	14	15	16	13	15	17
Bin Size	0.1	0.1	0.1	0.5	0.5	0.5	1	1	1	2	2	2
Stop Time	15	15.1	15.2	15	15.5	16	15	16	17	15	17	19

Figure 8.6: Start Time - Window Size - Stop Time Combinations for Study 3. Each column indicates a time interval for which HPC feature sets were generated (12 total).

However, as was noted in the previous experiment, F1 (albeit a better metric than accuracy/ precision/recall for unbalanced data) does not account for the "4th" (True Negative) quadrant of the binary confusion matrix. This results in the class identified as "positive" having a more significant impact than the negative class. Since it is important both that ransomware is correctly identified AND that benign operations are not incorrectly classified as ransomware, Matthews' Correlation Coefficient was used as an additional metric to compare the performance of each classifier/HPC combination, as it gives equal weight to all 4 quadrants of the binary confusion matrix, as described in Section 8.2.1. The resulting process that was used to select the best combinations of HPCs for each classifier is described in Section 8.2.2.

8.2.1 Matthews' Correlation Coefficient (MCC)

Matthews' Correlation Coefficient [210], shown in Equation 8.1 is a special case of the Pearson Correlation Coefficient (commonly used to measure the linear correlation of two variables) applied to binary classification problems. Using the binary confusion matrix in Figure 7.29, the MCC is given by equation 8.1, below.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{8.1}$$

It is particularly relevant to imbalanced data sets as it weighs all 4 quadrants of a binary confusion matrix equally, where as Accuracy is skewed by unequal classes, where True Positives (Quadrant II) might account for a significantly greater portion of possible correct predictions than True Negatives (Quadrant IV). Meanwhile, F1 factors in Precision (calculated from TPs and FPs in Quadrants II and III) and Recall (calculated from TPs and FNs Quadrants I and II). When calculated directly from a binary confusion matrix, F1 can be represented by equation 8.2.

$$F1 = \frac{2 \cdot TP}{2 \cdot TP + FN + FP} \tag{8.2}$$

For imbalanced data sets, both Accuracy and F1 metrics can present a greatly inflated view of classifier performance, as described in the example below.

8.2.1.1 EXAMPLE: Accuracy, F1, and MCC for Imbalanced Data.

Consider Study 1 (Section 7.3), where data was collected for 20 trials each of 20 benign operations (400 total benign operations), and 20 trials each of 3 ransomware operations (60 trials total ransomware operations), for a total of 460 trials. A scenario where 355 benign trials were correctly predicted ("True Benign") and 5 ransomware trials were correctly predicted ("True Ransomware") would result in the confusion matrix in Figure 8.7.

Using the equations in Section 7.5.3.3, the resulting performance metrics for the confusion matrix in Figure 8.7 are shown in Table 8.1. While the Accuracy, Precision, Recall, and F1 metrics appear reasonable, the vast majority of ransomware trials were predicted to be benign

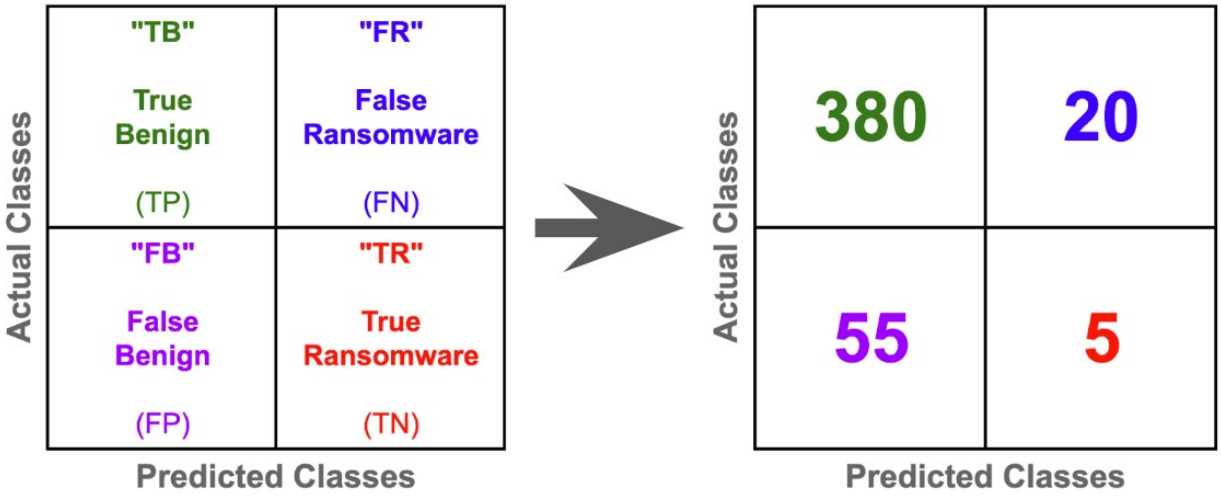


Figure 8.7: Example: Confusion Matrix for Matthews' Correlation Coefficient (MCC) vs. F1 Score for imbalanced data.

Table 8.1: Accuracy, Precision, Recall, F1, and MCC metrics for the Confusion Matrix shown in Figure 8.7.

ACCURACY	83.7%
PRECISION	87.4%
RECALL	95.0%
F1	0.91
MCC	0.05

operations. MCC in this scenario is 0.05, which indicates that the classifier only identified a single class decently, and the overall performance was quite poor.

8.2.2 HPC Feature Selection Process

To identify the optimal HPCs that would be used to train each classifier, all 5 classifiers were trained with each of the 90 HPCs collected in the 35-class data set, using the following process:

1. Each classifier was trained with each of the 90 HPCs, one at a time, for each of the twelve start time/bin size/stop time feature sets.

2. Accuracy, precision, recall, F1, and MCC for each classifier, start/window/stop time feature set, and HPC combination.

3. MCC was averaged across all start/window/stop time combinations for each HPC and classifier.

4. Ranked HPCs from highest to lowest MCC for each classifier.

5. Top 3 HPCs for each classifier were recorded.

6. Steps 1-5 were repeated 5 times for each of the 5 classifiers.

7. The HPC that was ranked highest across a majority of the 5 runs was selected as the best choice for that classifier across all feature sets.

8. The HPC(s) selected in the step 7 were held constant and steps 1-7 repeated, until the top 10 HPCs for each classifier were selected.

The 10 HPCs selected by consistently highest MCC for each classifier are shown in Figure 8.8. HPCs displayed in gray gave the highest MCC across five runs, but showed no appreciable improvement from the previous result.

Average MCC across each of the 4 window sizes (0.1 s window in blue, 0.5 window in red, 1s window in yellow, 2s window in purple) for each of the 5 classifiers for the number of optimized HPCs used to train each classifier are shown in Figure 8.9. Overall, average MCC increased as window size increased up to the 1s window, and the classifier performance when trained with 2s of data was classifier dependent. Bagged Tree, Fine KNN, and Narrow Neural Network results showed only minimal improvement after 6 HPCs were used for training,

	BAGGED TREE				SUBSPACE DISCRIMINANT				LINEAR SVM				FINE KNN				NARROW NEURAL NETWORK			
	HPC	Description	F1	MCC	HPC	Description	F1	MCC	HPC	Description	F1	MCC	HPC	Description	F1	MCC	HPC	Description	F1	MCC
1	48	INST_RETIRED.ANY	0.728	0.457	33	FRONTEND_RETIRED.L2_MISS_PS	0.737	0.469	33	FRONTEND_RETIRED.L2_MISS_PS	0.669	0.330	48	INST_RETIRED.ANY	0.726	0.453	37	FRONTEND_RETIRED.STLB_MISS_PS	0.758	0.515
2	55	ITLB_MISSES.WALK_PENDING	0.889	0.777	90	UOPS_ISSUED.SLOW_LEA	0.740	0.472	51	INT_MISC.RECOVERY_CYCLES	0.746	0.492	41	ICACHE_64B.IFTAG_STALL	0.879	0.758	51	INT_MISC.RECOVERY_CYCLES	0.842	0.684
3	81	MEM_LOAD_RETIRED.L2_HIT_PS	0.919	0.838	8	BR_MISP_RETIRED.CONDITIONAL_PS	0.770	0.538	69	L2_RQSTS.REFERENCES	0.855	0.710	39	ICACHE_64B.IFTAG_HIT	0.921	0.841	68	L2_RQSTS.PF_MISS	0.922	0.844
4	8	BR_MISP_RETIRED.CONDITIONAL_PS	0.928	0.855	64	L2_RQSTS.CODE_RD_MISS	0.777	0.553	75	MACHINE_CLEARS.MEMORY_ORDERING	0.875	0.750	58	L1D_PEND_MISS.PENDING	0.933	0.866	11	CPU_CLK_THREAD_UNHALTED.ONE_THREAD_ACTIVE	0.930	0.859
5	64	L2_RQSTS.CODE_RD_MISS	0.935	0.870	74	LONGEST_LAT_CACHE.REFERENCE	0.803	0.606	70	L2_RQSTS.RFO_HIT	0.898	0.795	2	BR_INST_RETIRED.FAR_BRANCH_PS	0.939	0.878	2	BR_INST_RETIRED.FAR_BRANCH_PS	0.933	0.867
6	2	BR_INST_RETIRED.FAR_BRANCH_PS	0.943	0.886	17	DSB2MITE_SWITCHES.COUNT	0.842	0.685	31	FP_ARITH_INST_RETIRED.SCALAR_DOUBLE	0.909	0.819	90	UOPS_ISSUED.SLOW_LEA	0.942	0.884	42	IDQ.ALL_MITE_CYCLES_4_UOPS	0.943	0.885
7	75	MACHINE_CLEARS.MEMORY_ORDERING	0.946	0.891	75	MACHINE_CLEARS.MEMORY_ORDERING	0.877	0.755	53	ITLB_MISSES.WALK_ACTIVE	0.921	0.842	31	FP_ARITH_INST_RETIRED.SCALAR_DOUBLE	0.944	0.889	43	IDQ.ALL_MITE_CYCLES_ANY_UOPS	0.948	0.896
8	56	L1D.REPLACEMENT	0.947	0.894	67	L2_RQSTS.PF_HIT	0.894	0.787	90	UOPS_ISSUED.SLOW_LEA	0.925	0.850	70	L2_RQSTS.RFO_HIT	0.945	0.891	56	L1D.REPLACEMENT	0.949	0.898
9	5	BR_INST_RETIRED.NEAR_TAKEN_PS	0.948	0.897	35	FRONTEND_RETIRED.LATENCY_GE_2	0.899	0.799	35	FRONTEND_RETIRED.LATENCY_GE_2	0.931	0.862	16	CYCLE_ACTIVITY.STALLS_L2_MISS	0.946	0.892	52	ITLB_MISSES.MISS_CAUSES_A_WALK	0.950	0.900
10	26	DTLB_STORE_MISSES.WALK_PENDING	0.949	0.897	77	MEM_INST_RETIRED.ALL_LOADS_PS	0.907	0.815	37	FRONTEND_RETIRED.STLB_MISS_PS	0.933	0.866	30	FP_ARITH_INST_RETIRED.256B_PACKED_SINGLE	0.946	0.892	77	MEM_INST_RETIRED.ALL_LOADS_PS	0.952	0.903

Figure 8.8: Top 10 ranked HPC features for each classifier for 5 rounds of cross validation and training.

while the Subspace Discriminant and Linear SVM classifier results improved for 8-10 features, depending on window size.

The following observations are noted:

1. Classifiers showed best results with longer windows of data (2s windows outperformed the smaller window sizes) with the exception of the 2 second window before operation start time. This is likely due to the inconsistency with the actual operation start time compared to Vtune start up time, and could be addressed by generating an operation start detector (future work).

2. Similar results were obtained from Bagged Tree, Fine KNN, and Narrow Neural Network classifiers with 6 HPCs (MCC of approximately 0.885 +/- 0.001). No appreciable improvement was gained by training these classifiers with additional HPC features.

3. Subspace Discriminant and Linear SVM showed consistent, definitive “best” HPC across feature sets across 5 runs for 8+ HPCs. However, even after 10 HPCs they still classified

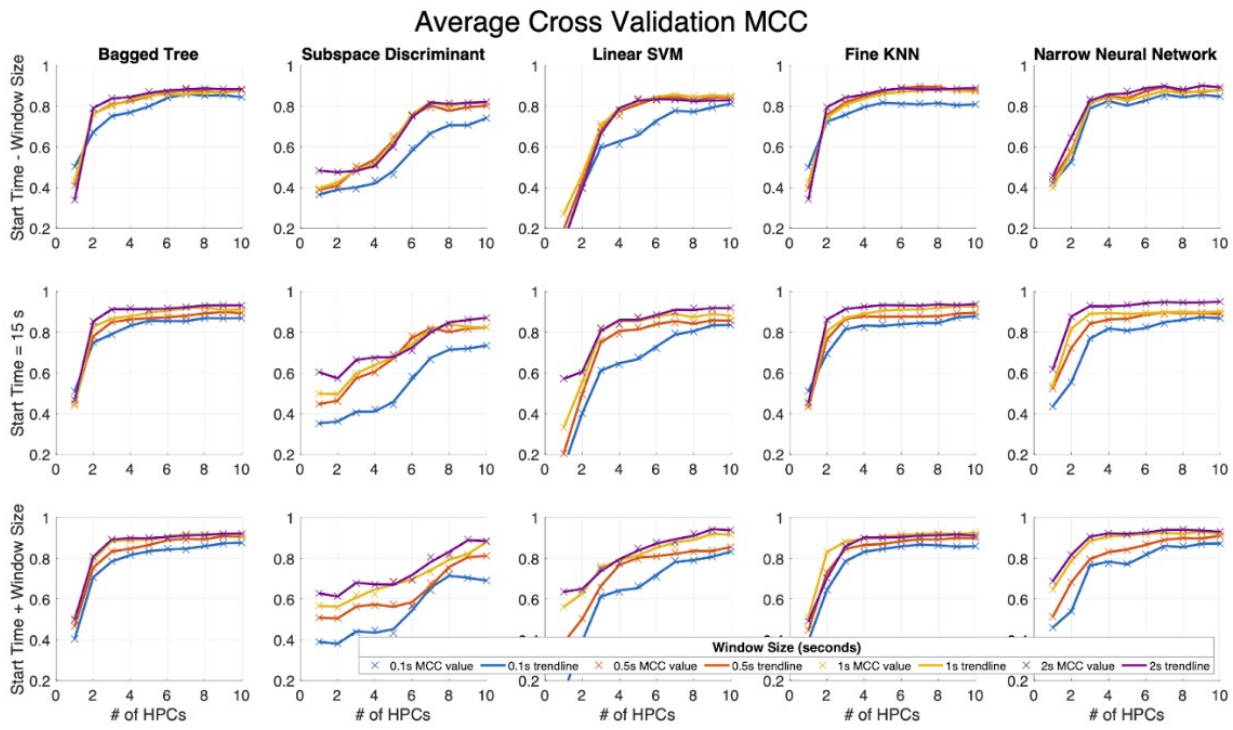


Figure 8.9: MCC scores averaged across each start/stop window for each classifier plotted against the number of optimized HPCs used to train the classifier.

less accurately than the Bagged Tree, Fine KNN, and Narrow Neural Network classifiers, which used fewer features. A possible future direction would be to continue to iterate on the "best" HPC for these two classifiers, in order to determine how many features can improve results before performance plateaus.

8.2.3 Cross-Validation of Training Data

Each classifier was trained with the top 1-10 optimized HPCs using 10-fold cross validation, and the resulting confusion matrices were examined. Figures 8.10-8.14 show cross validation confusion matrices for each classifier for the 0.1s, 0.5s, 1s, and 2s windows beginning at 15s (operation trigger time) for the top 6 HPCs from Figure 8.8. Numbers in the blue square in the top left quadrant of each matrix indicate the number of the (400) benign trials correctly predicted

as benign, while numbers in the blue in the bottom right quadrant indicate the number of (350) ransomware trials correctly predicted as ransomware.

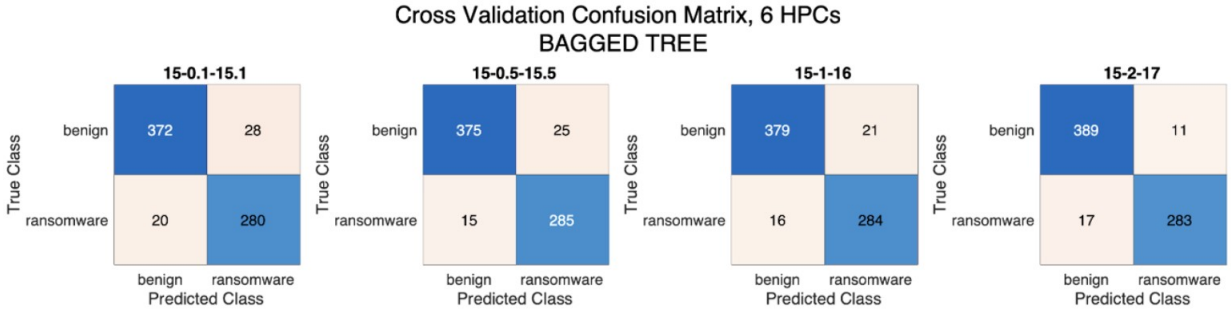


Figure 8.10: Cross validation confusion matrices for Bagged Tree Ensemble Classifier trained with 6 optimized HPCs. Window size increases from left to right.

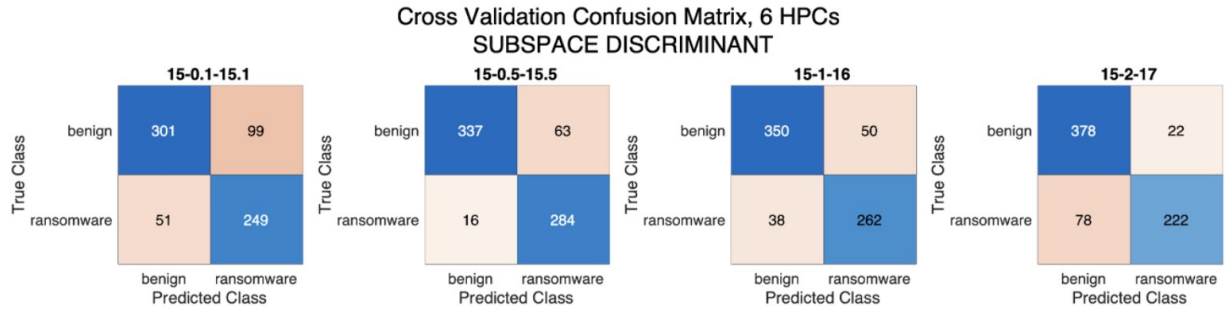


Figure 8.11: Cross validation confusion matrices for Subspace Discriminant Ensemble Classifier trained with 6 optimized HPCs. Window size increases from left to right.

MCC averaged across each set of window sizes (0.1s, 0.5s, 1s, 2s) for each classifier trained with 6 HPCs is shown in Figure 8.15. For the 0.1s window, Bagged Tree and Fine KNN slightly outperformed Narrow Neural Network on average, while Fine KNN predicted the most benign and ransomware operations correctly for the half second and 1 seconds windows. For a 2 second window, the Narrow Neural Network performed best on average with an MCC of 0.917. This corresponds to an F1 of 0.96 for the same classifier and 2s window.

Predictions (benign vs. ransomware) were analyzed for all trials and start/window/stop time features sets. Results for ransomware are in Figure 8.16, while Figure 8.17 show prediction

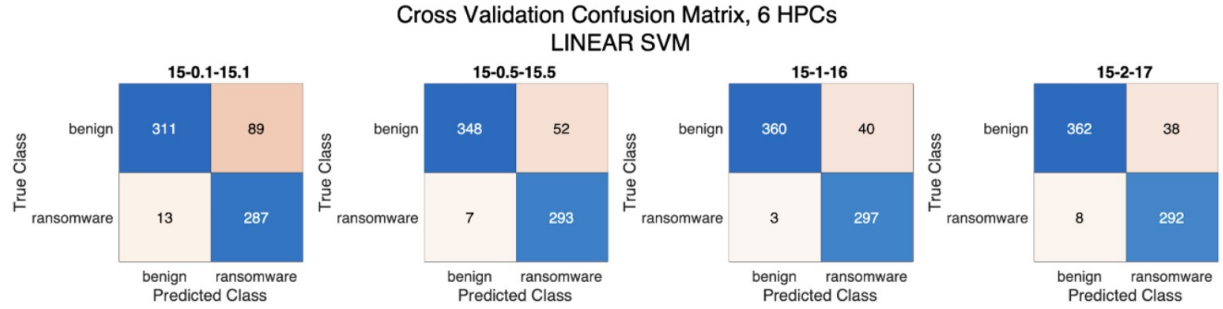


Figure 8.12: Cross validation confusion matrices for Linear SVM Classifier trained with 6 optimized HPCs. Window size increases from left to right.

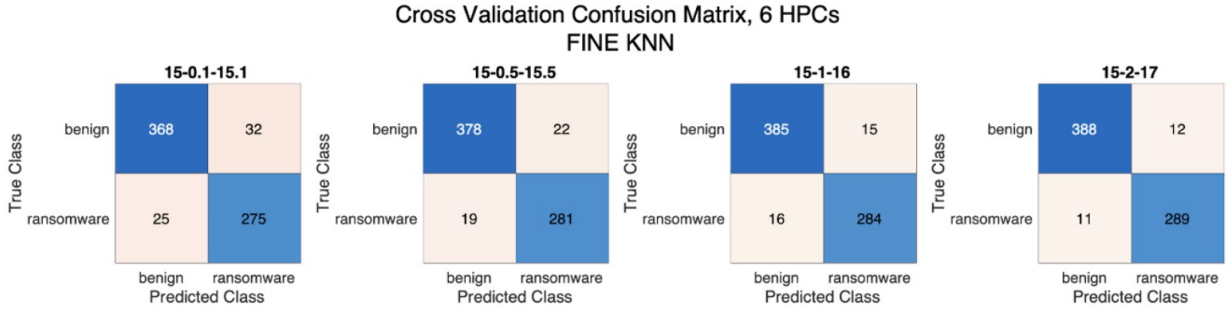


Figure 8.13: Cross validation confusion matrices for Fine KNN Classifier trained with 6 optimized HPCs. Window size increases from left to right.

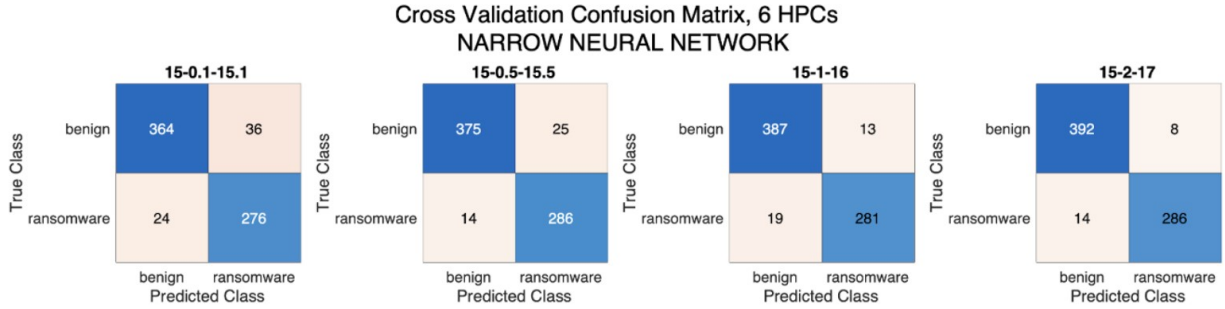


Figure 8.14: Cross validation confusion matrices for Narrow Neural Network Classifier trained with 6 optimized HPCs. Window size increases from left to right.

Average MCC by Window Size
for Classifiers trained with 6 HPCs

6 HPCs	0.1 s	0.5 s	1 s	2 s
FineKNN	0.8341	0.8844	0.9034	0.9077
NarrowNN	0.8301	0.8753	0.8940	0.9175
BaggedTree	0.8345	0.8702	0.8884	0.8951
LinearSVM	0.7175	0.8297	0.8634	0.8660
SubspaceDisc	0.5743	0.7005	0.7370	0.7179

Figure 8.15: Average MCC by window size for classifiers trained with 6 optimized HPCs.

accuracy results for benign operations. For both figures, operations for which 100% of trials were predicted correctly across all feature sets are in bold. Operation/classifier combinations which predicted less than 80% of trials correctly are in orange, while operation/classifier combinations that had fewer than 50% of trials predicted correctly are in red.

It is clear from Figures 8.16 and 8.17 that all classifiers except for the Subspace Discriminant Ensemble consistently predicted both benign and ransomware operations correctly, but all classifiers had significant issues correctly labeling "OS only" activity as benign. Additional analysis is required to determine which HPCs are required to better train the classifiers to correctly classify operating system background activity.

8.2.4 Classifier Training

Each of the 5 classifiers was trained using 1 through 9 optimized HPCs identified in Figure 8.8 as features with 10-fold cross validation for each of the start/stop windows in Figure 8.6. This resulted in nine trained classifiers, requiring 1-9 features as inputs for each of the twelve windows, resulting in 108 classifiers to evaluate for each of the five algorithms. These classifiers were evaluated with a separately collected set of test data, as described in Section 8.3.

	% Correct Predictions						BT					SD					LSVM					KNN					NNN				
	Overall	BT	SD	LSVM	KNN	NNN	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2
R_Babuk_1022	0.92	0.95	0.84	0.88	0.98	0.95	0.93	0.95	0.95	0.97	0.72	0.87	0.88	0.88	0.83	0.93	0.90	0.87	0.98	0.95	**1.00**	0.98	0.93	0.95	0.93	0.97
R_DarkSide_0521	0.93	0.98	0.82	0.85	**1.00**	1.00	0.98	0.98	0.98	0.98	0.68	0.73	0.87	**1.00**	0.95	**1.00**	0.78	0.65	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	**1.00**	**1.00**
R_DarkSide_Spr21	0.95	0.95	0.98	0.91	0.94	0.98	0.92	0.95	0.95	0.98	0.97	0.98	0.98	0.98	0.77	0.93	0.97	0.98	0.88	0.93	0.98	0.97	0.97	0.98	0.98	0.98
R_Gibberish_0422	0.98	0.97	**1.00**	**1.00**	0.98	0.96	0.95	0.97	0.98	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**	0.87	0.98	0.98	**1.00**
R_HiddenTear_0422	0.88	0.92	0.60	0.99	0.94	0.93	0.80	0.93	0.95	0.98	0.87	0.67	0.55	0.33	**1.00**	0.98	0.98	**1.00**	0.95	0.90	0.95	0.97	0.92	0.90	0.92	0.97
R_Phobos_0122	0.94	0.94	0.95	0.96	0.88	0.95	0.90	0.93	0.95	0.97	0.95	0.93	0.97	0.95	0.95	0.95	0.95	0.98	0.80	0.85	0.95	0.92	0.92	0.95	0.98	0.95
R_Phobos_0522	0.94	0.93	0.96	0.97	0.90	0.92	0.88	0.95	0.92	0.95	0.97	0.92	0.98	0.97	0.97	0.95	0.97	0.98	0.85	0.85	0.95	0.97	0.87	0.88	0.95	0.98
R_Phobos_0922	0.93	0.95	0.95	0.98	0.85	0.93	0.90	**1.00**	0.95	0.97	0.98	0.90	0.97	0.95	**1.00**	0.98	0.97	0.97	0.77	0.87	0.88	0.88	0.90	0.93	0.92	0.95
R_REvilSodinokibi_Spr21	0.96	0.96	**1.00**	0.96	0.93	0.95	0.95	0.97	0.97	0.95	**1.00**	**1.00**	**1.00**	**1.00**	0.85	**1.00**	**1.00**	**1.00**	0.82	0.98	0.97	0.93	0.95	0.97	0.95	0.95
R_Snatch_0422	0.93	0.94	0.96	0.97	0.89	0.91	0.87	0.97	0.95	0.97	0.97	0.92	0.98	0.97	0.97	0.95	0.97	0.98	0.80	0.87	0.93	0.97	0.83	0.87	0.95	0.98
R_Snatch_0521	0.98	0.99	0.99	0.97	0.98	0.96	0.98	0.98	**1.00**	**1.00**	**1.00**	0.98	0.98	**1.00**	0.92	**1.00**	0.98	0.98	0.98	0.97	**1.00**	0.98	0.92	0.98	0.97	0.98
R_Sodinokibi_0222	0.88	0.90	0.52	**1.00**	**1.00**	**1.00**	0.87	0.90	0.90	0.95	0.50	0.67	0.57	0.33	**1.00**	**1.00**	**1.00**	**1.00**	0.98	0.97	0.98	0.98	0.97	0.98	0.98	0.98
R_Sodinokibi_0722	0.91	0.90	0.75	1.00	0.99	0.94	0.97	0.87	0.92	0.83	0.48	0.73	0.80	0.97	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.90	0.95	0.98	0.92
R_WannaCry_0622	0.87	0.99	0.39	**1.00**	0.98	0.98	**1.00**	**1.00**	**1.00**	0.97	0.28	0.62	0.42	0.25	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	0.97	0.98	0.95	0.98	**1.00**	**1.00**
R_WannaCry_Spr22	0.76	0.73	0.77	0.88	0.73	0.69	0.82	0.80	0.65	0.63	0.95	0.80	0.78	0.55	0.97	0.98	0.80	0.75	0.77	0.80	0.63	0.70	0.82	0.72	0.65	0.58

Figure 8.16: Percentage of Correct Predictions by Classifier and Start/Stop Window for all ransomware trials.

	% Correct Predictions						BT				SD				LSVM				KNN				NNN			
	Overall	BT	SD	LSVM	KNN	NNN	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2
b_0S_only	0.40	0.59	0.33	0.17	0.51	0.41	0.48	0.62	0.63	0.62	0.02	0.32	0.37	0.63	0.00	0.00	0.32	0.35	0.43	0.55	0.50	0.57	0.13	0.35	0.52	0.63
b_7zip	0.78	0.93	0.28	0.77	0.95	0.96	0.95	0.97	0.92	0.90	0.05	0.32	0.32	0.42	0.77	0.73	0.77	0.80	0.95	0.97	0.97	0.93	0.97	0.95	0.97	0.95
b_SPEC01_7zip	0.97	1.00	0.90	0.93	1.00	1.00	**1.00**	**1.00**	0.98	**1.00**	0.87	0.83	0.97	0.95	0.93	0.85	0.97	0.98	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**	0.98	**1.00**
b_SPEC02_octave	0.82	0.89	0.67	0.74	0.88	0.91	0.82	0.92	0.88	0.93	0.60	0.60	0.73	0.75	0.67	0.70	0.78	0.80	0.80	0.90	0.88	0.93	0.82	0.92	0.97	0.93
b_SPEC03_Blender	0.80	0.91	0.60	0.65	0.89	0.93	0.85	0.90	0.95	0.95	0.27	0.58	0.78	0.78	0.37	0.63	0.80	0.82	0.80	0.93	0.93	0.90	0.88	0.92	0.92	0.98
b_SPEC04_CalculiX	0.95	0.94	0.95	0.96	0.96	0.95	0.92	0.92	0.93	**1.00**	0.92	0.93	0.98	0.98	0.97	0.95	0.95	0.97	0.92	0.95	0.98	**1.00**	0.88	0.93	0.97	**1.00**
b_SPEC05_Convolution	0.99	**1.00**	1.00	0.99	**1.00**	0.97	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	0.98	**1.00**	1.00	0.98	**1.00**	**1.00**	**1.00**	**1.00**	0.97	0.97	0.97	0.97
b_SPEC06_FFTW	0.96	0.96	0.90	0.98	1.00	0.95	0.92	0.98	0.95	**1.00**	0.72	0.98	0.97	0.95	**1.00**	0.97	0.98	0.98	**1.00**	0.98	**1.00**	**1.00**	0.93	0.97	0.97	0.93
b_SPEC07_fsi	1.00	**1.00**	**1.00**	1.00	**1.00**	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	**1.00**	**1.00**	0.98
b_SPEC08_handbrake	0.99	0.99	**1.00**	**1.00**	0.99	0.98	**1.00**	0.98	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	0.98	0.98	**1.00**	**1.00**	0.97	0.98	0.98
b_SPEC09_Kirchhoff	1.00	1.00	1.00	**1.00**	**1.00**	1.00	**1.00**	**1.00**	**1.00**	0.98	**1.00**	**1.00**	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98
b_SPEC10_lammps	1.00	0.99	0.99	**1.00**	**1.00**	**1.00**	0.98	0.98	0.98	**1.00**	**1.00**	**1.00**	0.98	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC11_luxrender	0.84	0.85	0.85	0.79	0.84	0.88	0.82	0.82	0.88	0.90	0.78	0.85	0.82	0.95	0.78	0.73	0.77	0.88	0.77	0.82	0.85	0.92	0.80	0.87	0.90	0.93
b_SPEC12_namd	0.99	1.00	0.98	**1.00**	1.00	0.98	**1.00**	0.98	**1.00**	**1.00**	**1.00**	0.98	0.97	0.95	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	**1.00**	**1.00**	**1.00**	0.95	0.98	**1.00**
b_SPEC13_WPCcfd	0.89	0.90	0.76	0.85	0.97	0.98	0.87	0.92	0.90	0.93	0.42	0.83	0.90	0.90	0.48	0.97	0.97	0.97	0.90	0.97	**1.00**	**1.00**	0.92	**1.00**	0.98	**1.00**
b_SPEC14_poisson	1.00	1.00	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	**1.00**	**1.00**	**1.00**	0.97	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC15_python36	0.88	0.95	0.73	0.79	0.97	0.98	0.93	0.93	0.98	0.97	0.47	0.75	0.90	0.78	0.55	0.77	0.92	0.93	0.97	0.97	0.98	0.97	0.95	0.98	**1.00**	0.97
b_SPEC16_rodiniaLifeSci	1.00	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.97	0.97	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC17_rodiniaCFD	0.94	0.98	0.99	0.76	0.99	0.95	0.95	0.97	0.98	**1.00**	0.98	0.98	**1.00**	**1.00**	0.18	0.93	0.97	0.97	**1.00**	0.97	**1.00**	**1.00**	0.98	0.92	0.95	0.97
b_SPEC18_srmp	0.98	**1.00**	0.99	0.94	0.99	0.97	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	0.97	0.90	0.95	0.97	0.93	**1.00**	0.98	0.98	**1.00**	0.95	0.95	0.98	**1.00**

Figure 8.17: Percentage of Correct Predictions by Classifier and Start/Stop Window for all benign trials.

8.3 Classifier Evaluation Results

This section describes the method to collect test data and evaluate the classifiers trained in Section 8.2.4 with a incremental number of optimized HPC features.

8.3.1 Test Data Set

To evaluate the trained classifiers, two sets of test data were collected and integrated into a single collection of test trials. The first half of testing data consisted of 5 trials each of the 35 total (15 ransomware, 20 benign) operations identified in Figure 8.2 in randomized order, collected on the same system used to collect the training data. The second set of test data was collected on a different system with identical hardware, and consisted of another 5 trials each for the 35 total operations, collected in a (different) randomized order. Both sets of test data were combined and the same start/stop windows of data shown in Figure 8.6 were provided as input to each of the trained classifiers in randomized order.

8.3.1.1 Classifier Performance

Each (start/window/HPC count) version of the classification algorithms was evaluated with test data at corresponding start times and window sizes for 1-9 HPC features. Figure 8.18 shows the average classification accuracy for each window size, calculated by averaging the accuracy results for the three start/stop combinations for each of the 0.1s, 0.5s, 1s, and 2s window sizes, for each classifier. Each data point for each window size and HPC count represents the average results of 3 sets of test data, so each individual subplot represents the evaluation results of all 108 (start/window/no. of features) versions of each classification algorithm. The corresponding

evaluation results by MCC shown in Figure 8.19. Three of the five classifiers evaluated (Bagged Tree, Fine KNN, and Narrow Neural Network) showed better than 90% classification accuracy for all windows with 4 HPC features or above, and only a marginal increase when trained with 6 or more HPC features. As the data set is still slightly unbalanced, the MCC results (Figure 8.19) provide more insight into the overall performance of the classifiers, and show that overall performance improves as window size increases up to 1s, but there is limited value from the additional second of data. Again, the Bagged Tree, Fine KNN, and Narrow Neural Network classifiers see only a marginal improvement in performance with more than 6 HPC features. Meanwhile, Subspace Discriminant and Linear SVM classifiers continue to improve with additional features, and it takes 9 HPC features for Linear SVM to reach the same overall performance of the Bagged Tree, Fine KNN, and Narrow Neural Network classifiers. The Subspace Discriminant classifier only achieves an MCC above 0.9 with a 2s window and 9 HPC features, with an MCC at least 0.1 lower than Bagged Tree, Subspace Discriminant, and Narrow Neural Network for 6 HPCs or fewer.

As the majority of classifiers saw only marginal improvement with more than 6 HPC features, Figure 8.20 summarizes the test results with the average MCC for each set of window sizes (0.1s, 0.5s, 1s, 2s) for each classifier tested with 6 HPC features. The Fine KNN test results have the highest MCC for all windows except 0.5s, where Bagged Tree has a slight edge.

8.3.1.2 Classifier Performance Results Analysis

The confusion matrices in Figures 8.21-8.25 show classification results for the 350 test trials at the 0.1, 0.5, 1, and 2 second windows of data beginning at operation start time (15s)

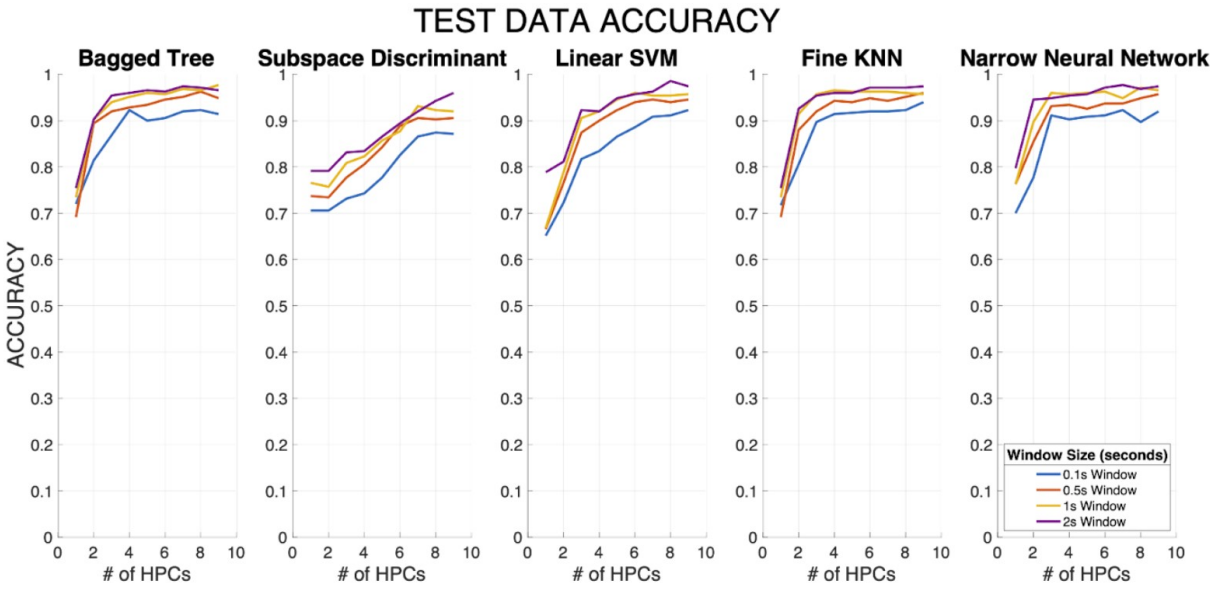


Figure 8.18: Classification Accuracy for classifiers evaluated with separate test data, collected in randomized order on two different systems with identical hardware.

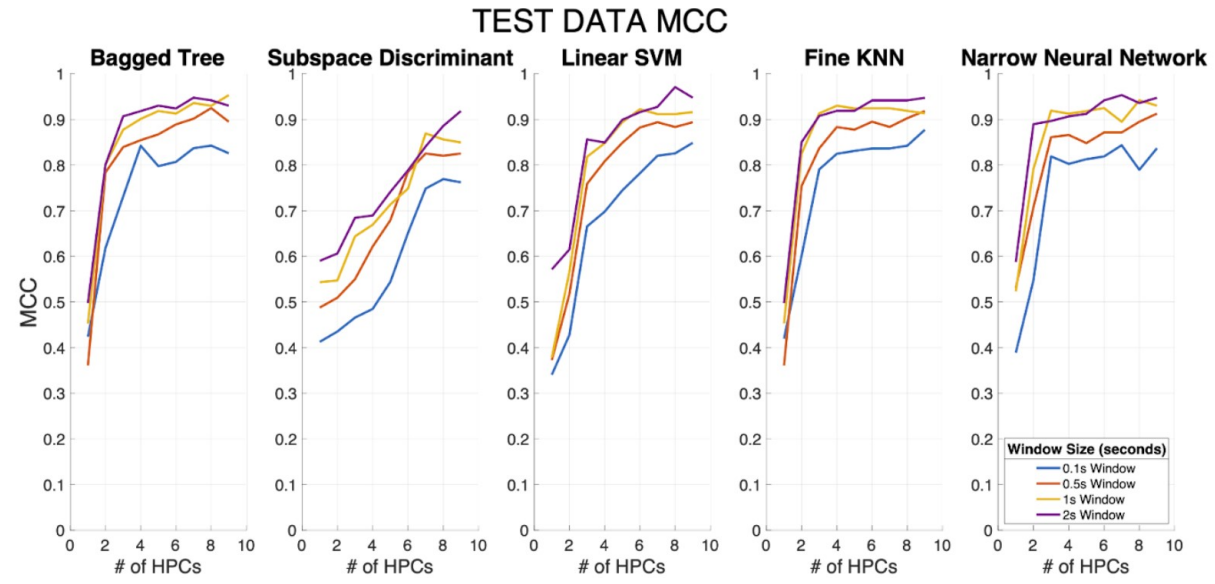


Figure 8.19: Classification MCC for classifiers evaluated with separate test data, collected in randomized order on two different systems with identical hardware.

Average MCC by Window Size for Classifiers tested with 6 HPCs

6 HPCs	0.1 s	0.5 s	1 s	2 s
FineKNN	0.8466	0.8835	0.9266	0.9206
NarrowNN	0.8290	0.8641	0.9225	0.9205
LinearSVM	0.7704	0.8690	0.9169	0.8993
BaggedTree	0.8371	0.8893	0.8913	0.9029
SubspaceDisc	0.6022	0.7212	0.7691	0.7915

Figure 8.20: Average MCC by window size for classifiers tested with 6 optimized HPC features.

for classifiers trained on 6 HPCs selected by highest MCC rank without any additional classifier optimizations. Classifier optimizations to improve classification results, such as weighting benign vs. ransomware mispredictions differently or considering additional factors (F1, overall accuracy) other than MCC alone, are left as future work.

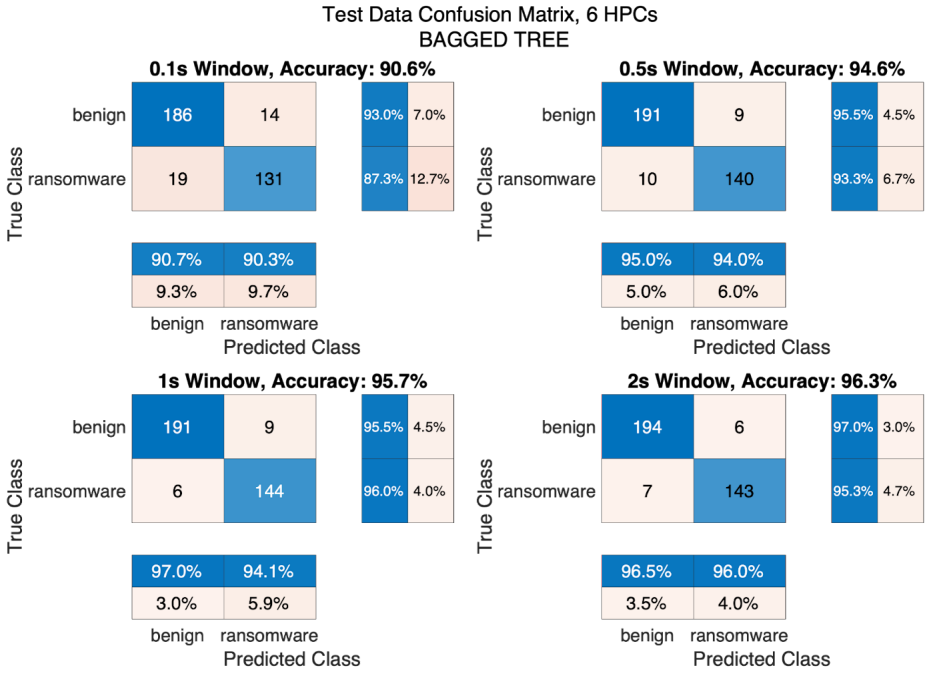


Figure 8.21: Confusion Matrix for Bagged Tree Ensemble classifier trained with 6 optimized HPC features for window sizes of 0.1s, 0.5s, 1s, and 2s at operation trigger time.

The highest accuracy numbers are obtained from the 1s and 2s windows for both the Fine

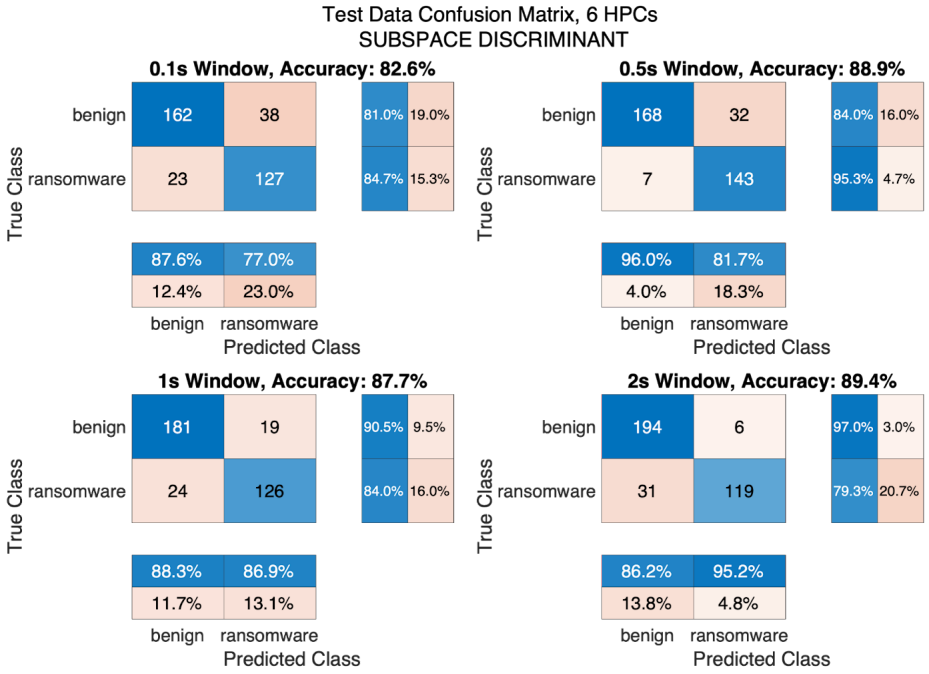


Figure 8.22: Confusion Matrix for Subspace Discriminant Ensemble classifier trained with 6 optimized HPC features for window sizes of 0.1s, 0.5s, 1s, and 2s at operation trigger time.

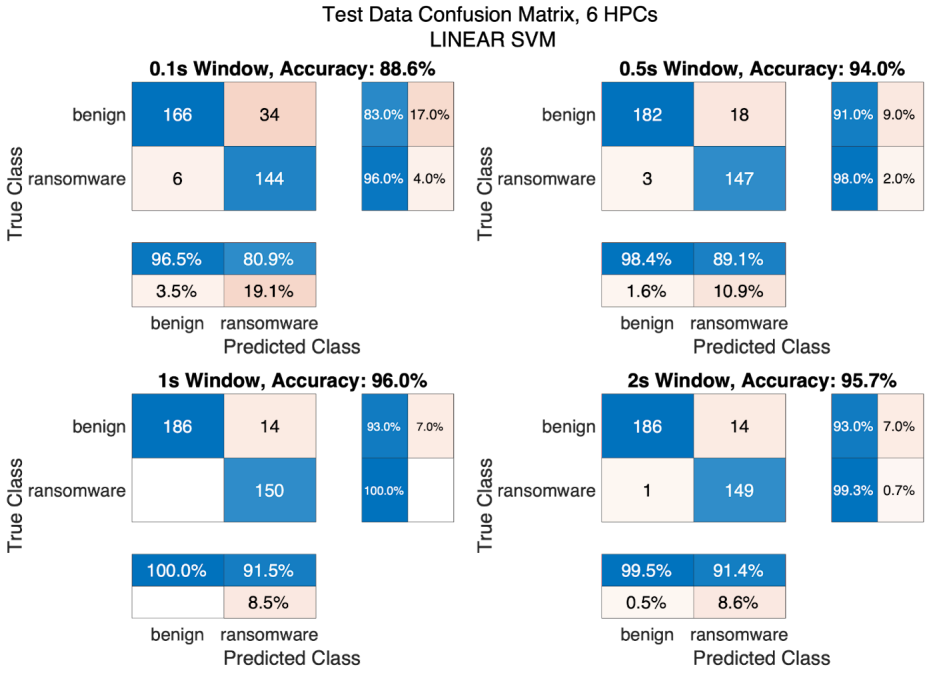


Figure 8.23: Confusion Matrix for Linear SVM classifier trained with 6 optimized HPC features for window sizes of 0.1s, 0.5s, 1s, and 2s at operation trigger time.

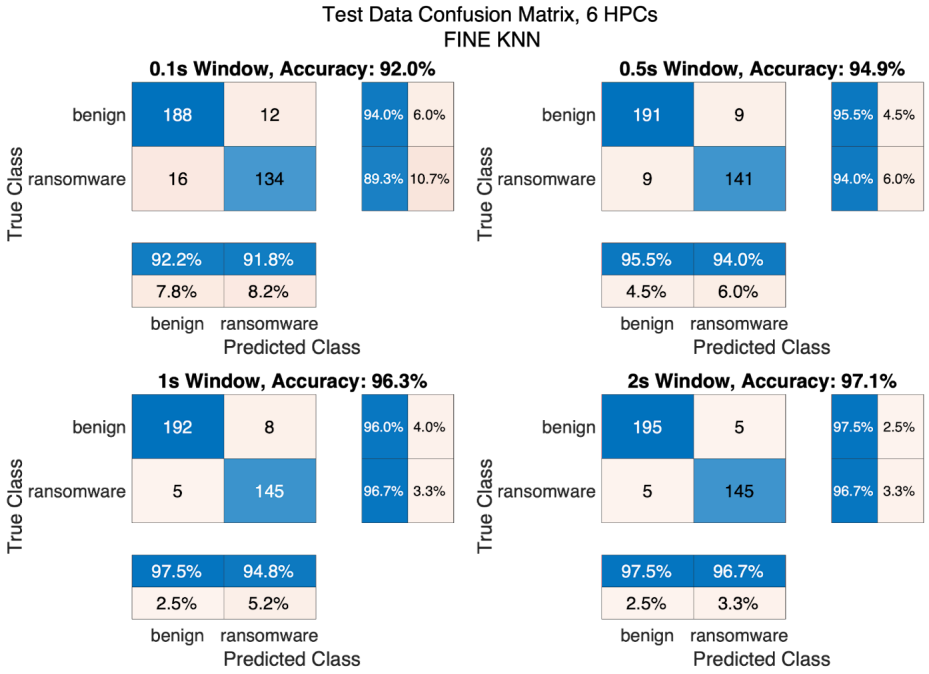


Figure 8.24: Confusion Matrix for Fine KNN classifier trained with 6 optimized HPC features for window sizes of 0.1s, 0.5s, 1s, and 2s at operation trigger time.

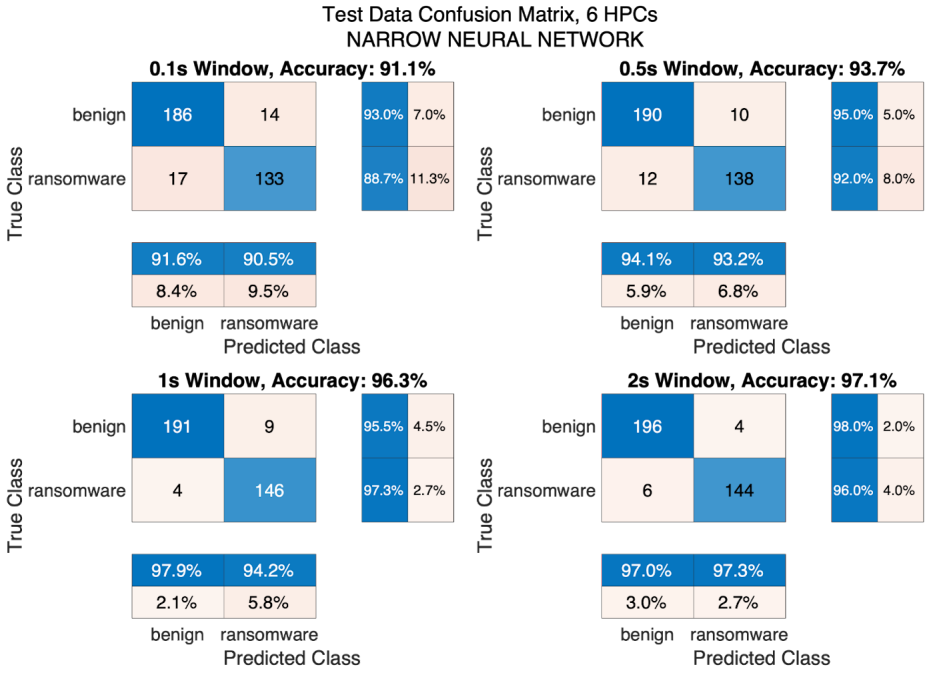


Figure 8.25: Confusion Matrix for Narrow Neural Network classifier trained with 6 optimized HPC features for window sizes of 0.1s, 0.5s, 1s, and 2s at operation trigger time.

KNN (Figure 8.24) and Narrow Neural Network (Figure 8.25) classifiers, with both showing 96.3% accuracy with the 1s window and 97.1% accuracy at the 2s window, albeit with slightly different False Positive and True Negative numbers. The Fine KNN classifier also had the highest accuracy numbers for the 0.1s and 0.5s windows. However, of all benign operations mispredicted as ransomware by the Fine KNN classifier (the upper right quadrant of each confusion matrix in Figure 8.24), 82% of errors were attributable to misprediction of "OS only" (system idle) trials. This consisted of eight of the 12 trials in the 0.1s window, 8 of 9 in the 0.5s window, 7 of 8 in the 1s window, and all 5 (of 5) in the 2s window. For the 1s and 2s windows of the Narrow Neural Network classifier (Figure 8.25), OS only misclassification accounted for 7/9 and 4/4 incorrect predictions, respectively. WannaCry was the most frequent Ransomware incorrectly identified as a benign operation, accounting for approximately half of the mispredictions. Figure 8.26 shows the percentage of correct predictions for all twelve sets of start/stop windows in Figure as predicted by the respective 6 HPC feature classifier. Figure 8.27 shows the comparable information for the benign operations. In both figures, perfect prediction across all 12 start/stop windows is indicated in bold. Predictions <80% correct are in orange font, while red font highlights <50% correct. The Fine KNN and Narrow Neural Network classifiers generated the most correct predictions across the board, but both identified <80% of WannaCry (Spr22 version) correctly, and all classifiers predicted <50% idle (OS only) operations correctly.

8.4 Towards Start-time Ransomware Detection

To enable further exploration of the significant number of misclassifications of "idle" (OS only) operations, the entire Study detailed in Sections 8.1-8.3 was replicated, with one change:

	% Correct Predictions					
	Overall	BT	SD	LSVM	KNN	NNN
R_Babuk_1022	0.90	0.93	0.85	0.94	0.93	0.83
R_DarkSide_0521	0.93	0.93	0.83	0.90	**1.00**	0.99
R_DarkSide_Spr21	0.93	0.90	0.96	0.89	0.97	0.95
R_Gibberish_0422	0.98	0.95	**1.00**	**1.00**	0.98	0.96
R_HiddenTear_0422	0.86	0.84	0.58	0.98	0.98	0.93
R_Phobos_0122	0.95	0.96	0.98	**1.00**	0.86	0.96
R_Phobos_0522	0.95	0.97	0.97	0.97	0.93	0.93
R_Phobos_0922	0.91	0.93	0.93	0.98	0.83	0.89
R_REvilSodinokibi_Spr21	0.97	0.98	**1.00**	0.98	0.97	0.95
R_Snatch_0422	0.95	**1.00**	0.93	0.95	0.98	0.89
R_Snatch_0521	0.99	**1.00**	**1.00**	0.96	0.97	**1.00**
R_Sodinokibi_0222	0.88	0.90	0.52	**1.00**	**1.00**	0.97
R_Sodinokibi_0722	0.94	0.96	0.77	**1.00**	**1.00**	0.98
R_WannaCry_0622	0.89	**1.00**	0.45	**1.00**	**1.00**	0.98
R_WannaCry_Spr22	0.82	0.78	0.83	0.92	0.78	0.79

Figure 8.26: Percentage of Correct Predictions for 6 HPC feature Classifiers for all (12) sets of Start/Stop Time Windows for all ransomware trials.

	% Correct Predictions					
	Overall	BT	SD	LSVM	KNN	NNN
b_0S_only	0.33	0.43	0.37	0.15	0.32	0.38
b_7zip	0.87	0.95	0.42	0.99	**1.00**	0.98
b_SPEC01_7zip	0.97	0.95	0.95	0.97	**1.00**	0.99
b_SPEC02_octave	0.91	0.97	0.74	0.89	0.96	0.98
b_SPEC03_Blender	0.88	0.96	0.71	0.78	0.96	0.99
b_SPEC04_CalculiX	0.97	0.97	0.95	0.97	0.97	0.98
b_SPEC05_Convolution	0.98	0.98	0.99	0.98	0.99	0.95
b_SPEC06_FFTW	0.96	0.99	0.84	**1.00**	**1.00**	0.98
b_SPEC07_fsi	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC08_handbrake	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC09_Kirchhoff	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC10_lammps	1.00	0.99	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC11_luxrender	0.92	0.95	0.94	0.89	0.92	0.91
b_SPEC12_namd	1.00	0.99	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC13_WPCcfd	0.94	0.96	0.85	0.90	0.98	**1.00**
b_SPEC14_poisson	0.98	0.94	**1.00**	**1.00**	**1.00**	0.95
b_SPEC15_python36	0.91	0.97	0.80	0.85	0.97	0.98
b_SPEC16_rodiniaLifeSci	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC17_rodiniaCFD	0.93	0.93	0.98	0.76	0.98	**1.00**
b_SPEC18_srmp	0.99	**1.00**	0.99	0.98	0.99	0.97

Figure 8.27: Percentage of Correct Predictions for 6 HPC feature Classifiers for all (12) sets of Start/Stop Time Windows for all benign trials.

"Windows" of data were replaced by a (visually identified) operation start time, as described in the following Section.

8.4.1 Identification of Operation Start Time

As evidenced from Figure 8.4, the true start time of the actual operation tested varied over a range of tenths of seconds to nearly 15 seconds between trial runs of the same operation. To mitigate this while repeating the entire experiment, the following steps were taken:

1. For each individual trial (of both training and testing data), operation start time was visually identified and recorded to a tenth of a second. For "OS only" (idle) trials, which did not include an operation, operation start time was considered to be 15.0 seconds.

2. Features for the same 0.1, 0.5, 1, and 2s durations were generated for each trial, beginning at the start time identified for each individual trial.

3. Classifiers identified in Section 7.5.2.2 were iteratively trained and HPCs generating the highest MCC values were ranked using the process in Section 8.2.2 to identify the top 10 HPCs for classifier training and follow-on testing. Figure 8.28 shows the resulting ranking.

4. Classifiers were trained with the top 1-9 MCC-ranked HPCs, as described in Section 8.2.3. Cross validation Accuracy and MCC results are shown in Figure 8.29.

5. Test data with visually identified start times for the same 0.1, 0.5, 1, and 2 second durations were provided to trained classifiers for performance analysis. Results are shown in Figure 8.30.

	BAGGED TREE			SUBSPACE DISCRIMINANT			LINEAR SVM			FINE KNN			NARROW NEURAL NETWORK		
	HPC	Description	MCC	HPC	Description	MCC	HPC	Description	MCC	HPC	Description	MCC	HPC	Description	MCC
1	67	L2_RQSTS.PF_HIT	0.627	64	L2_RQSTS.CODE_RD_MISS	0.635	64	L2_RQSTS.CODE_RD_MISS	0.634	67	L2_RQSTS.PF_HIT	0.616	64	L2_RQSTS.CODE_RD_MISS	0.675
2	45	IDQ.MS_MITE_UOPS	0.902	79	MEM_INST_RETIRED.SPLIT_STORES_PS	0.652	69	L2_RQSTS.REFERENCES	0.708	2	BR_INST_RETIRED.FAR_BRANCH_PS	0.899	39	ICACHE_64B.IFTAG_HIT	0.856
3	75	MACHINE_CLEARS.MEMORY_ORDERING	0.926	62	L2_RQSTS.ALL_DEMAND_REFERENCES	0.670	85	UOPS_DISPATCHED_PORT.PORT_7	0.823	77	MEM_INST_RETIRED.ALL_LOADS_PS	0.940	62	L2_RQSTS.ALL_DEMAND_REFERENCES	0.923
4	2	BR_INST_RETIRED.FAR_BRANCH_PS	0.939	8	BR_MISP_RETIRED.CONDITIONAL_PS	0.676	60	L2_LINES_OUT.NON_SILENT	0.849	18	DSB2MITE_SWITCHES.PENALTY_CYCLES	0.955	46	IDQ.MS_UOPS	0.950
5	80	MEM_LOAD_RETIRED.L1_HIT_PS	0.952	12	CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE	0.698	7	BR_MISP_RETIRED.ALL_BRANCHES_PS	0.875	14	CYCLE_ACTIVITY.CYCLES_L1D_MISS	0.964	85	UOPS_DISPATCHED_PORT.PORT_7	0.952
6	7	BR_MISP_RETIRED.ALL_BRANCHES_PS	0.958	59	L2_LINES_IN.ALL	0.715	32	FRONTEND_RETIRED.L1I_MISS_PS	0.886	35	FRONTEND_RETIRED.LATENCY_GE_2	0.972	2	BR_INST_RETIRED.FAR_BRANCH_PS	0.964
7	55	ITLB_MISSES.WALK_PENDING	0.959	18	DSB2MITE_SWITCHES.PENALTY_CYCLES	0.734	56	L1D.REPLACEMENT	0.890	57	L1D_PEND_MISS.FB_FULL	0.972	18	DSB2MITE_SWITCHES.PENALTY_CYCLES	0.969
8	44	IDQ.DSB_UOPS	0.964	85	UOPS_DISPATCHED_PORT.PORT_7	0.775	67	L2_RQSTS.PF_HIT	0.897	78	MEM_INST_RETIRED.SPLIT_LOADS_PS	0.971	51	INT_MISC.RECOVERY_CYCLES	0.975
9	35	FRONTEND_RETIRED.LATENCY_GE_2	0.960	46	IDQ.MS_UOPS	0.802	35	FRONTEND_RETIRED.LATENCY_GE_2	0.913	30	FP_ARITH_INST_RETIRED.256B_PACKED_SINGLE	0.972	88	UOPS_EXECUTED.X87	0.974
10	77	MEM_INST_RETIRED.ALL_LOADS_PS	0.964	50	INT_MISC.CLEAR_RESTEER_CYCLES	0.854	6	BR_INST_RETIRED.NOT_TAKEN	0.925	65	L2_RQSTS.DEMAND_DATA_RD_HIT	0.972	48	INST_RETIRED.ANY	0.971

Figure 8.28: Top 10 ranked HPC features for each classifier for 5 rounds of cross validation and training, with features generated for durations of 0.1, 0.5, 1, and 2s from visually identified operation start time.

8.4.2 Cross Validation and Testing Results

Figures 8.29 and 8.30 show the Accuracy (top) and MCC (bottom) as the number of HPC features used from training/testing increases to 9 HPC features for Training Cross Validation and Test Data, respectively. While the cross validation results are comparable to those obtained in Section 8.2.3, closer inspection of Figure 8.30 shows a slight decline in Accuracy and MCC across all classifiers for Test data at all window sizes, with a significant decrease (from 90% to 60%) in classification Accuracy for the 0.1s window of data using the visual method to identify operation start time. Figure 8.31, which has been reproduced from Figures 8.18 and 8.19 at the same scale as Figures 8.30 and 8.30 for comparison purposes. Respective MCC values dropped from approaching/passing 0.8 as the number of optimized HPC features increased (Figure 8.31, bottom) to a best result of 0.4 (for a Linear SVM using 2-4 HPC features), with most classifiers generating MCCs $\leq$0.2.

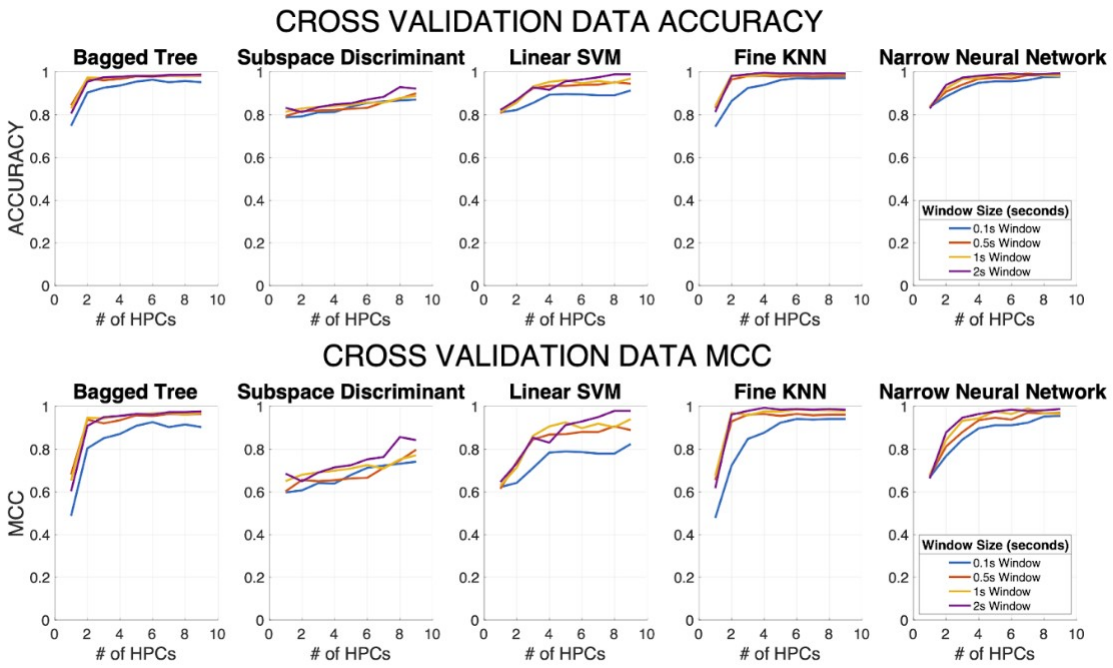


Figure 8.29: Cross Validation Accuracy (top) and MCC (bottom) for classifiers trained with the top 1-9 HPC features by MCC rank generated for durations of 0.1, 0.5, 1, and 2s from visually identified operation start time.

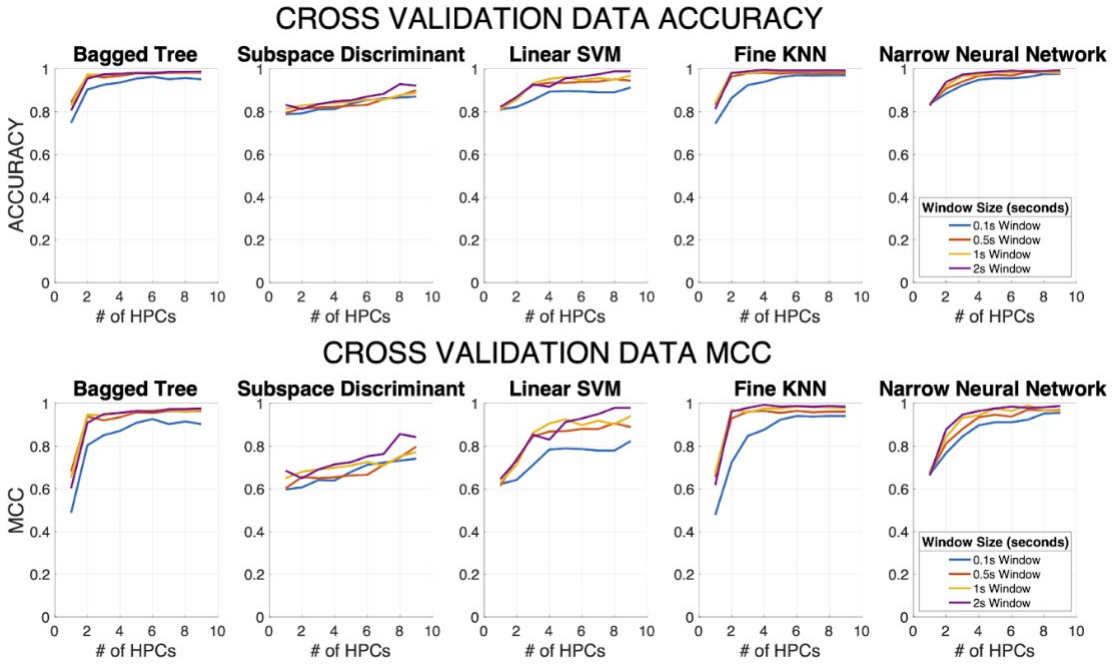


Figure 8.30: Test Data Accuracy (top) and MCC (bottom) for classifiers trained with the top 1-9 HPC features by MCC rank generated for durations of 0.1, 0.5, 1, and 2s from visually identified operation start time.

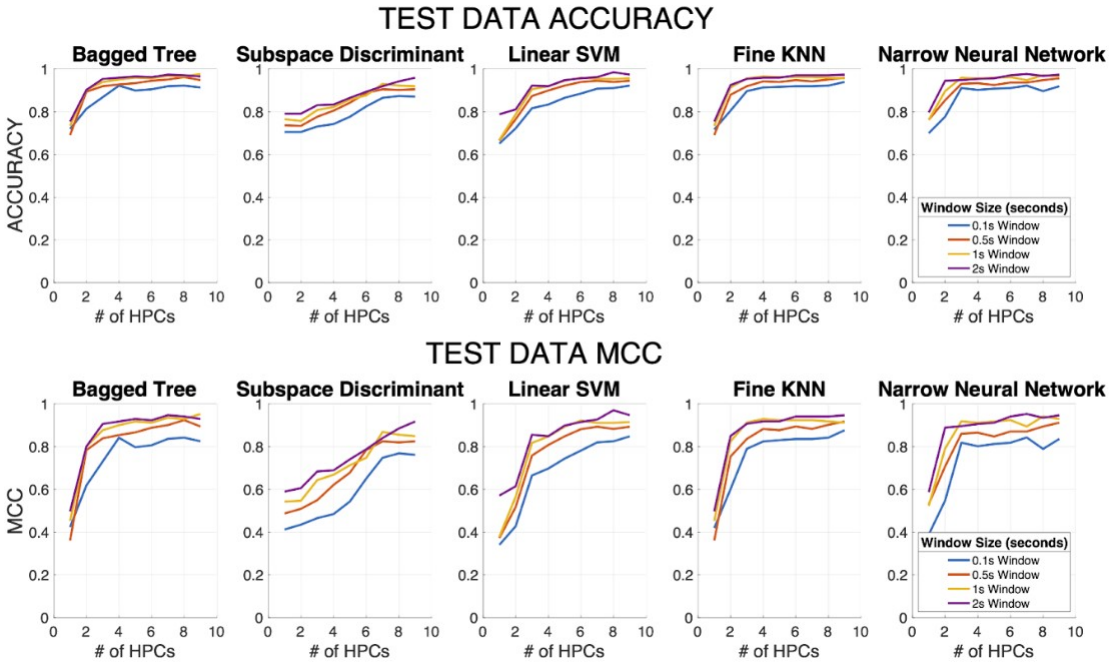


Figure 8.31: Summary of Test Data Accuracy (top) and MCC (bottom) for classifiers trained with the top 1-9 HPC features by MCC rank generated for windows of 0.1, 0.5, 1, and 2s beginning at 15s, adapted from Section 8.3.1 for comparison purposes.

8.4.3 Results

Figures 8.32 and 8.33 show the overall percentage of correct predictions for actual start time for data tested along with a breakdown by classifier and window size for benign and ransomware operations, respectively. For both figures, operations for which 100% of trials were predicted correctly across all feature sets are in bold. Operation/classifier combinations which predicted less than 80% of trials correctly are in orange, while operation/classifier combinations that had fewer than 50% of trials predicted correctly are in red.

Overall classification results were poorer than the "15s Start" results in Section 8.3.1, due primarily to four factors:

1. Poor overall performance of the 0.1s window after operation start time. This could have

been due to inconsistencies in determining the operation start time visually and recording the start time manually. This could be resolved by determining a threshold start value, but programming this is not insignificant due to the noisiness of the trial data introduced by background operations performed by the OS. Alternately, results in Section 8.3.1 could indicate that the probability of correctly classifying an operation an operation in progress with a tenth of a second of performance counter data is better than predicting an operation with the first tenth of a second of performance counter data.

2. The SPEC `Convolution` benchmark was overwhelmingly predicted to be ransomware, which the exception of the 2 second window for the Bagged Tree and Narrow Neural Network classifiers (as well as the 0.5s window for the NNN).

3. The SPEC `Python 3.6` benchmark, which performs a variety of math operations, was exclusively predicted to be ransomware by the Bagged Tree Classifier. The other 4 algorithms were moderately to 100% successful for the smallest windows, but best case performance for the 1s window or greater was only 0%-20% correct.

4. Three classifiers (SD, LSVM, NNN) had difficulties classifiying the June 2022 version of WannaCry at most window sizes.

Conversely, prediction of the "OS only" operation improved significantly (in all cases but the Linear SVM with a 1 second window), bringing the Percentage of Correct predictions up from 33% to 86% overall, and over 90% if the 1 second window LSVM results are discounted. The head-to-head comparison of overall % of correct predictions of benign operations is shown in Figure 8.34, while the corresponding values for ransomware operations are in Figure 8.35.

	% Correct Predictions						BT				SD				LSVM				KNN				NNN			
	Overall	BT	SD	LSVM	KNN	NNN	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2
b_0S_only	0.86	0.85	0.98	0.73	0.85	0.90	0.80	0.70	**1.00**	0.90	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	0.00	0.90	0.60	0.90	**1.00**	0.90	0.90	0.90	0.80	**1.00**
b_7zip	0.92	**1.00**	0.70	0.98	0.95	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.90	0.70	0.20	0.90	**1.00**	**1.00**	**1.00**	0.80	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**
b_SPEC01_7zip	0.83	0.85	0.73	0.85	0.90	0.80	0.70	0.80	0.90	**1.00**	0.30	0.80	0.80	**1.00**	0.50	0.90	**1.00**	**1.00**	0.60	**1.00**	**1.00**	**1.00**	0.20	**1.00**	**1.00**	**1.00**
b_SPEC02_octave	0.78	0.85	0.78	0.60	0.78	0.90	0.80	0.90	0.80	0.90	0.70	0.80	0.80	0.80	0.70	0.40	0.30	**1.00**	0.40	**1.00**	0.90	0.80	0.80	0.90	0.90	**1.00**
b_SPEC03_Blender	0.99	0.98	**1.00**	**1.00**	0.95	**1.00**	0.90	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.80	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC04_CalculiX	1.00	**1.00**	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC05_Convolution	0.16	0.28	0.00	0.00	0.00	0.53	0.20	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	**1.00**	0.00	**1.00**
b_SPEC06_FFTW	1.00	**1.00**	**1.00**	**1.00**	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.90
b_SPEC07_fsi	0.90	0.90	0.93	0.88	0.93	0.88	0.60	**1.00**	**1.00**	**1.00**	0.70	**1.00**	**1.00**	**1.00**	0.50	**1.00**	**1.00**	**1.00**	0.70	**1.00**	**1.00**	**1.00**	0.50	**1.00**	**1.00**	**1.00**
b_SPEC08_handbrake	0.94	0.93	**1.00**	**1.00**	0.88	0.90	0.70	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.90	0.60	**1.00**	**1.00**	0.60	**1.00**	**1.00**	**1.00**
b_SPEC09_Kirchhoff	0.97	**1.00**	**1.00**	**1.00**	**1.00**	0.83	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.30	**1.00**	**1.00**	**1.00**
b_SPEC10_lammps	0.73	0.80	0.75	0.80	0.80	0.50	0.30	**1.00**	0.90	**1.00**	0.10	0.90	**1.00**	**1.00**	0.20	**1.00**	**1.00**	**1.00**	0.30	0.90	**1.00**	**1.00**	0.10	0.10	**1.00**	0.80
b_SPEC11_luxrender	0.78	0.78	0.78	0.78	0.78	0.78	0.20	0.90	**1.00**	**1.00**	0.10	**1.00**	**1.00**	**1.00**	0.10	**1.00**	**1.00**	**1.00**	0.10	**1.00**	**1.00**	**1.00**	0.10	**1.00**	**1.00**	**1.00**
b_SPEC12_namd	0.90	0.95	0.80	0.90	0.93	0.90	0.80	**1.00**	**1.00**	**1.00**	0.20	**1.00**	**1.00**	**1.00**	0.60	**1.00**	**1.00**	**1.00**	0.70	**1.00**	**1.00**	**1.00**	0.60	**1.00**	**1.00**	**1.00**
b_SPEC13_WPCcfd	0.92	0.93	0.93	0.85	0.90	**1.00**	0.80	**1.00**	**1.00**	0.90	**1.00**	0.90	0.80	**1.00**	**1.00**	**1.00**	0.50	0.90	0.70	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC14_poisson	0.87	0.93	0.85	0.93	0.90	0.75	0.80	**1.00**	0.90	**1.00**	0.40	**1.00**	**1.00**	**1.00**	0.70	**1.00**	**1.00**	**1.00**	0.60	**1.00**	**1.00**	**1.00**	0.30	0.70	**1.00**	**1.00**
b_SPEC15_python36	0.29	0.00	0.58	0.35	0.23	0.28	0.00	0.00	0.00	0.00	0.90	**1.00**	0.20	0.20	0.90	0.30	0.20	0.00	0.60	0.20	0.10	0.00	0.10	**1.00**	0.00	0.00
b_SPEC16_rodiniaLifeSci	0.79	0.83	0.75	0.83	0.78	0.75	0.30	**1.00**	**1.00**	**1.00**	0.00	**1.00**	**1.00**	**1.00**	0.30	**1.00**	**1.00**	**1.00**	0.10	**1.00**	**1.00**	**1.00**	0.00	**1.00**	**1.00**	**1.00**
b_SPEC17_rodiniaCFD	0.89	0.83	**1.00**	0.90	0.85	0.88	0.30	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.60	**1.00**	**1.00**	**1.00**	0.40	**1.00**	**1.00**	**1.00**	0.50	**1.00**	**1.00**	**1.00**
b_SPEC18_srmp	0.94	0.80	0.98	0.98	0.95	0.98	0.20	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**	0.80	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**

Figure 8.32: % of Correct Predictions for benign operations, using 0.1, 0.5, 1, and 2s windows at actual operation start time (identified visually) and 9 HPC features. Operations for which 100% of trials were predicted correctly across all feature sets are in bold. Operation/classifier combinations which predicted less than 80% of trials correctly are in orange, while operation/classifier combinations that had fewer than 50% of trials predicted correctly are in red.

	% Correct Predictions						BT				SD				LSVM				KNN				NNN			
	Overall	BT	SD	LSVM	KNN	NNN	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2	0.1	0.5	1	2
R_Babuk_1022	0.94	0.98	0.85	0.90	0.98	**1.00**	0.90	**1.00**	**1.00**	**1.00**	0.60	0.90	0.90	**1.00**	0.70	0.90	**1.00**	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
R_DarkSide_0521	0.73	0.78	0.68	0.58	0.85	0.78	0.20	0.90	**1.00**	**1.00**	0.00	0.80	**1.00**	0.90	0.00	0.30	**1.00**	**1.00**	0.60	0.80	**1.00**	**1.00**	0.30	0.90	0.90	**1.00**
R_DarkSide_Spr21	0.91	0.88	0.88	0.85	0.95	0.98	0.60	0.90	**1.00**	**1.00**	0.50	**1.00**	**1.00**	**1.00**	0.40	**1.00**	**1.00**	**1.00**	**1.00**	0.80	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**
R_Gibberish_0422	0.88	0.80	0.90	0.90	0.88	0.93	0.60	**1.00**	0.90	0.70	0.60	**1.00**	**1.00**	**1.00**	0.60	**1.00**	**1.00**	**1.00**	0.60	0.90	**1.00**	**1.00**	0.70	**1.00**	**1.00**	**1.00**
R_HiddenTear_0422	0.84	0.85	0.73	0.90	0.93	0.78	0.60	0.90	0.90	**1.00**	0.30	0.90	0.90	0.80	0.70	0.90	**1.00**	**1.00**	0.80	**1.00**	0.90	**1.00**	0.40	0.90	0.80	**1.00**
R_Phobos_0122	0.88	0.90	0.83	0.85	0.95	0.88	0.60	**1.00**	**1.00**	**1.00**	0.30	**1.00**	**1.00**	**1.00**	0.40	**1.00**	**1.00**	**1.00**	0.80	**1.00**	**1.00**	**1.00**	0.50	**1.00**	**1.00**	**1.00**
R_Phobos_0522	0.82	0.83	0.75	0.78	0.90	0.83	0.40	0.90	**1.00**	**1.00**	0.10	0.90	**1.00**	**1.00**	0.20	0.90	**1.00**	**1.00**	0.60	**1.00**	**1.00**	**1.00**	0.40	0.90	**1.00**	**1.00**
R_Phobos_0922	0.84	0.83	0.83	0.83	0.88	0.83	0.40	0.90	**1.00**	**1.00**	0.40	0.90	**1.00**	**1.00**	0.40	0.90	**1.00**	**1.00**	0.60	0.90	**1.00**	**1.00**	0.40	0.90	**1.00**	**1.00**
R_REvilSodinokibi_Spr21	0.90	0.90	0.83	0.88	0.95	0.95	0.60	**1.00**	**1.00**	**1.00**	0.30	**1.00**	**1.00**	**1.00**	0.50	**1.00**	**1.00**	**1.00**	0.80	**1.00**	**1.00**	**1.00**	0.80	**1.00**	**1.00**	**1.00**
R_Snatch_0422	0.99	0.98	**1.00**	**1.00**	0.98	0.98	0.90	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**
R_Snatch_0521	0.91	0.90	0.88	0.90	0.95	0.90	0.60	**1.00**	**1.00**	**1.00**	0.50	**1.00**	**1.00**	**1.00**	0.60	**1.00**	**1.00**	**1.00**	0.80	**1.00**	**1.00**	**1.00**	0.60	**1.00**	**1.00**	**1.00**
R_Sodinokibi_0222	1.00	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
R_Sodinokibi_0722	0.90	0.88	0.83	0.90	0.93	0.98	0.50	**1.00**	**1.00**	**1.00**	0.40	0.90	**1.00**	**1.00**	0.60	**1.00**	**1.00**	**1.00**	0.70	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**
R_WannaCry_0622	0.49	0.70	0.13	0.35	0.78	0.50	0.00	0.80	**1.00**	**1.00**	0.00	0.10	0.20	0.20	0.00	0.00	**1.00**	0.40	0.10	**1.00**	**1.00**	**1.00**	0.10	0.60	0.30	**1.00**
R_WannaCry_Spr22	0.86	0.83	0.78	0.93	0.78	0.98	0.50	0.90	**1.00**	0.90	0.30	**1.00**	0.80	**1.00**	0.70	**1.00**	**1.00**	**1.00**	0.10	**1.00**	**1.00**	**1.00**	0.90	**1.00**	**1.00**	**1.00**

Figure 8.33: % of Correct Predictions for ransomware operations, using 0.1, 0.5, 1, and 2s windows at actual operation start time (identified visually) and 9 HPC features. Operations for which 100% of trials were predicted correctly across all feature sets are in bold. Operation/classifier combinations which predicted less than 80% of trials correctly are in orange, while operation/classifier combinations that had fewer than 50% of trials predicted correctly are in red.

	15s Start Time						Operation Start Time					
	% Correct Predictions						% Correct Predictions					
	Overall	BT	SD	LSVM	KNN	NNN	Overall	BT	SD	LSVM	KNN	NNN
b_0S_only	0.33	0.43	0.37	0.15	0.32	0.38	0.86	0.85	0.98	0.73	0.85	0.90
b_7zip	0.87	0.95	0.42	0.99	**1.00**	0.98	0.92	**1.00**	0.70	0.98	0.95	0.98
b_SPEC01_7zip	0.97	0.95	0.95	0.97	**1.00**	0.99	0.83	0.85	0.73	0.85	0.90	0.80
b_SPEC02_octave	0.91	0.97	0.74	0.89	0.96	0.98	0.78	0.85	0.78	0.60	0.78	0.90
b_SPEC03_Blender	0.88	0.96	0.71	0.78	0.96	0.99	0.99	0.98	**1.00**	**1.00**	0.95	**1.00**
b_SPEC04_CalculiX	0.97	0.97	0.95	0.97	0.97	0.98	1.00	**1.00**	**1.00**	0.98	**1.00**	**1.00**
b_SPEC05_Convolution	0.98	0.98	0.99	0.98	0.99	0.95	0.16	0.28	0.00	0.00	0.00	0.53
b_SPEC06_FFTW	0.96	0.99	0.84	**1.00**	**1.00**	0.98	1.00	**1.00**	**1.00**	**1.00**	**1.00**	0.98
b_SPEC07_fsi	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.90	0.90	0.93	0.88	0.93	0.88
b_SPEC08_handbrake	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.94	0.93	**1.00**	**1.00**	0.88	0.90
b_SPEC09_Kirchhoff	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.97	**1.00**	**1.00**	**1.00**	**1.00**	0.83
b_SPEC10_lammps	1.00	0.99	**1.00**	**1.00**	**1.00**	**1.00**	0.73	0.80	0.75	0.80	0.80	0.50
b_SPEC11_luxrender	0.92	0.95	0.94	0.89	0.92	0.91	0.78	0.78	0.78	0.78	0.78	0.78
b_SPEC12_namd	1.00	0.99	**1.00**	**1.00**	**1.00**	**1.00**	0.90	0.95	0.80	0.90	0.93	0.90
b_SPEC13_WPCcfd	0.94	0.96	0.85	0.90	0.98	**1.00**	0.92	0.93	0.93	0.85	0.90	**1.00**
b_SPEC14_poisson	0.98	0.94	**1.00**	**1.00**	**1.00**	0.95	0.87	0.93	0.85	0.93	0.90	0.75
b_SPEC15_python36	0.91	0.97	0.80	0.85	0.97	0.98	0.29	0.00	0.58	0.35	0.23	0.28
b_SPEC16_rodiniaLifeSci	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.79	0.83	0.75	0.83	0.78	0.75
b_SPEC17_rodiniaCFD	0.93	0.93	0.98	0.76	0.98	**1.00**	0.89	0.83	**1.00**	0.90	0.85	0.88
b_SPEC18_srmp	0.99	**1.00**	0.99	0.98	0.99	0.97	0.94	0.80	0.98	0.98	0.95	0.98

Figure 8.34: Comparison of % of Correct Predictions for benign operations, with 15s Start Time Results (from Figure 8.27) on the left, and visually identified actual operation start time shown on the right. Just with just 6 HPC features were required for the results on the left, while 9 HPC features were required to obtain the results on the right.

	15s Start Time						Operation Start Time					
	% Correct Predictions						% Correct Predictions					
	Overall	BT	SD	LSVM	KNN	NNN	Overall	BT	SD	LSVM	KNN	NNN
R_Babuk_1022	0.90	0.93	0.85	0.94	0.93	0.83	0.94	0.98	0.85	0.90	0.98	**1.00**
R_DarkSide_0521	0.93	0.93	0.83	0.90	**1.00**	0.99	0.73	0.78	0.68	0.58	0.85	0.78
R_DarkSide_Spr21	0.93	0.90	0.96	0.89	0.97	0.95	0.91	0.88	0.88	0.85	0.95	0.98
R_Gibberish_0422	0.98	0.95	**1.00**	**1.00**	0.98	0.96	0.88	0.80	0.90	0.90	0.88	0.93
R_HiddenTear_0422	0.86	0.84	0.58	0.98	0.98	0.93	0.84	0.85	0.73	0.90	0.93	0.78
R_Phobos_0122	0.95	0.96	0.98	**1.00**	0.86	0.96	0.88	0.90	0.83	0.85	0.95	0.88
R_Phobos_0522	0.95	0.97	0.97	0.97	0.93	0.93	0.82	0.83	0.75	0.78	0.90	0.83
R_Phobos_0922	0.91	0.93	0.93	0.98	0.83	0.89	0.84	0.83	0.83	0.83	0.88	0.83
R_REvilSodinokibi_Spr21	0.97	0.98	**1.00**	0.98	0.97	0.95	0.90	0.90	0.83	0.88	0.95	0.95
R_Snatch_0422	0.95	**1.00**	0.93	0.95	0.98	0.89	0.99	0.98	**1.00**	**1.00**	0.98	0.98
R_Snatch_0521	0.99	**1.00**	**1.00**	0.96	0.97	**1.00**	0.91	0.90	0.88	0.90	0.95	0.90
R_Sodinokibi_0222	0.88	0.90	0.52	**1.00**	**1.00**	0.97	1.00	**1.00**	0.98	**1.00**	**1.00**	**1.00**
R_Sodinokibi_0722	0.94	0.96	0.77	**1.00**	**1.00**	0.98	0.90	0.88	0.83	0.90	0.93	0.98
R_WannaCry_0622	0.89	**1.00**	0.45	**1.00**	**1.00**	0.98	0.49	0.70	0.13	0.35	0.78	0.50
R_WannaCry_Spr22	0.82	0.78	0.83	0.92	0.78	0.79	0.86	0.83	0.78	0.93	0.78	0.98

Figure 8.35: Comparison of % of Correct Predictions for ransomware operations, with 15s Start Time Results (from Figure 8.27) on the left, and visually identified actual operation start time shown on the right. Just with just 6 HPC features were required for the results on the left, while 9 HPC features were required to obtain the results on the right.

8.5 Top Performance Counter Feature Comparison

Unsurprisingly, the process of iteratively ranking and selecting performance counter features based on highest MCC value generated different results for the generic operation start time experiment (Section 8.3) and the actual operation start time experiment (Section 8.4), as both benign and ransomware operations experienced a time delay between intended and actual start time. This section introduces a third set of performance counter feature rankings and provides a comparison of top ranked performance counter features.

8.5.1 25 second Start Time Feature Set

To provide another set of rankings for comparison purposes, a third set of training features were generated with a generic 25 second start time, and the process in Section 8.2.2 was repeated to rank this 3rd scenario. This resulted in rankings from 3 different scenarios:

1. **15 second "Snapshot"**. This set of feature rankings was obtained while some operations were in their beginning stages while others had not yet started executing.

2. **Actual Operation Start**. This set of feature rankings suggests a set of performance counters that might be useful to identify the start of an operation.

3. **25 second "Snapshot"**. This set of feature rankings ensured virtually all trials were well underway, including the ransomware and SPEC operations with delayed starts.

8.5.2 Performance Counter Ranking Discussion

A listing of the top 3 ranked features for each of the three sets of rankings and each of the five classification algorithms tested is included in Figure 8.36. Hardware event count for the duration of the 30 second trial for the three performances counters that were ranked 1 or 2 repeatedly across the five classification models are shown in Figures 8.37-8.39, with the 20 benign operations displayed in the top 5 rows and the 15 ransomware operations displayed in the bottom 4 rows. The y-axis is the same for each set of operations per hardware counter, although the range varies for different hardware counter plots. For example, Figure 8.38 shows `L2_RQSTS.CODE_RD_MISS` event counts, or the raw number of L2 cache misses from fetching in-structions. In this case misses are on a comparable scale until ransomware execution begins, and then the event count increases significantly for most ransomware in comparison to staying level or dropping for most benign operations. In contrast, in Figure 8.39 (`L2_RQSTS.PF_HIT`), the y-axis ranges from 0 to 15 million hardware events in order to capture the large number requests from the L1/L2/L3 hardware prefetchers or Load software prefetches that hit L2 cache, in comparison to an event number so low (less than 500,000) that ransomware operations are not visible at that scale. This demonstrates that performance counter feature selection can be useful based not only on events that are typically indicative of ransomware, but also on events that occur less in ransomware than in other operations.

8.6 MASC Experiments Summary and Future Research Directions

In summary, Chapters 7 and 8 described a series of 2 proofs-of-concept and three experiments with increasing scope to explore the ability of micro-architectural-based side-channel

			15s Start Ranking				"Actual Start" Ranking				25s Start Ranking			
HPC	Description	Event counts...	BT	SD	LSVM	KNN	BT	SD	LSVM	KNN	BT	SD	LSVM	KNN
1	BR_INST_RETIRED.CONDITIONAL_PS	Conditional branch instructions retired using Precise Event Based Sampling (PEBS, which captures address of exact instruction that caused event).	-	-	-	-	-	-	-	-	-	-	3	
2	BR_INST_RETIRED.FAR_BRANCH_PS	Far branch instructions retired using Precise Event Based Sampling (PEBS, which captures address of exact instruction that caused event)	6	-	5	5	4	-	-	2	6			
7	BR_MISP_RETIRED.ALL_BRANCHES_PS	All mispredicted macro branch instructions retired using PEBS.	-	-	-	-	6	-	5	-	-	2		2
8	BR_MISP_RETIRED.CONDITIONAL_PS	Mispredicted conditional branch instructions retired using PEBS.	4	3	-	-	-	4	-	-	-	-	4	4
13	CPU_CLK_UNHALTED.REF_TSC	Number of reference cycles when the core is not in a halt state (i.e. running HLT or MWAIT inst) on a dedicated fixed counter; can approximate elapsed time while the core was not in a halt state.	-	-	-	-	-	-	-	-	-	3		
33	FRONTEND_RETIRED.L2_MISS_PS	Retired instructions who experienced Instruction L2 Cache true miss (using PEBS).	-	1	1	-	-	-	-	-	-			
37	FRONTEND_RETIRED.STLB_MISS_PS	Retired instructions that experienced STLB (Shared/Second-level Translation Lookaside Buffer, which stores recent translations of virtual to physical memory) true miss.	-	10	-	1	-	-	-	-	-			4
39	ICACHE_64B.IFTAG_HIT	Instruction fetch tag lookups that hit in the instruction cache (L1I). Counts at 64-byte cache-line granularity.	-	-	3	-	-	-	-	2				
41	ICACHE_64B.IFTAG_STALL	Cycles where a code fetch is stalled due to L1 instruction cache tag miss.	-	-	2	-	-	-	-	-				
42	IDQ.ALL_MITE_CYCLES_4_UOPS	Number of cycles 4 uops were delivered to the Instruction Decode Queue (IDQ) from the MITE (legacy decode pipeline) path.	-	-	-	6	-	-	-	-	5*		3	
45	IDQ.MS_MITE_UOPS	Counts the number of uops initiated by MITE and delivered to Instruction Decode Queue (IDQ) while the Microcode Sequencer (MS) is busy, including uops that may 'bypass' the IDQ.	-	-	-	-	2	-	-	-				
48	INST_RETIRED.ANY	Number of instructions retired from execution (or retirement of the last uOp for multiple uOp instructions). Counting continues during hardware interrupts, traps, and inside interrupt handlers. Note: Event is counted by a designated fixed counter, leaving the four (eight when Hyperthreading is disabled) programmable counters available for other events.	1	-	1	-	-	-	-	10*				
51	INT_MISC.RECOVERY_CYCLES	Core cycles the Resource allocator was stalled due to recovery from an earlier branch misprediction or machine clear event.	-	-	2	-	2	-	-	8*			4	
55	ITLB_MISSES.WALK_PENDING	1 per cycle for each PMH (Page Miss Handler) that is busy with a page walk for an instruction fetch request.	2	-	-	-	7	-	-	-				
62	L2_RQSTS.ALL_DEMAND_REFERENCES	Demand requests to L2 cache.	-	-	-	-	-	3	-	3				
63	L2_RQSTS.ALL_PF	Total number of requests from the L2 hardware prefetchers.	-	-	-	-	-	-	-	-		1		
64	L2_RQSTS.CODE_RD_MISS	L2 cache misses when fetching instructions.	5	4	-	-	-	1	1	1		2	1	
66	L2_RQSTS.MISS	All requests that miss L2 cache.	-	-	-	-	-	-	-	-		3		
67	L2_RQSTS.PF_HIT	Requests from the L1/L2/L3 hardware prefetchers or Load software prefetches that hit L2 cache.	-	8	-	-	1	-	8	1		1	2	1
68	L2_RQSTS.PF_MISS	Requests from the L1/L2/L3 hardware prefetchers or Load software prefetches that miss L2 cache.	-	-	-	3	-	-	-	-				
69	L2_RQSTS.REFERENCES	All L2 requests.	-	-	3	-	-	-	2	-				
75	MACHINE_CLEARS.MEMORY_ORDERING	Number of Memory Ordering Machine Clears detected, as a result of: (a) memory disambiguation, (b) external snoop, or (c) cross SMT-HW-thread snoop (stores) hitting load buffer.	7	7	4	-	3	-	-	-		5		
77	MEM_INST_RETIRED.ALL_LOADS_PS	All retired load instructions, using PEBS.	-	10	-	-	10*	-	-	3	-			
79	MEM_INST_RETIRED.SPLIT_STORES_PS	Retired store instructions that split across a cacheline boundary, using PEBS.	-	-	-	-	-	2	-	-				
81	MEM_LOAD_RETIRED.L2_HIT_PS	Retired load instructions with L2 cache hits as data sources.	3	-	-	-	-	-	-	-				
85	UOPS_DISPATCHED_PORT.PORT_7	Counts, on the per-thread basis, cycles during which at least one uop is dispatched from the Reservation Station (RS) to port 7.	-	-	-	-	-	8	3	5				2
90	UOPS_ISSUED.SLOW_LEA	Number of slow LEA (Load Effective Address) uops being allocated. A uop is generally considered SlowLEA if it has 3 sources (e.g. 2 sources + immediate) regardless if as a result of LEA instruction or not.	-	2	8	6	-	-	-	-				

Figure 8.36: Titles and descriptions of top 3 ranked performance counters for all combinations of classifiers at 15 seconds, 25 seconds, and actual operation start time.

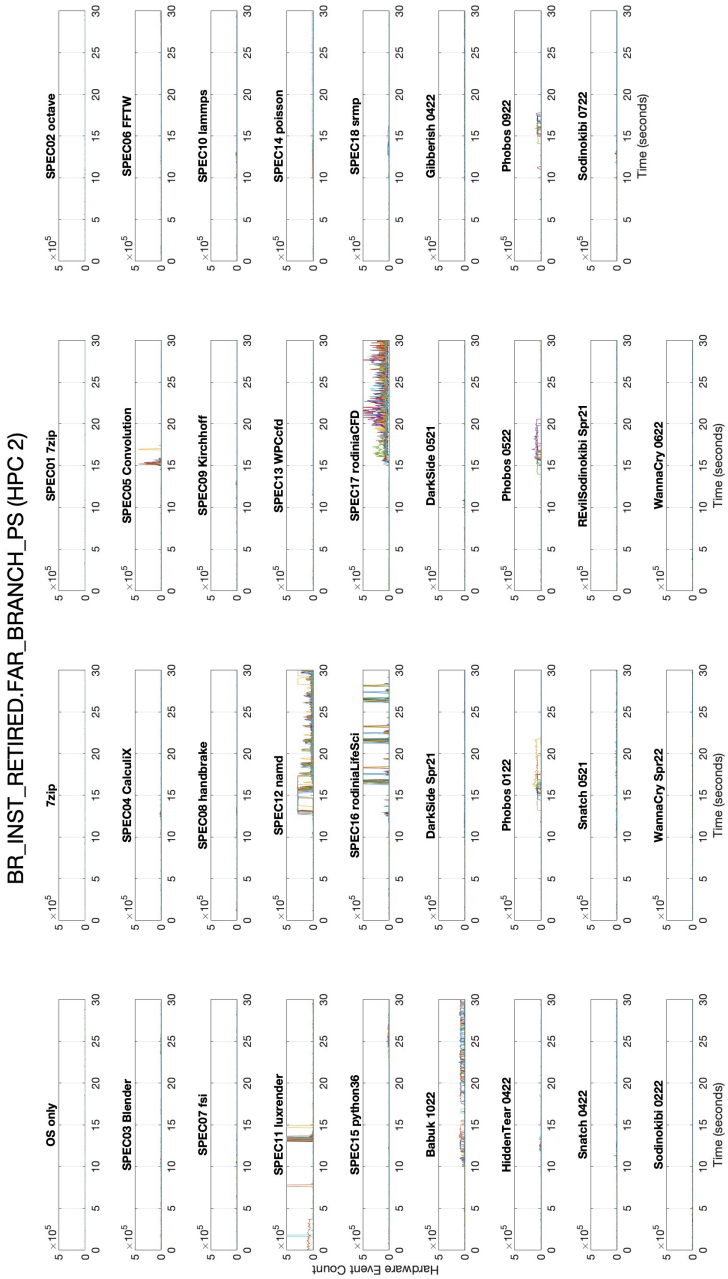


Figure 8.37: Summary of all training trials and all 35-classes of benign and ransomware operations for performance counter 2: `BR_INST_RETIRED.FAR_BRANCH_PS`. The 20 benign operations are displayed in the top 5 rows, while the 15 ransomware operations are displayed in the bottom 4 rows.

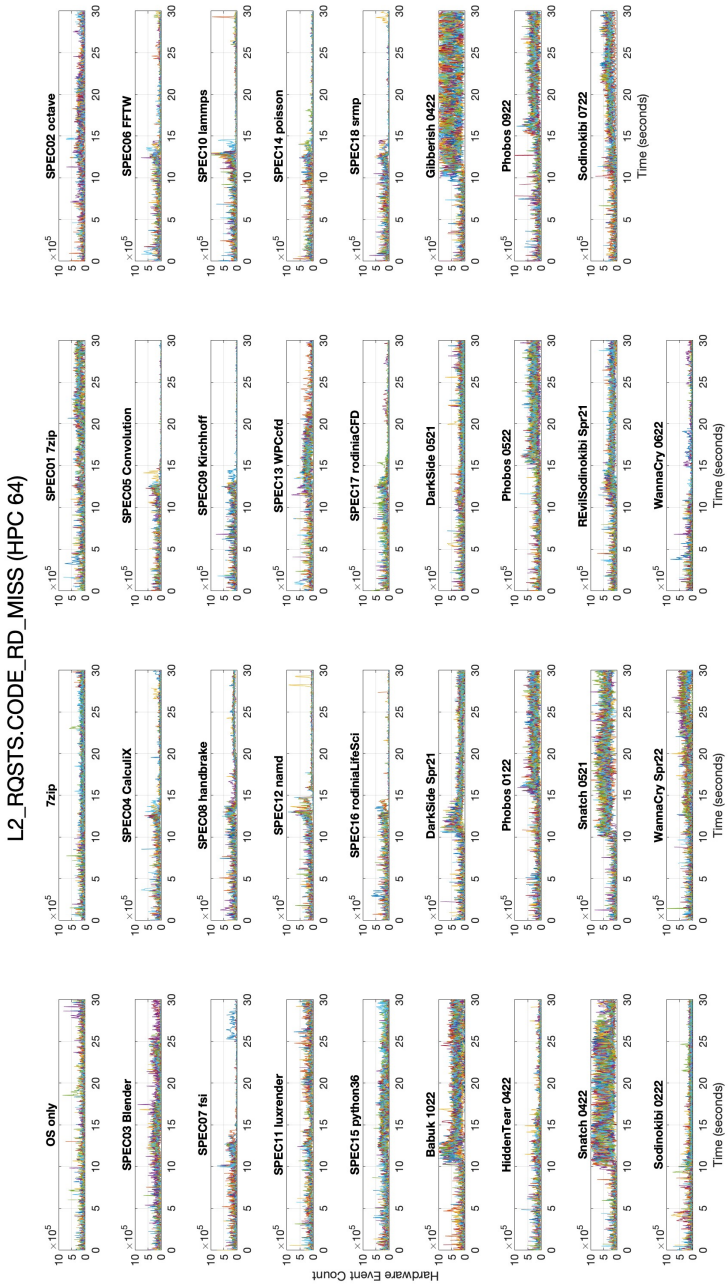


Figure 8.38: Summary of all training trials and all 35-classes of benign and ransomware operations for performance counter 64: `L2_RQSTS.CODE_RD_MISS`. The 20 benign operations are displayed in the top 5 rows, while the 15 ransomware operations are displayed in the bottom 4 rows.

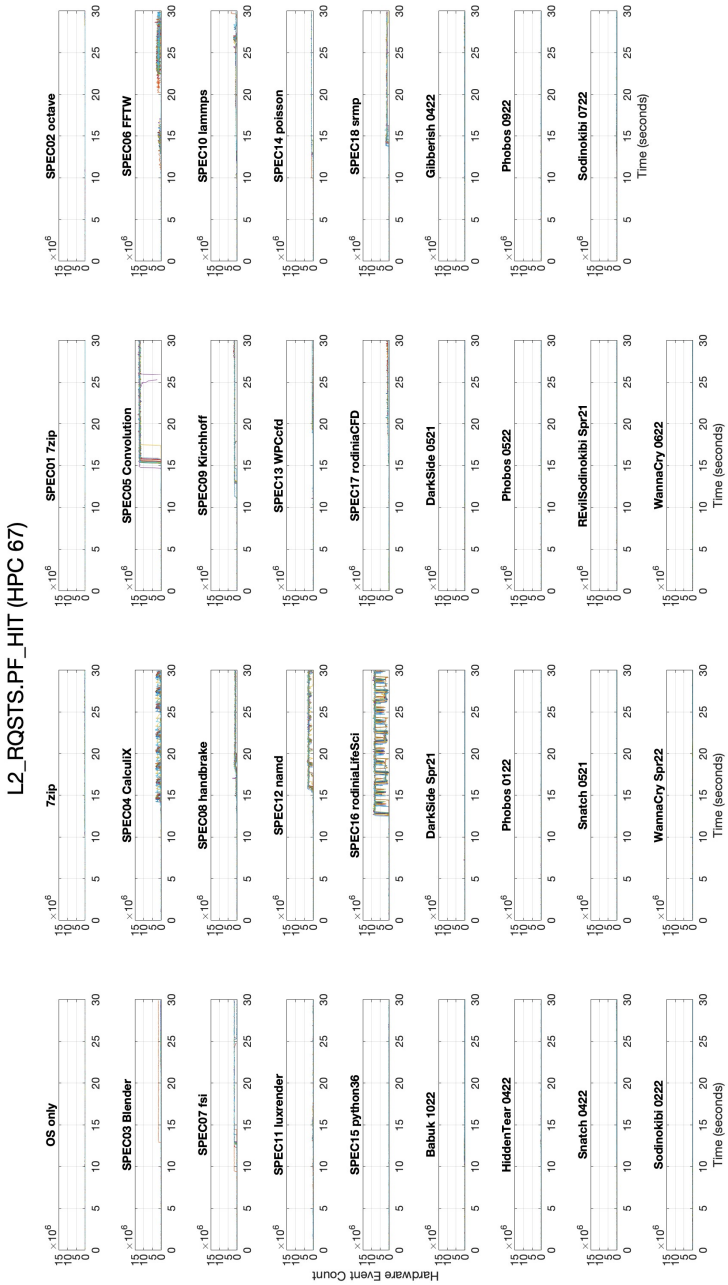


Figure 8.39: Summary of all training trials and all 35-classes of benign and ransomware operations for performance counter 67: `L2_RQSTS.PF_HIT`. The 20 benign operations are displayed in the top 5 rows, while the 15 ransomware operations are displayed in the bottom 4 rows.

analysis to distinguish ransomware operations from system idle, benign 7zip encryption, and standard SPEC benchmarks. The micro-architectural side-channel was accessed with hardware performance counters collected through Vtune Performance Analyzer, and over 230 counters were examined for their contribution to the accuracy, precision, and recall of classifier predictions. Five classifiers that resulted in the highest accuracy scores were selected for additional performance analysis, and the F1 score generated by each classification model trained with a single performance counter feature for a series of window durations from 0.1 to 2 seconds was used as a threshold to determine which performance counters warranted further consideration. The subset of 90 performance counters were again tested individually to evaluate overall classifier performance, as indicated by calculating Matthews Correlation Coefficient (MCC) generated by the individual counter. After a series of 5 repetitions, the counter that most frequently generated the highest overall MCC (averaged across all window sizes) for each classifier was identified. This process was repeated iteratively, adding one counter that most frequently generated the highest overall MCC with each iteration, and ultimately generating a ranking of the top 10 performance counter features for each classification model evaluated.

The process of iteratively training classifiers and selecting the feature that most frequently generated the highest MCC to obtain a top 10 performance counter ranking was repeated for 3 different phases of operation execution (15 s snapshot window, actual operation start, and 25 s snapshot window) for each trial, and the resulting rankings (Figure 8.36) were compared. Although related work collected a relatively small number of performance counters and provided limited ranking information, ranging from just 5 (`instructions`, `cache-references`, `cache-misses`, `branches`, `branch-misses` [196,197,203]) with no ranking information, to 11 ranked using negative correlation (Top 3, in order:(`cache-misses`, `task clock`,

`branches` [202], to some subset of 16 groups of performance counters, of which the `TLB_DATA` group stood out [198], some trends emerged. Many of the highest ranking counters involved instructions, caches, and branches, and the results obtained in this Chapter were similar, but with additional granularity and consideration of significantly more possible features. While SPEC benchmarks were chosen for their computational intensity, a drawback is that they may overemphasize certain performance counters due to that same computational intensity. Future work should include additional more typical benign operations. While this contributes additional imbalace to the data set, it can be mitigated by considering MCC as the true indicator of classifier performance.

Interestingly, while other works intentionally limited the number of counters collected to the number of dedicated registers, the results of Study 3 demonstrate only marginal improvement for 3 of 5 classifiers tested after 6 performance counter features. However, as all classification training and testing was performed with non-optimized models, with the assumption that results are likely to improve with the selection of a single model and optimization of its parameters, it is possible that additional features could improve the results of an optimized classification model.

Overall, Study 3 showed that classification of ransomware in a given window of time with over 90% accuracy is viable with just 4-6 HPC features for three of the five classifiers (Fine KNN, Narrow Neural Network, Bagged Tree Ensemble). For the 15 s start window, the Fine KNN classifier provided the best overall results, and the Narrow Neural Network classifier performance was comparable at the 1 and 2 second windows. "OS only" trials comprised the vast majority of benign operations incorrectly classified as ransomware, which is likely due to the generic training/testing on windows around operation trigger time rather than identifying the exact start time of each trial. For the actual operation start, 6-9 features were required to obtain $>$ 90%

test data, with the exception of the 0.1s window, which performed poorly regardless of algorithm or number of features. This is likely due to a combination of the averaging of overall MCC values with the other 3 (larger) windows tested, combined with human error in identifying exact operation start time manually for over 1000 trials. Future work could address this discrepancy by determining a method to automatically detect start time, as well as optimizing performance counters specifically for smaller windows of time. Results for "OS only" operations improved to 86% prediciton correctness overall (from 33%) for all 5 classifiers across all time windows, with only LSVM producing results $\leq$ 85%. This improvement came at a cost, however: The percentage of correct predictions for several `SPEC` Benchmarks decreased, with `Convolution` and `python36` dropping to 16% (from 98%) and 29% (from 91%) overall. This is very likely due to the choice to manually selecting an operation start time where the benchmark was beginning its most intense period of computations, rather than the beginning of the many second ramp up period.

Additional follow on work will develop the capability to collect a reduced set of the top ranked performance counters in real-time, and examine time series classification methods such as Long Short-term Memory Neural Networks, Time Series Forest/Random Interval Spectral Ensemble (RISE), and Shapelet-Based Classifiers to enhance classification results.

Chapter 9: Current-based Power Side-channel Analysis Experiments

This chapter details two sets of proof-of-concept studies followed by full experiments leveraging the Power Side-channel (PSC) accessed via current draw analysis. In both sets of experiments, the general side-channel analysis procedure involves collecting and verifying training data, generating power spectral density features for classifier training, and then testing trained classifiers with separate test data to evaluate classifier performance. Section 9.1 describes an initial exploration into the ability to differentiate read and write operations on SSDs with varying interfaces, form factors, and memory technologies. Section 9.2 advances that work for classification of read, write, and idle operations using current-draw measurements. In Section 9.3, a proof-of-concept evaluation of leveraging the PSC for ransomware classification is presented. Section 9.4 describes the analysis and results of current-draw PSC analysis for data collected in tandem with hardware performance counters. This chapter concludes with a summary of PSC results in Section 9.5.

9.1 Proof of Concept: Power Side-channel Analysis via Current Measurements for Detection of SSD File Operations

Since the majority of ransomware execution consists of encryption operations, initial work focused on verifying power side-channel analysis via current measurements to distinguish basic

(read/write) file operations on SSDs of varying form factors, interfaces, and memory technologies. This section describes the initial examination of the Intel Optane 900P, 905P, and H10 SSDs using current-draw-based side-channel analysis techniques. Power usage for the Intel Optane drives is monitored and compared with four industry-standard NAND Flash SSDs. Each drive was found to have a unique power signature, which is presented. Preliminary power analysis shows promising results towards future classification of read and write activity across the different memory devices.

9.1.1 Method

This section describes the hardware and software involved in data collection and analysis.

9.1.1.1 Hardware

The test setup is shown in Figure 9.1 and described in more detail in the following subsections.

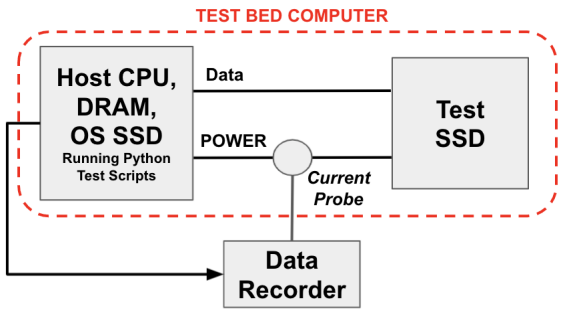


Figure 9.1: Experimental setup for monitoring power supplied to the test SSD from the host computer. Current is measured with a current probe, amplified, and acquired and stored with the data recorder.

Host Computer. The test bed computer was an Intel Core i5-4570 System with 8 GB DDR3 running Windows 10 Pro. The system largely reflected an off-the-shelf configuration and

included two drives, the first of which contained the operating system and all software required for testing. The second drive was the test SSD.

Test SSDs. Seven SSDs were tested, as shown in Table 9.1. The 2.5" Samsung and Crucial SATA drives were examined in previous work [211–213], and their newest M.2 PCIe counterparts were selected for comparison purposes, as all four drives use similar versions of 3D NAND technology. Two Optane-only drives were selected in both form factors, as well as the hybrid Optane H10, which uses 3DXP for caching and QLC NAND for storage. In this work, the H10 is used in its default mode with the 3DXP portion of the SSD deactivated.

Table 9.1: Solid State Drives Tested

	Crucial MX500	**Crucial P5**	**Samsung 850 EVO**	**Samsung 970 EVO Plus**	**Intel Optane 900P**	**Intel Optane 905P**	**Intel Optane H10**
Model	CT250MX-500SSD1	CT500P5-SSD8	MZ-75E250	MZ-V7S500	SSDPE21-D280GA	SSDPEL1-D380GAX1	HBRPEKNX-0202A01
Capacity	250 GB	500 GB	250 GB	500 GB	280 GB	380 GB	32GB + 512GB
Read Speed	560 MB/s	3400 MB/s	540 MB/s	3500 MB/s	2500 MB/s	2500 MB/s	2400 MB/s
Write Speed	510 MB/s	3000 MB/s	520 MB/s	3200 MB/s	2000 MB/s	2200 MB/s	1800 MB/s
Form Factor	2.5"	M.2 2280	2.5"	M.2 2280	2.5"	M.2 22110	M.2 2280
Interface	SATA III	PCIe 3.0 x4 NVMe	SATA III	PCIe 3.0 x4 NVMe	PCIe 3.0 x4 NVMe	PCIe 3.0 x4 NVMe	PCIe 3.0 x4 NVMe
Technology	3D NAND	3D TLC NAND	3D V-NAND	3D V-NAND TLC	3DXP	3DXP	3DXP and 3D NAND QLC

Test SSD Interface. In this work we test both PCIe and SATA drives. All PCIe drives were connected to the host computer by mounting to a PCIe adapter card which was connected to the motherboard through a PCIe extension cable. The Crucial MX500 and Samsung 850 EVO were connected to the host computer via the SATA III connector.

Current Probe. A Tektronix TCP312 current probe (Tektronix, Inc. Beaverton, Oregon, USA) with the Tektronix TCPA300 current probe amplifier was placed around the relevant power line(s) for each device (determined by experimental observations). The probe was placed around all five 12 V lines of the PCIe extension cable for all PCIe drives (all drives except for the Crucial MX500 and Samsung 850 EVO) and around the the 5 V line of the SATA III drives (Crucial

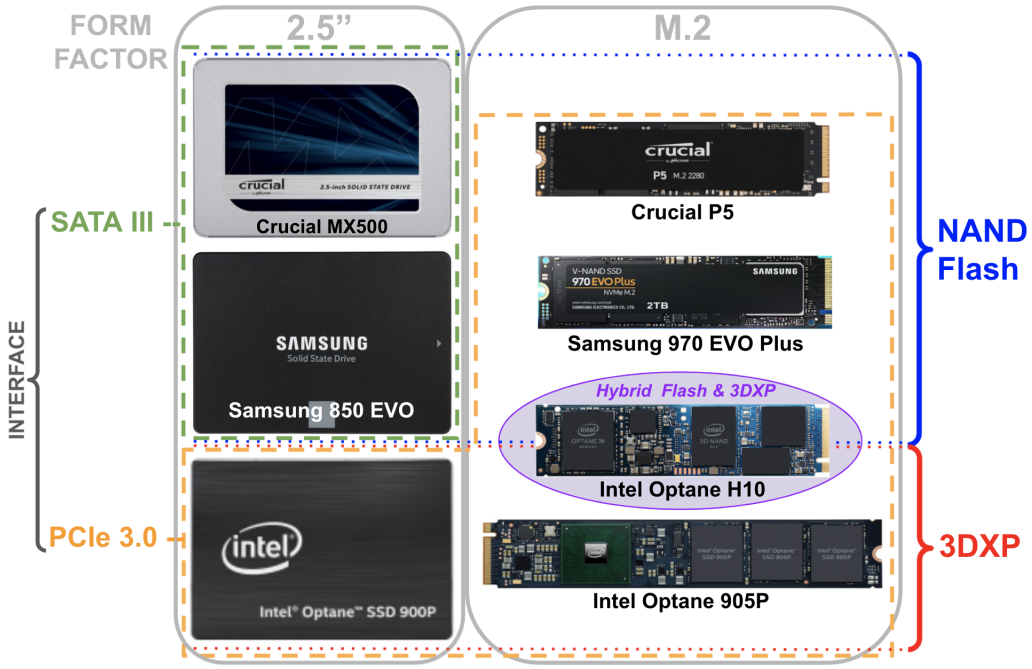


Figure 9.2: Seven SSDs were tested. Drives were selected to cover multiple form factors (2.5” SSD and M.2 module), interface types (SATA III and PCIe 3.0 x4), and technologies (3D NAND and 3DXP), as indicated above.

MX500 and Samsung 850 EVO).

Data Recorder. All data were collected by a GEN3i High-Speed Data recorder with a GN412 High-Speed 100 MSps Data Acquisition Card (HBM, Inc. Darmstadt, Germany) at a rate of 200 kSps with 14 bits per sample. Previous work validated that this sampling rate and resolution allowed adequate sampling to capture fluctuations in SSD current important to identifying reads and writes while maintaining a resolution of less than 1 mA based upon a 10 A range [213].

9.1.1.2 Data Collection Procedure.

The data collection process was fully automated through the use of a Python test script. The script generated files of random bytes at sizes of 100, 250, 500, 750, and 1000 MB, which

were used for all drives. The recorder was triggered from the serial port of the host computer to record approximately 60 seconds of data, with an approximate delay of 2 s before the operation was triggered to allow for collection of a baseline quiescent period. Fifteen trials were collected at each file size. A flow chart for the script is presented in Figure 9.3.

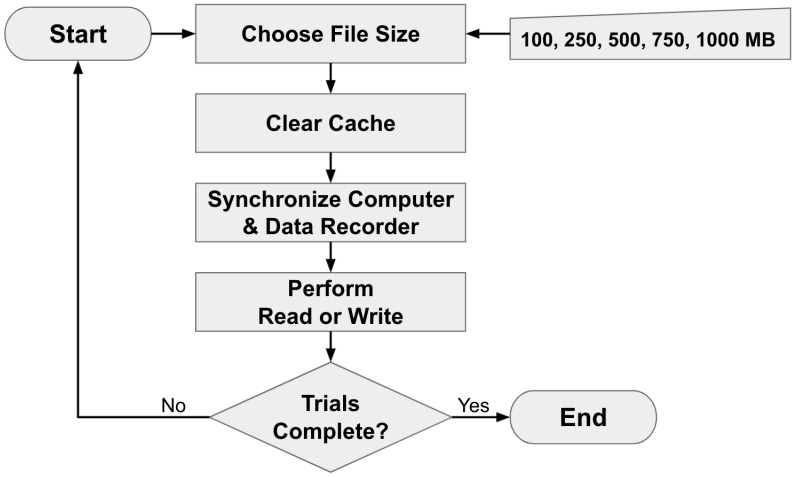


Figure 9.3: Data Collection Flow Chart

9.1.1.3 Data Analysis

Based on prior work demonstrating that high amplitude spike and pulse-like activity characterizes read and write SSD operations [211–213], the average power of the longest detected pulse was used as a figure of merit for comparing across experimental conditions (drive–file-size–operation combinations). Pulses were defined and detected using an automated procedure, as follows: First, to normalize against potential baseline wander across trials, we subtracted the mean of the quiescent period between 0 and 1.5 s from each trial's recording. We then squared the current signal to achieve a representation proportional to its instantaneous power. The 95th percentile of the empirical distribution of this squared signal was used as the pulse detection

threshold for each trial. The squared signal was smoothed with a symmetric, sliding window median filter (0.001 s window for write trials, 0.1 s window for read trials) and upward and downward threshold crossings were used to define and delimit distinct pulses. The RMS current of the pulse with the longest duration was then multiplied by the nominal drive supply voltage to compute the average power (W), which we used as a simple univariate measure to describe each trial.

9.1.2 Results

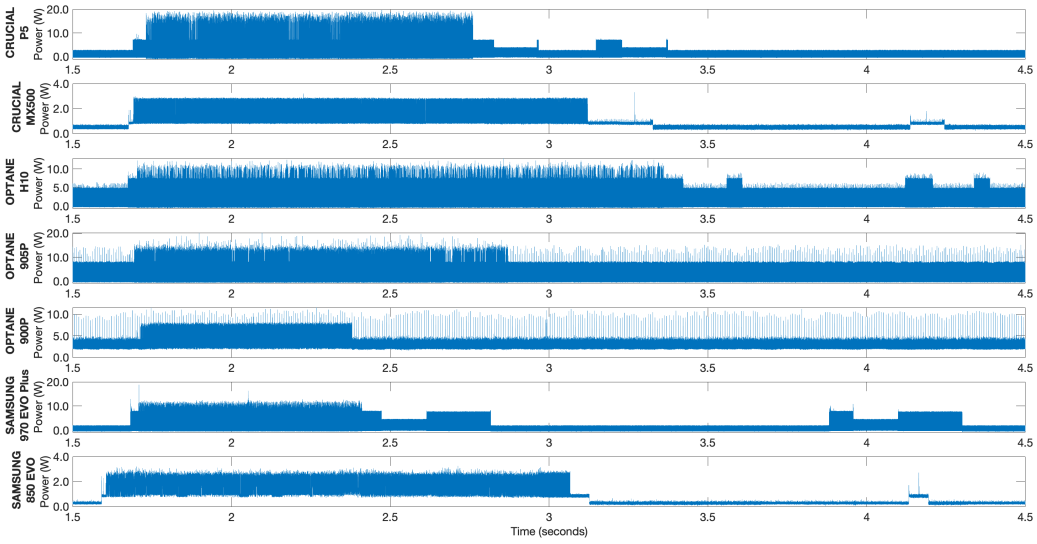


Figure 9.4: Representative Reads of 500 MB files for all Crucial, Optane, and Samsung SSDs tested. Fifteen trials were conducted at each of 5 file sizes. Power is provided through the 12V line to all PCIe drives (all Optane drives, the Crucial P5, and the Samsung 970 EVO Plus), and through the 5V line for the SATA III SSDs (Crucial MX500 and Samsung 850 EVO). Note: Y-axis for each drive is scaled to show detailed signature and should not be used for direct power comparison.

Representative raw data power signatures of each drive for reads (Figure 9.4) and writes (Figure 9.5) at 500 MB are presented in this section, as well as the longest pulse power (Figure

9.6), as described in Section 9.1.1.

9.1.2.1 Power Signatures

Each drive type was found to have a unique power signature for reads and writes. Figures 9.4 and 9.5 show representative power traces for each drive for 500 MB reads and writes, respectively.

The flash-based Samsung and Crucial PCIe SSDs exhibit similar power signatures and durations of read and write operations (shown in the 1st and 6th rows of Figures 9.4 and 9.5), as is the case for the SATA versions of the Samsung and Crucial drives (rows 2 and 7 of Figures 9.4 and 9.5).

The center rows (3, 4, 5) of both figures depict power signatures of the three Intel Optane SSDs. All three signatures contain more high frequency background noise than their NAND flash counterparts, with the read activity for the Intel Optane 905P (4th row of Figure 9.4) somewhat obscured by the background noise, and the 900P (5th row of Figure 9.4) falling completely below the background noise level. The read activity of the Intel Optane 900P SSD is also shorter in duration than the other drives, while the 905P is comparable to the M.2 PCIe Crucial (1st row) and Samsung (6th row) SSDs.

All read operations across drives and file sizes consistently demonstrated increasing read duration as file size increased, as expected. As shown in Figure 9.5, write operations showed less consistent results across all drives; the same was true across file sizes. Similar to prior work [211], the majority of the write samples for all drives included a trail of pulses which may be related to the file system or operating system in general. Further investigation is required to

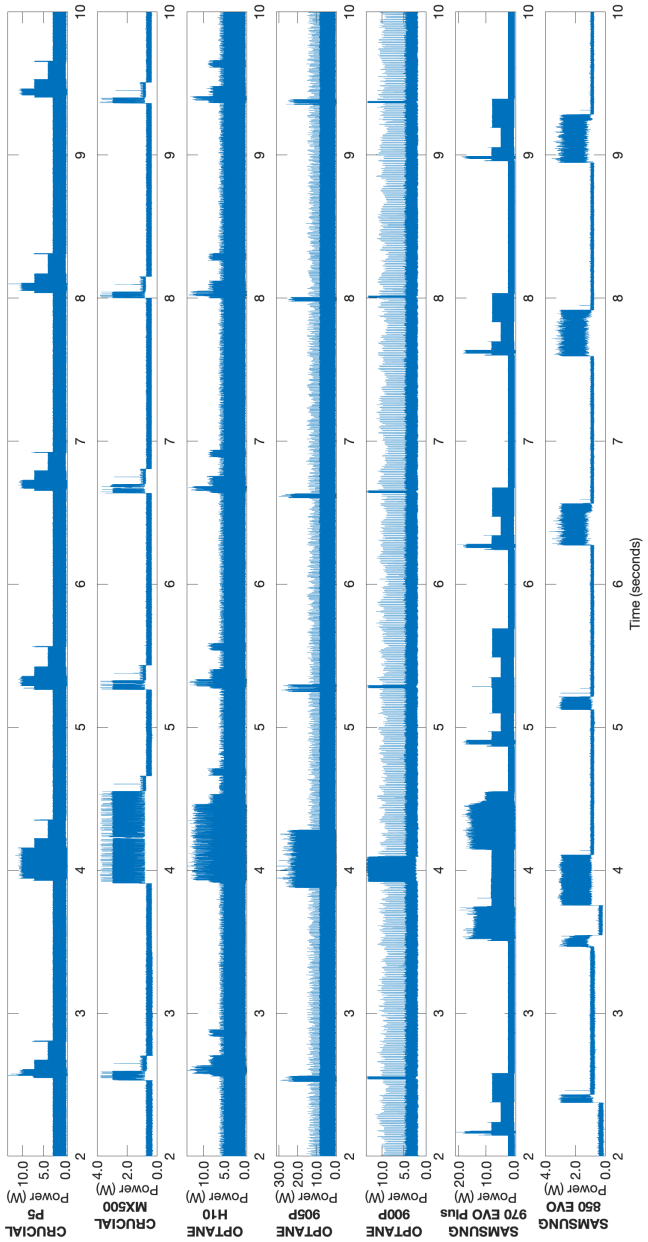


Figure 9.5: Representative Writes of 500 MB files for all Crucial, Optane, and Samsung SSDs tested. Fifteen trials were conducted at each of 5 file sizes. Power is provided through the 12V line to all PCIe drives (all Optane drives, the Crucial P5, and the Samsung 970 EVO Plus), and through the 5V line for the SATA III SSDs (Crucial MX500 and Samsung 850 EVO). Note: Y-axis for each drive is scaled to show detailed signature and should not be used for direct power comparison.

fully understand if or how this trail is specifically related to the write activity.

9.1.2.2 Classification Potential

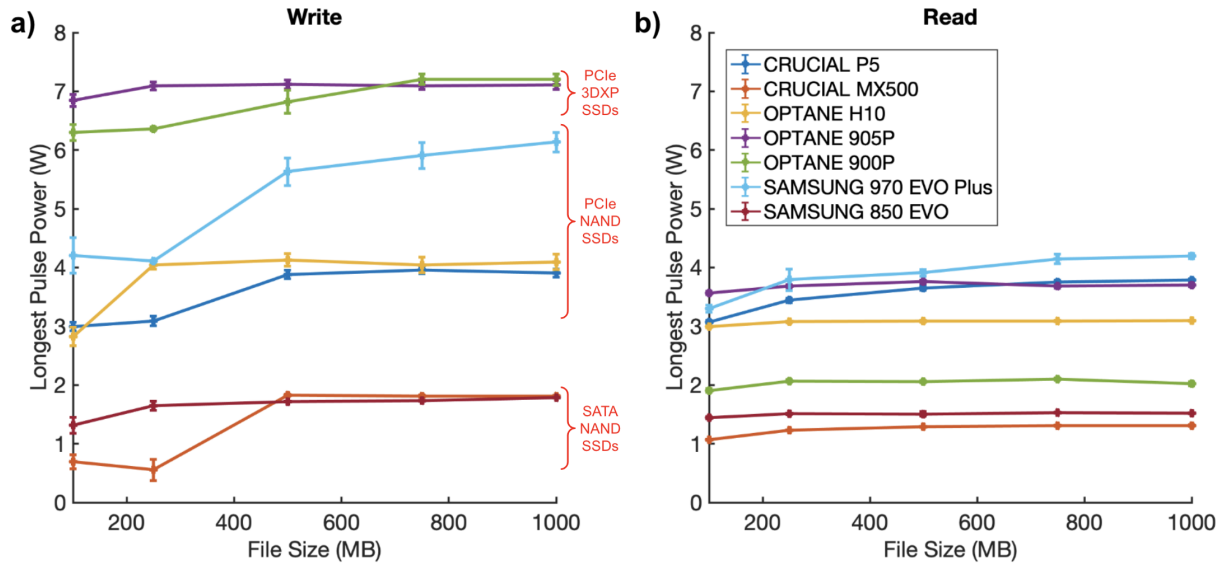


Figure 9.6: Longest pulse average power for (a) writes and (b) reads as a function of file size. Values plotted are averages across trials (n = 15). Error bars are +/- 2 standard errors.

Average power of the longest activity pulse, as shown in Figure 9.6, indicates the potential to distinguish drives based on read activity at each file size, in addition to being able to classify reads vs. writes within drives at each file size for the 3DXP and PCIe NAND drives, using this single feature alone.

A significant difference between the high power writes of the Intel Optane 3DXP SSDs and the lower power writes of the Crucial and Samsung flash SSDs can be seen in Figure 9.6a. The seven SSDs tested can be grouped by technology, as shown on the right side of the plot, with distinct separation between the groups. PCIe 3DXP SSDs require the highest write power, while PCIe NAND SSDs use lower write power, and SATA SSDs have the lowest write power requirements, as measured by the longest pulse.

Using longest pulse power to discriminate drives without prior knowledge of the file size also appears feasible. Although the cluster of three overlapping lines at the top of Figure 9.6b indicates that it might be difficult to distinguish the Samsung 970 EVO Plus, Optane 905P, and Crucial P5 drives based upon read power alone, using a combination of write and read power appears to be a promising way of discriminating these and all other drives irrespective of file size.

For example, an SSD with an average longest read pulse power of 2 W could be identified as an Optane 900P drive in Figure 9.6b without knowing file size, but a drive with a read power of 3.5 W might be a Samsung 970 EVO Plus, Optane 905P, or Crucial P5. The drive with the 3.5 W read power could be further refined based on the write pulse average power (Figure 9.6a), where a value close to 7 W would indicate the Optane 905P, while a value in the 4-6 W range would indicate the Samsung 970 EVO Plus, and the 3-4 W range would indicate the Crucial P5.

In Figures 9.4-9.6, the Optane H10, which uses a combination of 3DXP for caching and QLC NAND for storage, behaves similarly to the M.2 PCIe NAND drives, as the 3DXP portion of the SSD was not activated. Further data collection and investigation is required to determine the average power of the longest pulse characteristics when the H10 is operated using 3DXP and NAND flash technology.

9.1.3 Follow-on Power Side-channel Analysis Work

The longest pulse power metric derived from the time-domain power signatures is an encouraging preliminary indicator that classification of operations across different SSDs is possible. In this work, we used power-based side-channel analysis to compare seven different solid state

drives across different technologies (PCM and NAND Flash), interfaces (SATA III and PCIe 3.0 w/ NVMe), and form factors (2.5" and M.2), and presented time domain power signatures for both read and write operations. We examined the average power associated with the longest pulse for each of these drives and operations, compared them, and highlighted differences across drives of various technologies and interfaces. In doing so, we verified our current draw analysis methods with new SSD technologies, interfaces, and form factors, as a starting point for comparison with temperature side-channel measurements.

9.2 Study: Classification of Read, Write, and Idle Operations with Current-based Power Side-channel Analysis

This study leveraged the Section 9.1 results to focus on the newest drives and extend the experiment to include classification.

9.2.1 Experimental Setup

This section describes the experimental setup used to complete the current-based side channel data collection on the four SSDs included in this study.

9.2.1.1 Hardware

The hardware set up for this experiment is nearly identical to Section 9.1.1.1, but only four SSDs were tested, and the collection procedure was adjusted to facilitate randomization of the trial order. The SSDs tested are shown in Table 9.2, and were selected to enable further exploration of newer commercially available SSDs with like interfaces and form factors. The

data collection procedure for this experiment is described in Section 9.2.1.2.

Table 9.2: Solid State Drives Tested

	Crucial P5	**Samsung 970 EVO Plus**	**Western Digital Black SN750**	**Intel Optane 905P**
Model	CT500P5-SSD8	MZ-V7S500	SSDPE21-D280GA	SSDPEL1-D380GAX1
Technology	3D TLC NAND	3D TLC NAND	3D NAND	3DXP
Firmware Version	R311	2B2QEXM7	00SJ10WD	1D380480
Capacity	500 GB	500 GB	500 GB	380 GB
Read Speed	3400 MB/s	3500 MB/s	3430 MB/s	2500 MB/s
Write Speed	3000 MB/s	3200 MB/s	2600 MB/s	2200 MB/s
Form Factor	M.2 2280	M.2 2280	M.2 2280	M.2 22110
Interface	PCIe 3.0 x4 NVMe	PCIe 3.0 x4 NVMe	PCIe 3.0 x4 NVMe	PCIe 3.0 x4 NVMe

9.2.1.2 Data Collection Procedure

The order of data collection for all possible combinations of the four SSDs, three file operations, and three file sizes was fully randomized. For each trial, the test drive was identified and connected to the host computer. The randomly indicated file operation and size was fed to a Python test script at the command line, which generated files of random bytes at sizes of 100, 500, or 1000 MB, and were used for all drives. The test script triggered the data recorder from the serial port of the host computer to record approximately 30 seconds of data, with an approximate delay of 2 s before the operation was triggered to allow for collection of a baseline quiescent period. Twenty trials were collected for each SSD, file operation, and file size combination, totalling 560 trials. The host computer was powered down before changing out the SSD for the next trial. A flow chart for the randomized collection process is presented in Figure 9.3. Representative time

domain and spectral plots for read, write, and idle for each of the four drives are shown in Figures 9.8, 9.9, and 9.10, respectively.

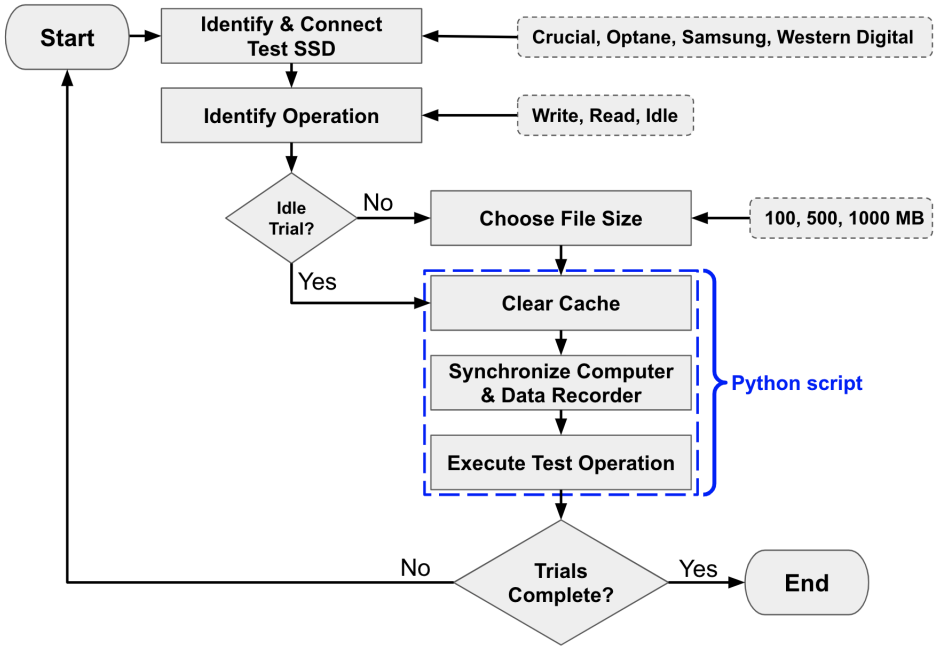


Figure 9.7: Data Flow for collection of read, write, and idle operations on four SSDs under test.

9.2.2 Analysis

In this section, we discuss the analysis techniques we employ on the data collected during this work. The section opens by describing the features of the SSD current draw signal that are used to distinguish between firmware version groupings and closes with a discussion of the classification algorithms utilized.

9.2.2.1 Frequency Domain-based Feature Generation

Observations to be classified were generated by first extracting the initial 28 seconds of each 30-second recording of SSD current. (The final two seconds were trimmed a priori to minimize

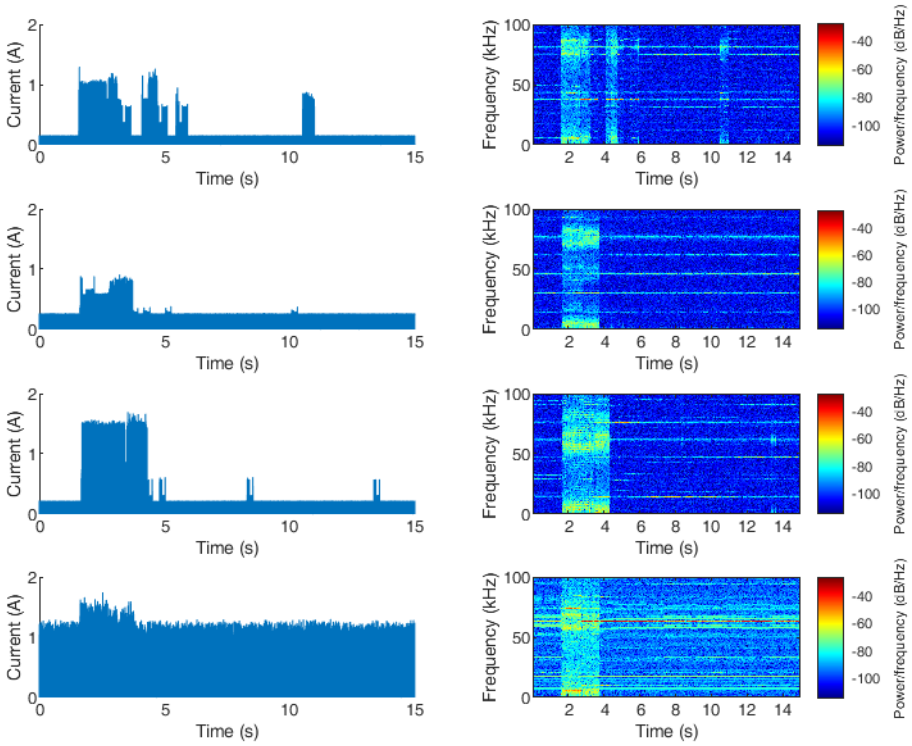


Figure 9.8: Representative time domain (left) and spectrogram (right) plots for 1 GB read operations for the Samsung, Western Digital, Crucial, and Optane drives, respectively (top to bottom). (Values smaller than -115 dB in the spectrogram plots are thresholded to -115 dB to facilitate visualization by improving image contrast.).

the influence of potential noise transients at the end of a file.) A description of the subsequent feature extraction procedure follows.

Each 28 s segment of SSD current activity was transformed to the frequency domain by using Welch's modified periodogram method [214] (7 s Hamming-windowed sub-segments with 3.5 s of overlap; 2^{21}-point Discrete Fourier Transforms) to estimate the power spectral density (PSD) of the signal. The PSD estimate was then integrated in non-overlapping frequency bins of 25 Hz width, spanning 0 to 5 kHz, and converted to the decibel scale, yielding a 1 x 200 preliminary feature vector for each observation.

Prior to classifier training, Principal Components Analysis (PCA) was applied to reduce

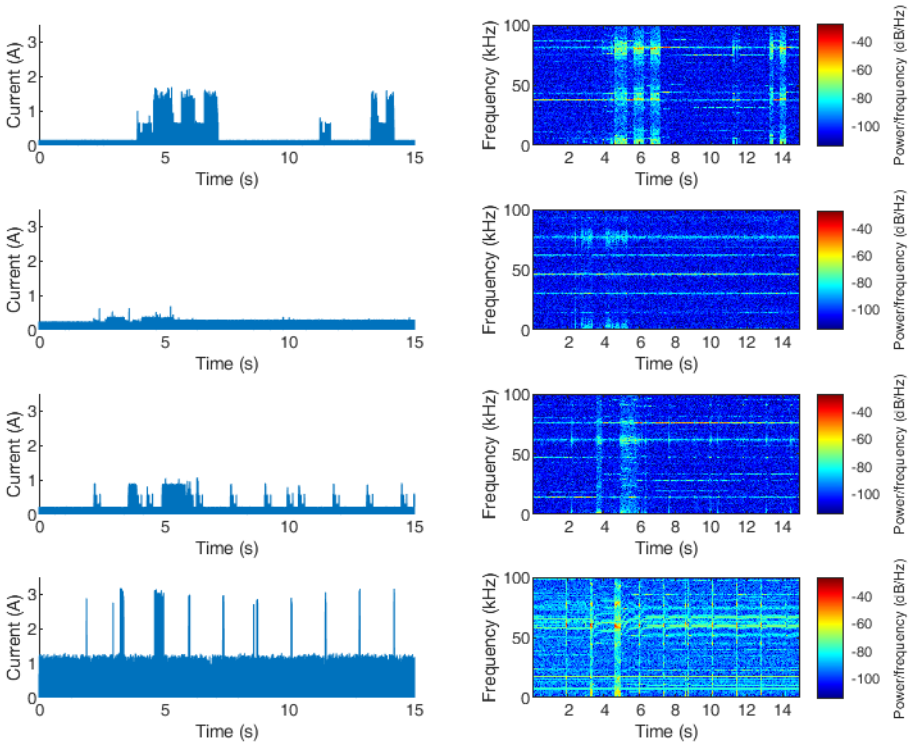


Figure 9.9: Representative time domain (left) and spectrogram (right) plots for 1 GB write operations for the Samsung, Western Digital, Crucial, and Optane drives, respectively (top to bottom). (Values smaller than -115 dB in the spectrogram plots are thresholded to -115 dB to facilitate visualization by improving image contrast.).

the dimensionality of the feature set. Specifically, we retained the minimum number of principal components required to account for at least 95% of the data variance. The reduced feature set (∼10 features, depending on the cross-validation fold; see Section 9.2.2.2 for classification details) was used in training. Test data were mapped to the principal component space of the training data before being classified.

9.2.2.2 Classification

Two different classification experiments were conducted, each involving training and testing 28-way Random Forest classifiers. Random Forests are among the many model classes capable

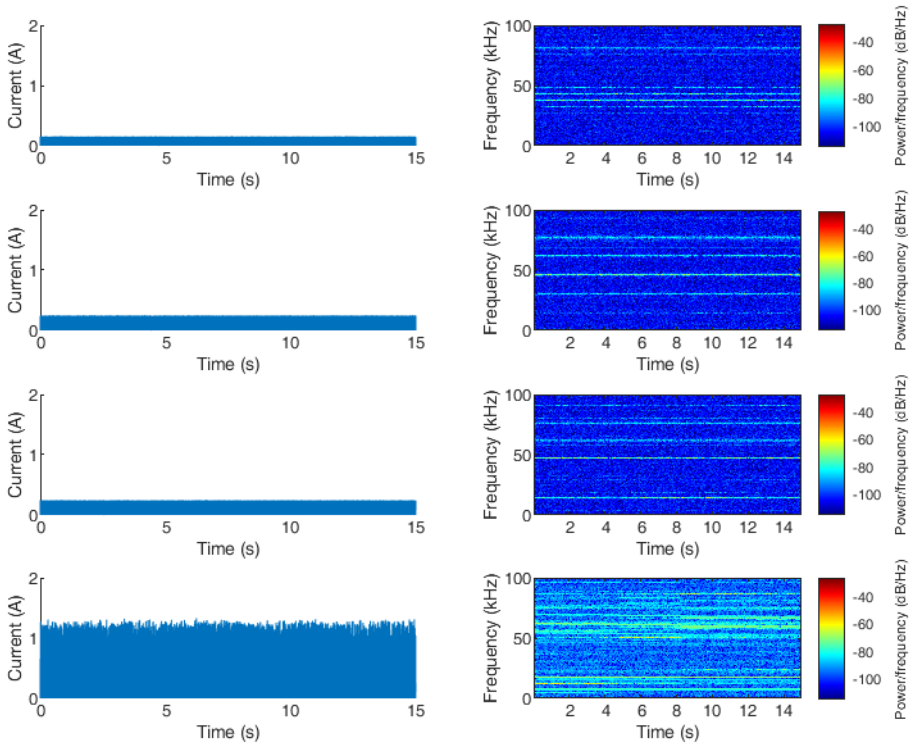


Figure 9.10: Representative time domain (left) and spectrogram (right) plots for the idle state for the Samsung, Western Digital, Crucial, and Optane drives, respectively (top to bottom). (Values smaller than -115 dB in the spectrogram plots are thresholded to -115 dB to facilitate visualization by improving image contrast.).

of representing nonlinear decision boundaries, and were chosen due to their success in related problems. The 28 classes were *idle*, *read*, and *write* operations for each of four SSDs (i.e., drives from four different manufacturers), with three different file sizes - 100 MB, 500 MB, and 1000 MB - for both *read* and *write* operations.

In the first classification experiment, we use stratified 10-fold cross-validation to estimate the classifier's capacity to generalize to unseen data collected *from the same physical set of drives* used in training, and *in relative temporal proximity* (e.g., within weeks) to collection of the training set. The characterization of proximity as weeks simply reflects the time span over which the training set itself was collected.

In the second experiment, we investigate the classifier's ability to generalize even more broadly. Specifically, we use a hold-out set approach to estimate performance on unseen data collected from: 1) *the same physical drives used in training, but at a relatively distant point in time*, e.g., ∼one year later; and 2) *a different physical set of drives* than used in training. The entire data set from the first classification experiment was used to train a single 28-way Random Forest classifier. This classifier was then evaluated on a test set collected approximately one year after the training set. The test set was composed of an approximately equal balance between new data from the physically identical four drives used in training and data from four physically different drives of the same type used in training.

9.2.3 Results

In this section, we discuss our results. Consistent with the findings of prior work [26], we focus on three modes of operation for the SSD: (1) a low power *idle* state, (2) a *read* state, and (3) a *write* state. The *idle* state is defined as the state in which no user-initiated read or write operation is being executed.

We begin by using data from all three of these states to train a classifier and examine its performance. We then improve the performance of the classifier by reducing the training data set to just the read operations. In both cases, we examine the ability of the classifier to classify by drive model as well as, where appropriate, drive operation and operation file size. We also examine the ability to classify between 3D NAND and 3D XPoint (PCM) technologies. In all cases, we are training the classifier on data collected over a year prior to the test data and the test data is comprised of samples from both the original drive used to create the training set (a

year earlier) as well as a second drive previously unseen by the classifier. For clarity, the data collected from the first drive in 2021 (and used for training) is identified as the *2021 Drive 1* data set while the test data collected from both drives in 2022 is identified as the *2022 Drive 1* data set and the *2022 Drive 2* data set, respectively.

This section also identifies and discusses the potential presence of wear and other drive anomalies and their effect on the performance of the classifier. It is worth explicitly highlighting that the use of our side-channel technique provided us initial visibility on the issues which were subsequently confirmed through the use of additional drive health assessment tools. In addition, it leads to the finding that any dips in classifier performance can be correlated to specific observable behavior found within the recorded time domain current measurements.

9.2.3.1 Classification using Read and Write operations as well as the Idle state

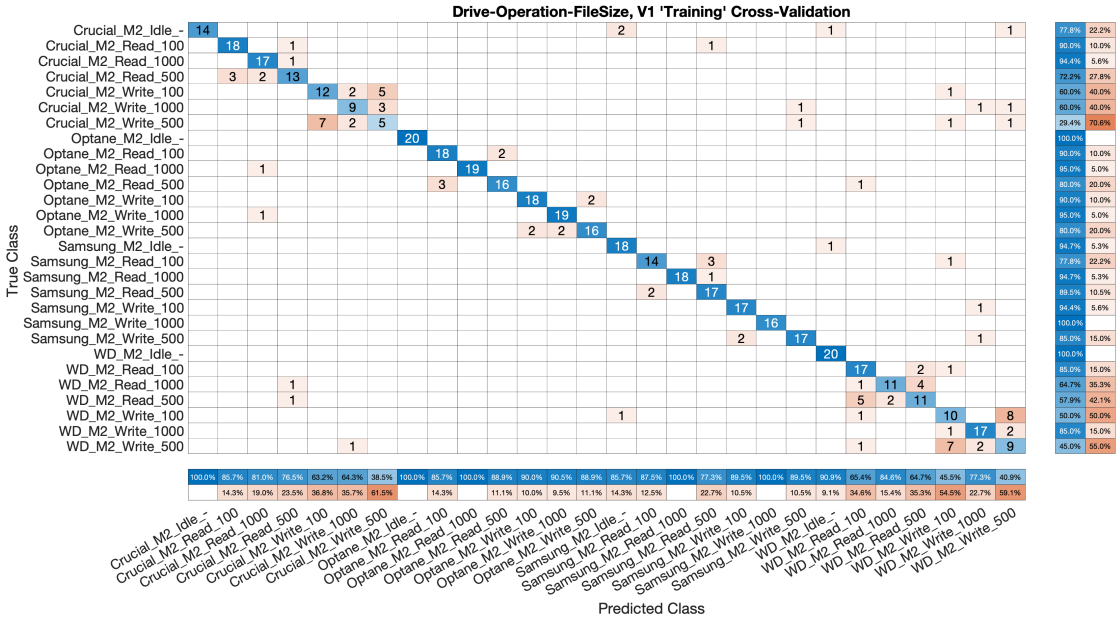
The 10-fold stratified cross-validation matrix using the training data and the classification confusion matrix for the full test set are provided in Figure 9.11. The overall accuracy (the percentage of correct predictions) in the cross-validation of the training set in Figure 9.11(a) is 80.2% while overall accuracy for the test data in Figure 9.11(b) is 73.6%. The recall (a measure of the ability of a classifier to find *all the relevant cases* of a specific class) can be seen to vary in the test set classification between 29.4% and 100%. The precision (a measure of the ability of a classifier to identify *only the relevant cases* of a specific class) can be seen to vary in the test set classification between 38.5% and 100%. Classification results are also shown for the individual 2022 data sets in Figure 9.12. Accuracy for the *2022 Drive 1* data set is 78.2% and accuracy for the *2022 Drive 2* data set is 68.9%. For comparison, chance in the 28-way classifier

of Figure 9.11(b) using the full 2022 test set is 3.6% and, thus, the results are significantly better than chance.

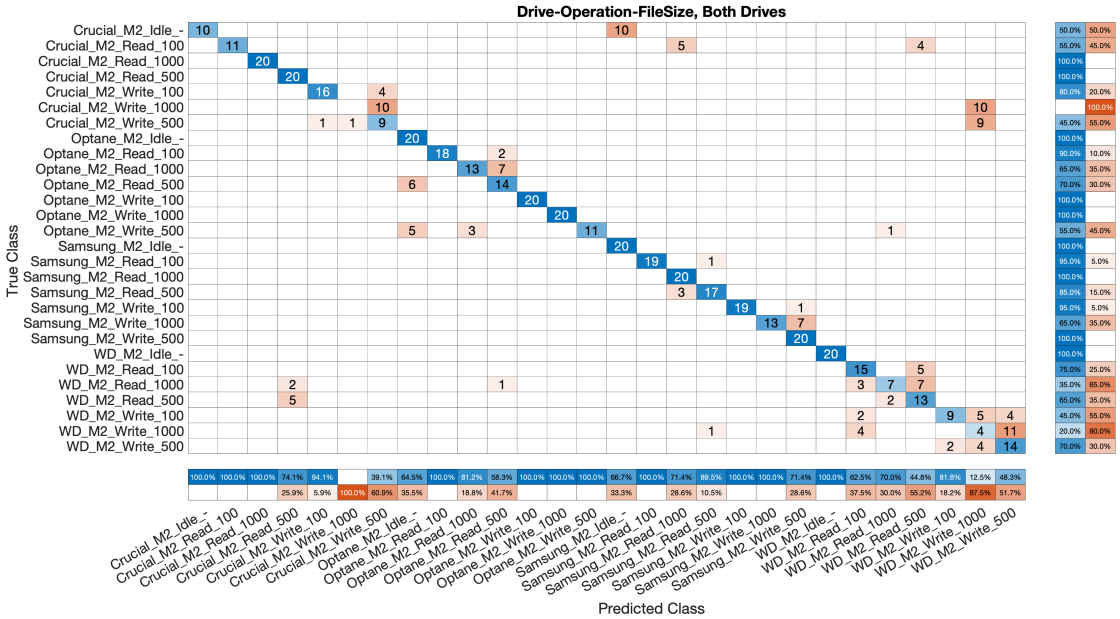
To handle the use case where we are concerned about unauthorized operations on the drive, we would be interested in classifying by operation only. In other words, can we tell which operations are occurring (read or writes) using our current-based side-channel analysis? When we group our classification results by operation, we find the overall accuracy of the classifier for the test data in Figure 9.13 is 96.1%. The recall can be seen to be 93% or better for each of the the three operations and the precision is 87.9% or better. This is in comparison to the 42.9% figure associated with the best performing chance model for this specific 3-way classifier. Classification results are also shown for the individual 2022 data sets in Figure 9.14.

To handle the second use case where we are interested in discovering counterfeit drives, the classification should be accomplished by drive model. When we group our classification results by drive model, we find the overall accuracy of the classifier for the test data in Figure 9.15 is 91.4%. The average recall is 91.5% and can be seen to be 72.9% or better for each of the the four drives while the average precision is 91.8% and is 84.5% or better for each individual drive. This is in comparison to the 25.0% figure associated with chance in this 4-way classifier. Classification results are also shown for the individual 2022 data sets in Figure 9.16. It can be seen that performance for the Crucial drive is dramatically poorer for the *2022 Drive 2* data set. Discussion and analysis of these results will left as future work.

For completeness, we also include the classification results by drive and operation in Figures 9.17 and 9.18. The overall accuracy for the test data in Figure 9.17(a) is 87.9%. The recall can be seen to vary in the test set classification between 50.0% and 100% while the precision can be seen to vary in the test set classification between 64.5% and 100%. Classification results are also

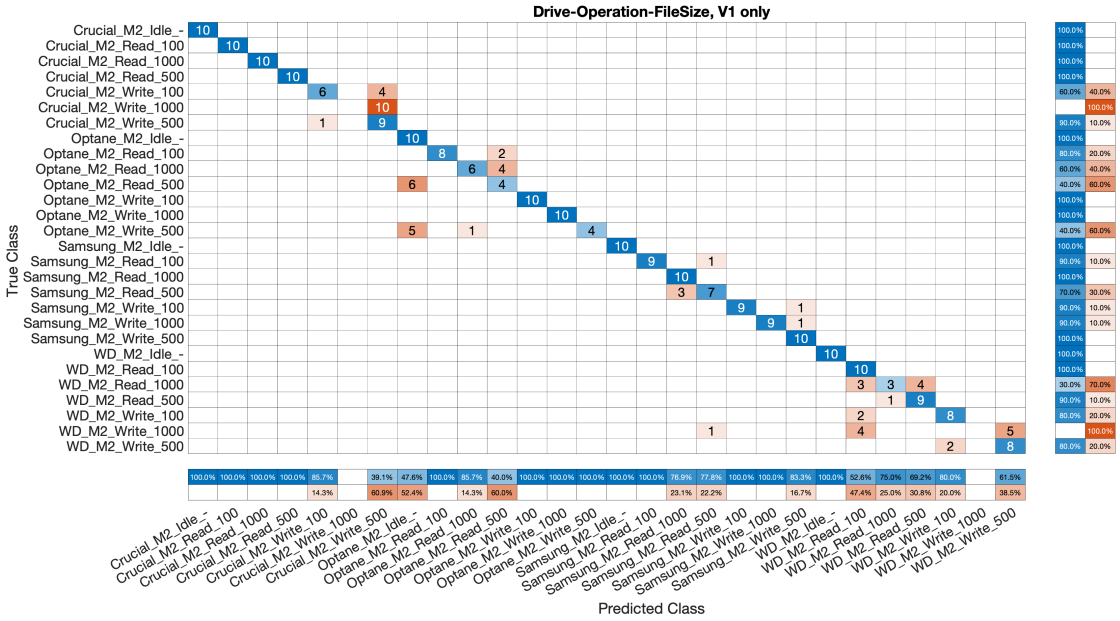


(a) Cross-validation results: Drive/Operation/File size

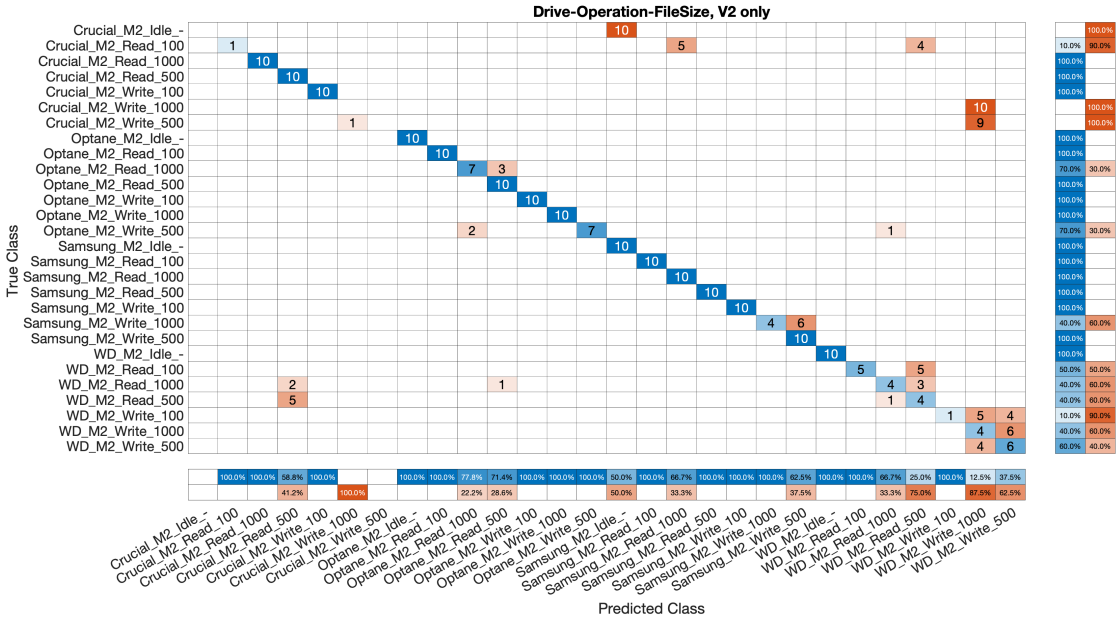


(b) Classification results, V1 and V2: Drive/Operation/File size

Figure 9.11: Classification results: Drive/Operation/File size



(a) Full classification results for *2022 Drive 1* data



(b) Full classification results for *2022 Drive 2* data

Figure 9.12: Full classification results for the individual test data sets: *2022 Drive 1* data and *2022 Drive 2* data.

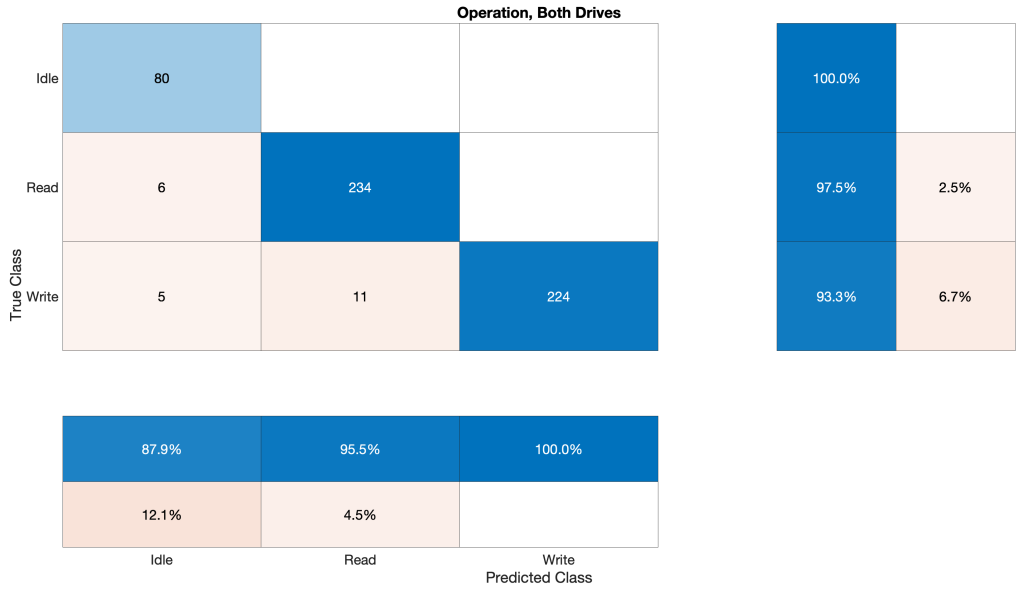


Figure 9.13: Classification by operation.

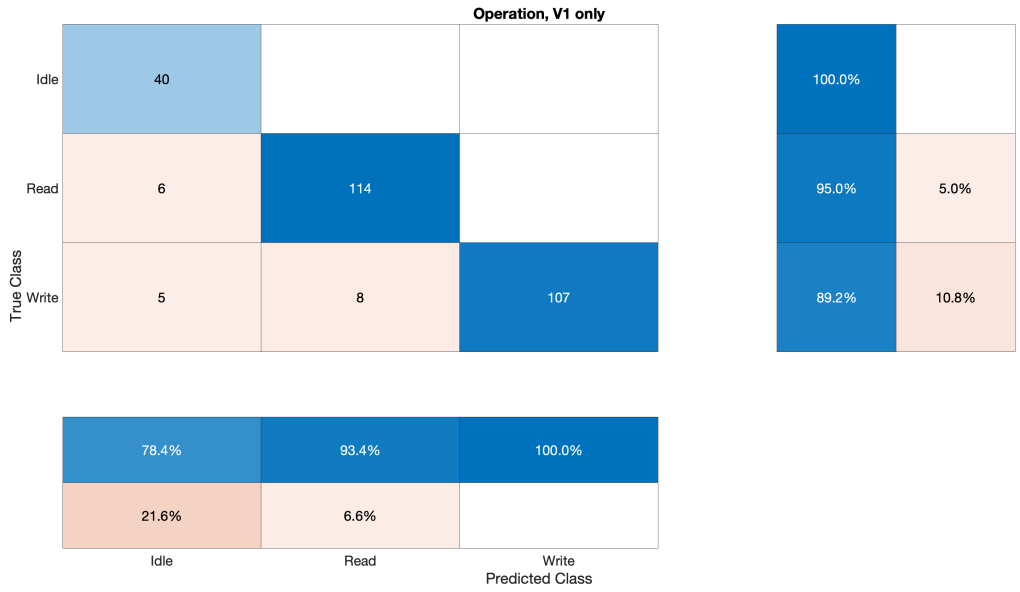
shown for the individual 2022 data sets in Figure 9.18. For comparison in all cases, chance in a 12-way classifier such as this is 10.7% and thus, in all cases, the results are again much better than chance.

9.3 Proof of Concept: Accessing the Power Side-channel via Current Draw for Ransomware Identification

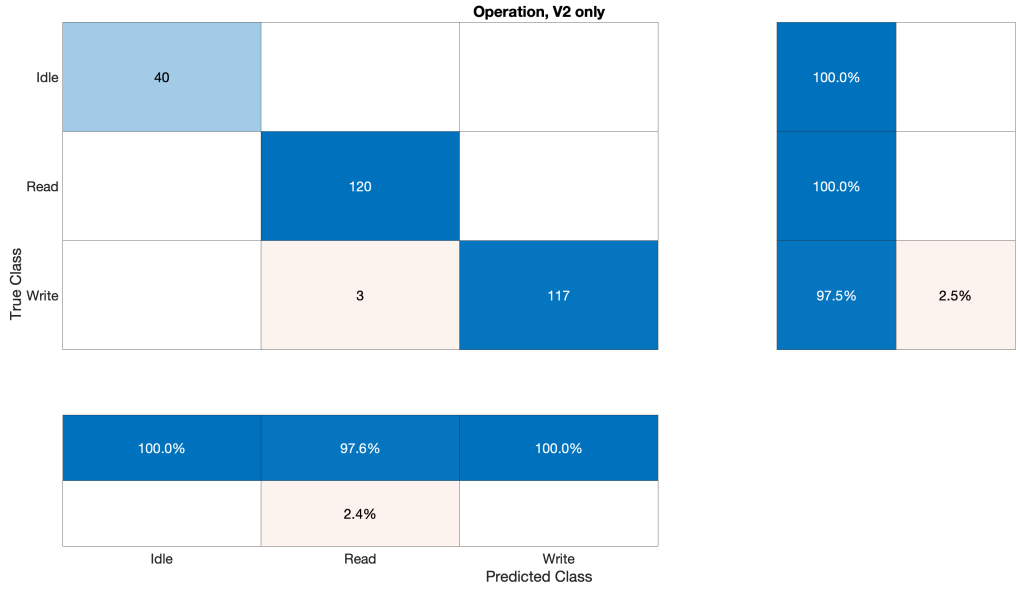
This section describes the hardware set-up, initial current and frequency results, and presents trial classification data.

9.3.1 Method

This data collection in this study was performed in conjunction with Section 7.2.



(a) Classification results by operation for *2022 Drive 1* data



(b) Classification results by operation for *2022 Drive 2* data

Figure 9.14: Classification results by operation for the individual test data sets: *2022 Drive 1* data and *2022 Drive 2* data.

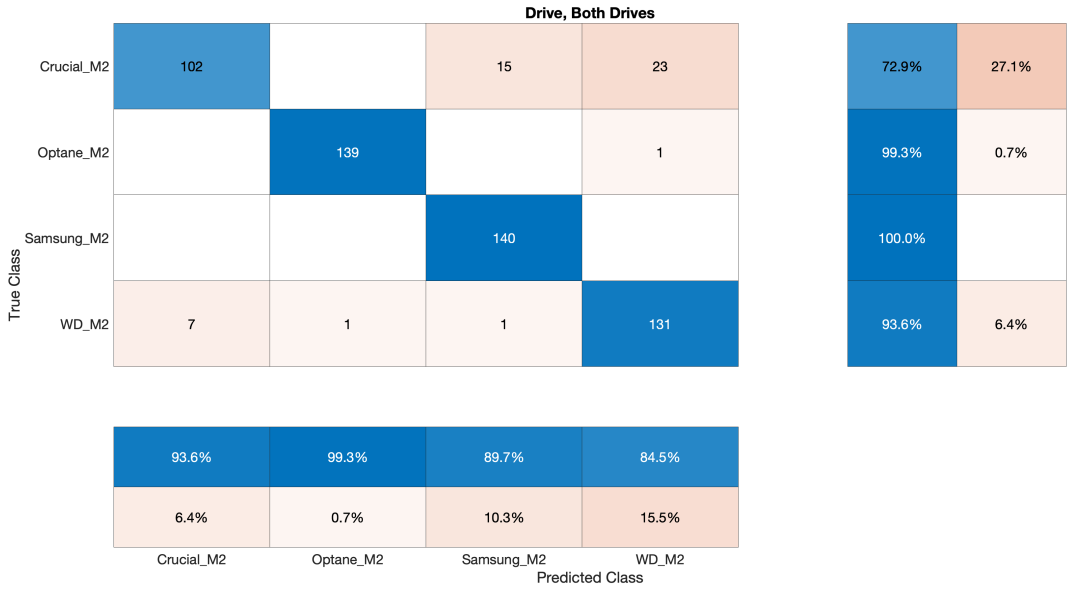


Figure 9.15: Classification by drive.

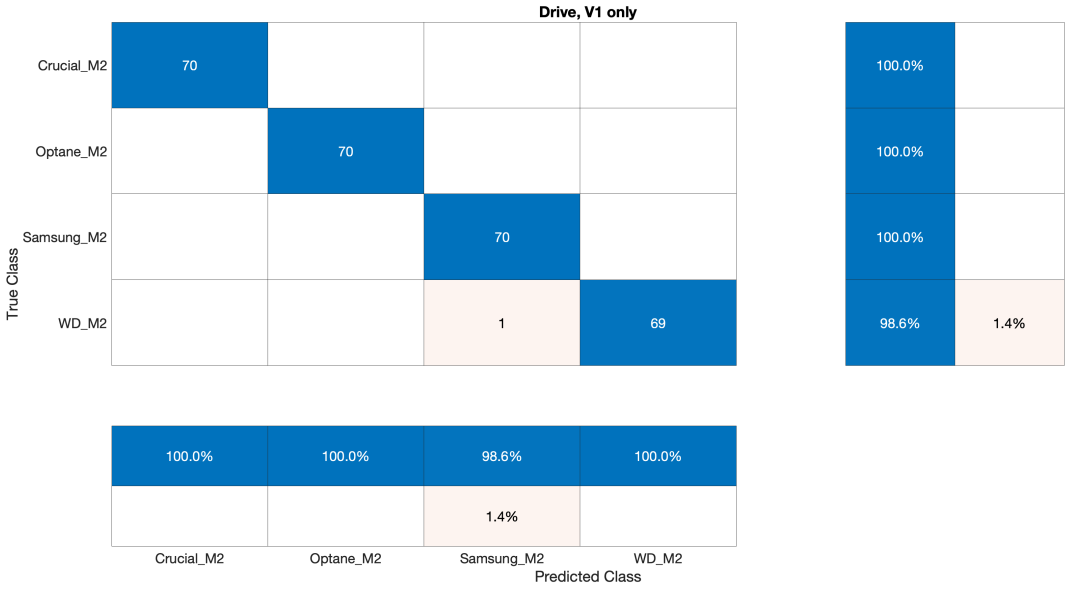
9.3.1.1 Hardware Setup

To collect current measurements during idle, encryption, and ransomware execution, we use a similar hardware set up to Section 9.1.1.1, with the same current probe and amplifier, but with current probe looped around the SATA power line to the SSD containing the Windows 10 Pro operating system and 104 GB of data. Collection apparatus is shown in Figure 9.19. Since the execution of encryption causes the system to become compromised, after each trial the SSD is reformatted and an identical image of the OS and data is reloaded using Redo Rescue[1].

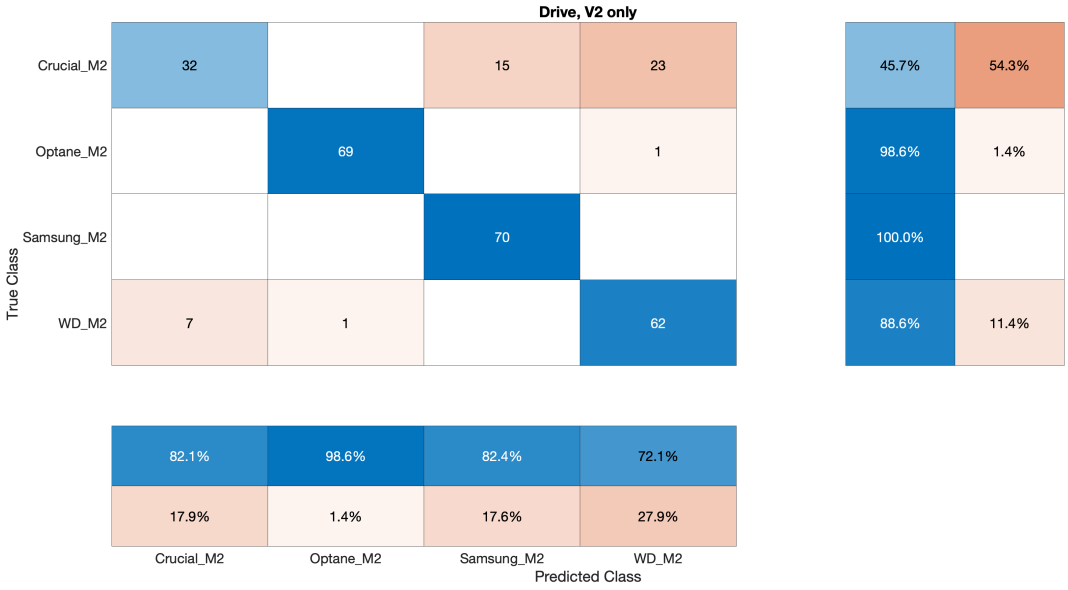
9.3.1.2 Ransomware Current and Frequency Signatures

Example current signatures of idle, 7zip encryption, and live ransomware samples (Darkside, Sodinokibi, WannaCry) are shown in Figure 9.20. Data is recorded for 150 seconds, with

[1]http://redorescue.com



(a) Classification results by by drive model for *2022 Drive 1* data



(b) Classification results by by drive model for *2022 Drive 2* data

Figure 9.16: Classification results by by drive model for the individual test data sets: *2022 Drive 1* data and *2022 Drive 2* data.

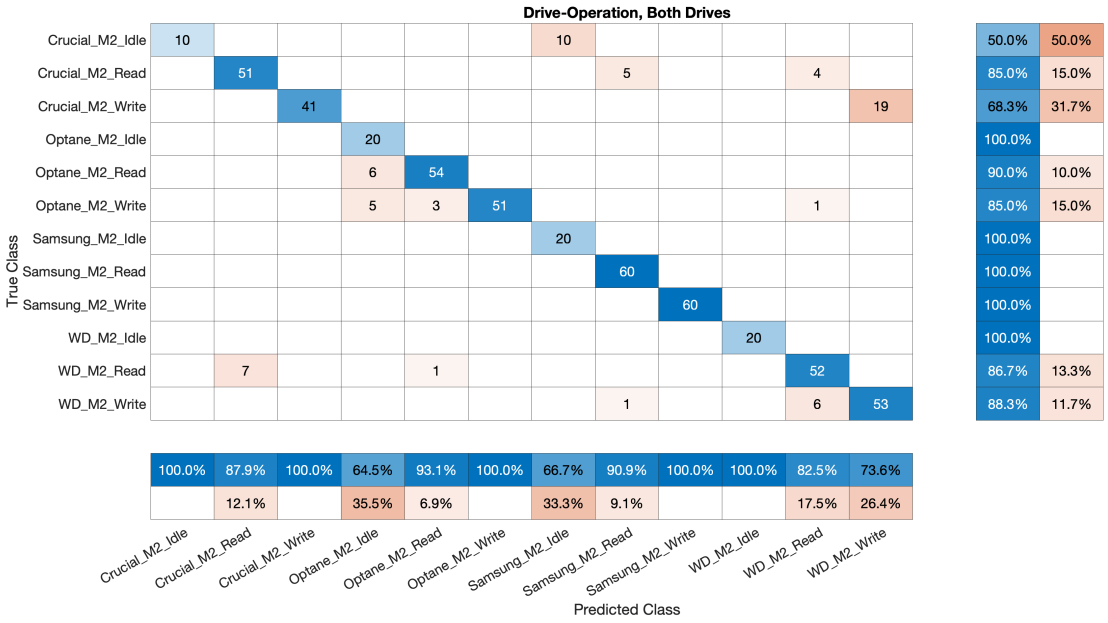
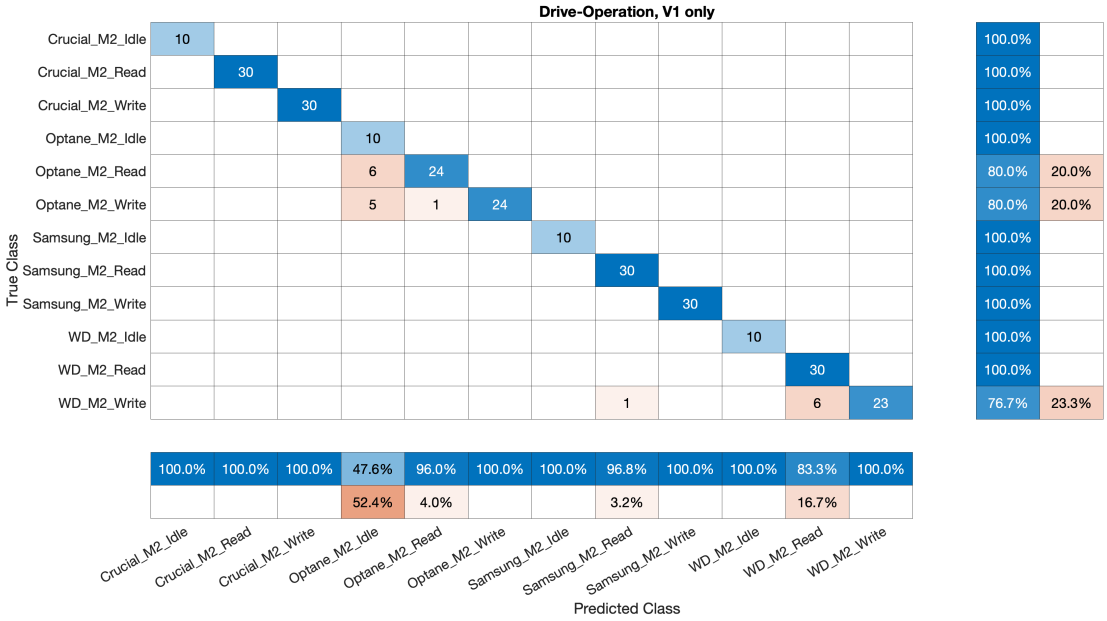


Figure 9.17: Classification results by drive model and operation for both *2022 Drive 1* data and *2022 Drive 2* data sets.

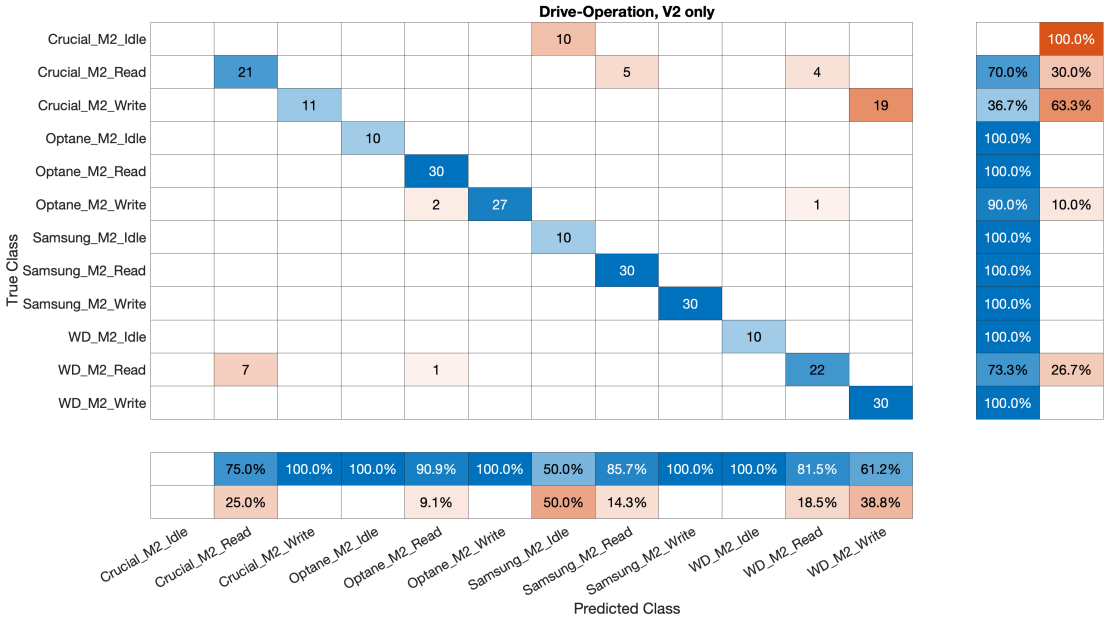
encryption and Ransomware triggered manually at the 30 second mark. Figure 9.21 shows one second of collection at the 60 second mark. Distinct current signatures are evident in both figures.

The frequency spectrum of the example current measurements of idle, 7zip encryption, and live ransomware samples (Darkside, Sodinokibi, WannaCry) are shown in Figure 9.22. Figure 9.23 narrows in on the low band frequency components up to 1 kHz one second of collection at the 60 second mark. With distinct frequency components, classification based on current data alone is promising.

For initial classification to determine whether we can differentiate between encryption running as part of ransomware and encryption running as part of non-threatening application, we used a randomized collection plant of ten 300 second samples of current measurements for each of the 5 scenarios: idle, 7zip encryption, Darkside ransomware, Sodinokibi ransomware, and WannaCry ransomware. We used principal component analysis with frequency features and a



(a) Classification results by drive model and operation for *2022 Drive 1* data



(b) Classification results by drive model and operation for *2022 Drive 2* data

Figure 9.18: Classification results by drive model and operation for the individual test data sets: *2022 Drive 1* data and *2022 Drive 2* data.

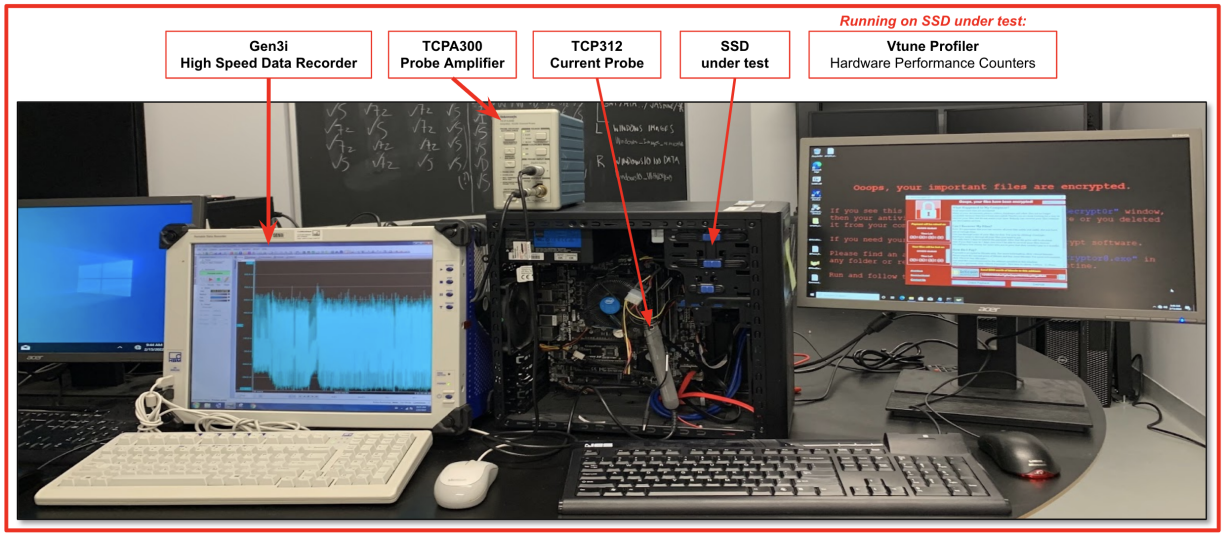


Figure 9.19: Collection apparatus.

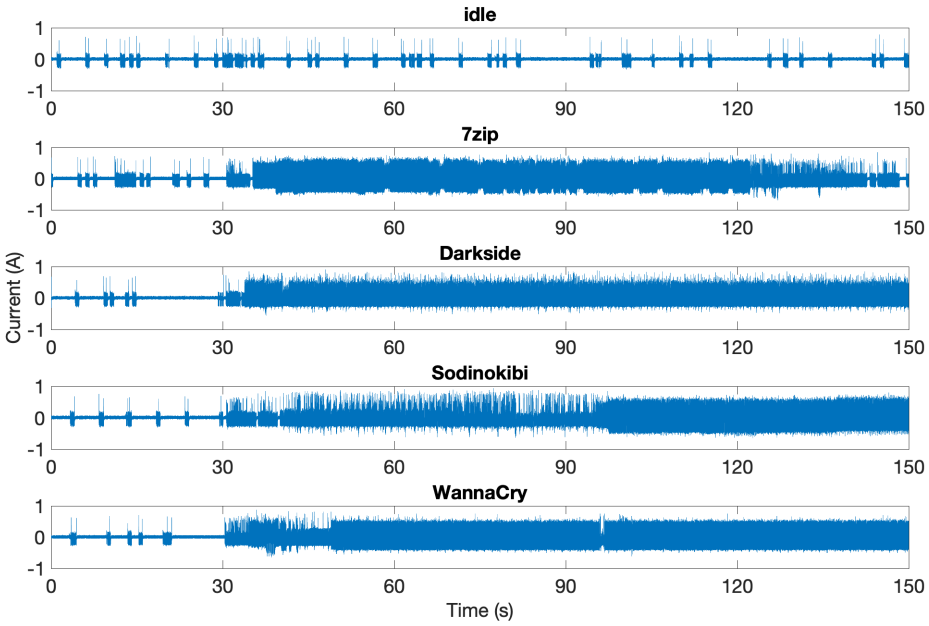


Figure 9.20: Current Signatures of idle, 7zip encryption, and live ransomware samples (Darkside, Sodinokibi, WannaCry). Data is recorded for 150 seconds, with encryption and Ransomware triggered manually at the 30 second mark.

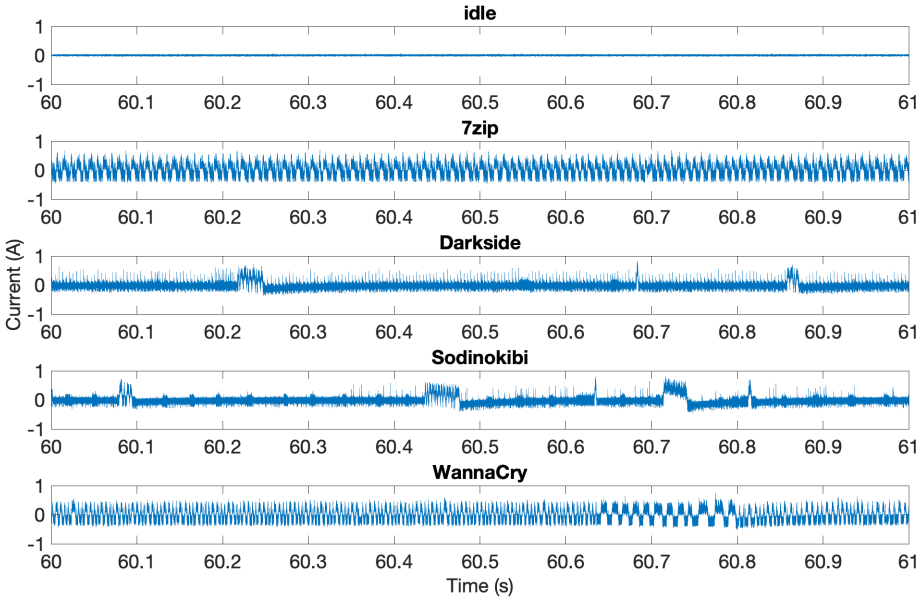


Figure 9.21: 1 second section of Current Signatures of idle, 7zip encryption, and live ransomware samples (Darkside, Sodinokibi, WannaCry).

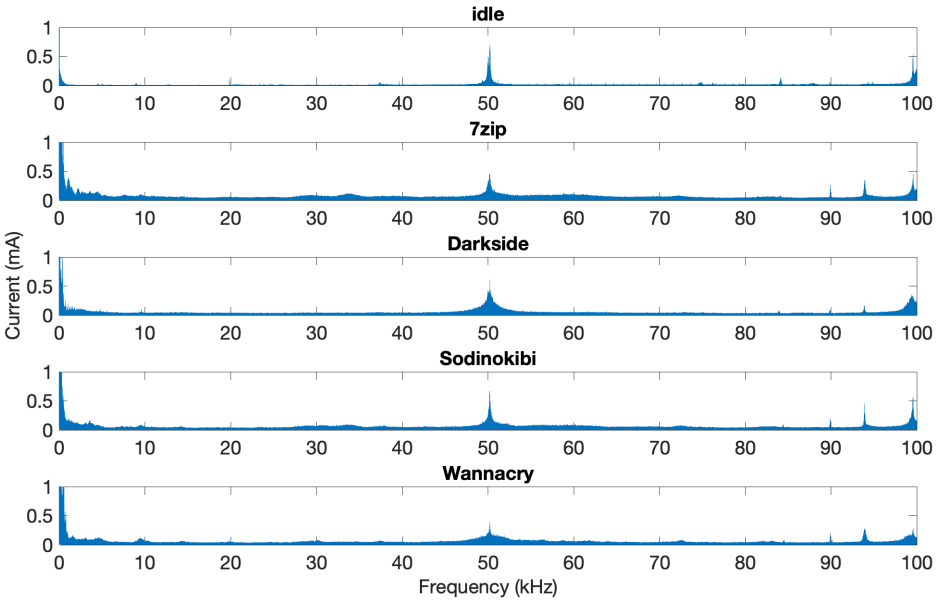


Figure 9.22: Frequency Spectrum of idle, 7zip encryption, and live ransomware (Darkside, Sodinokibi, WannaCry) at sampling rate of 200kHz with amplitude truncated after 1 mA.

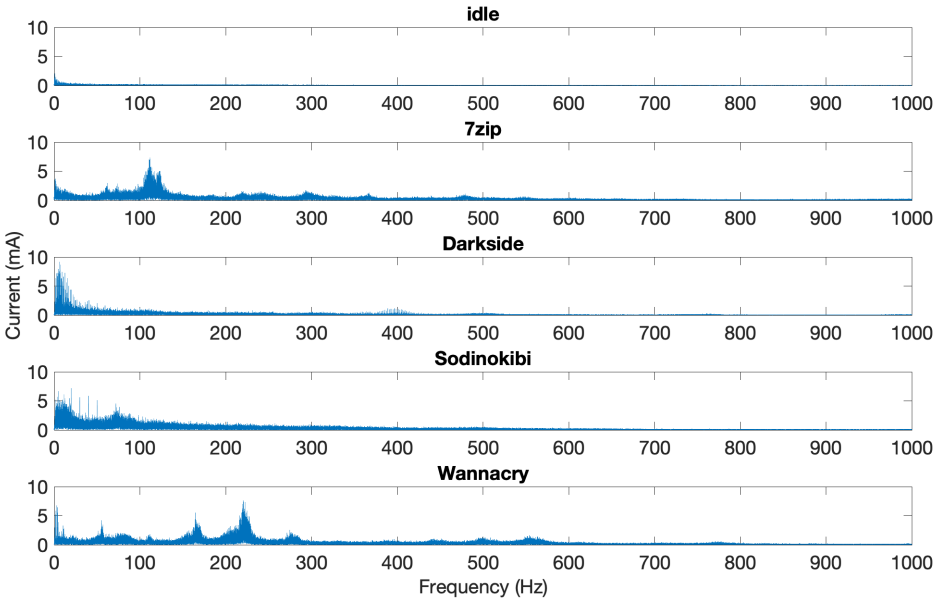


Figure 9.23: 1kHz of Frequency Spectrum of idle, 7zip encryption, and live ransomware (Darkside, Sodinokibi, WannaCry) at sampling rate of 200kHz.

Random Forest classifier. Classification results showed ∼80% accuracy for 5-way classification and up to ∼87% accuracy for binary (ransomware vs. benign) classification, as shown in the confusion matrix in 9.24.

9.3.2 Current Side-Channel for Ransomware Detection

Cursory analysis of current measurements showed distinct current signatures for different operation types, as shown in Figure 9.25, where the top subplot shows 175 seconds of OS background ("idle") activity, the 2nd row shows 7zip encryption, and the bottom 3 rows show DarkSide, Sodinokibi, and WannaCry ransomware, respectively. In each trial, 30 seconds of OS background activity followed by 30 seconds of current associated with running Vtune Profiler is evident, with the designated trial operation beginning around the 60 second mark. Figure 9.26 shows the first 5 seconds of the designated trial operation, beginning at 60 seconds into the trial.

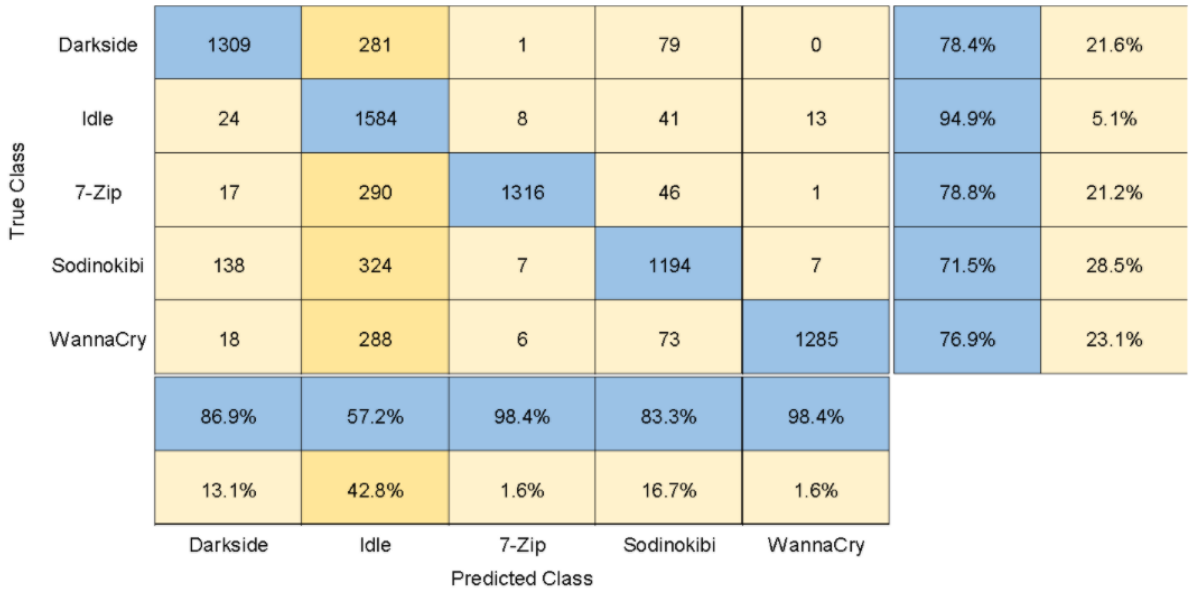


Figure 9.24: Confusion matrix for 5-way classification of idle, 7zip encryption, and live ransomware (Darkside, Sodinokibi, WannaCry).

As Both the Vtune application and encryption/ransomware operations were triggered manually, and the lack of synchronization across trials is clear.

9.4 Study: 35-class Current-draw-based Side-channel Analysis for Ransomware Classification

This section describes the analysis of the power side-channel accessed by monitoring the current supplied to an SSD containing the OS and system storage. The current data analyzed in this section was collected simultaneously with Vtune HPCs in Section 8.1 in order to facilitate direct comparisons between HPC-based and Current-based results.

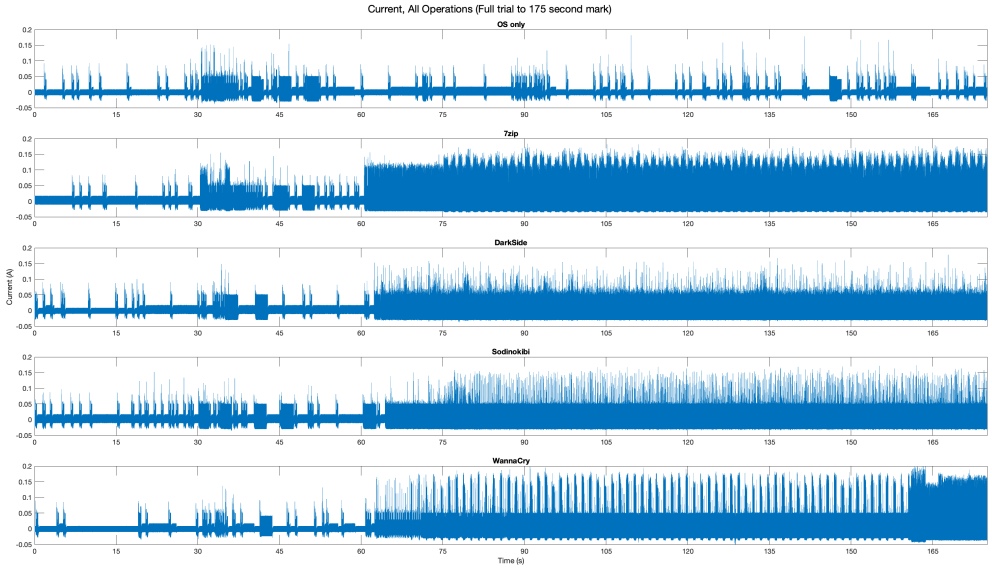


Figure 9.25: Current for a representative trial for each operation type from trial initiation for 175 seconds.

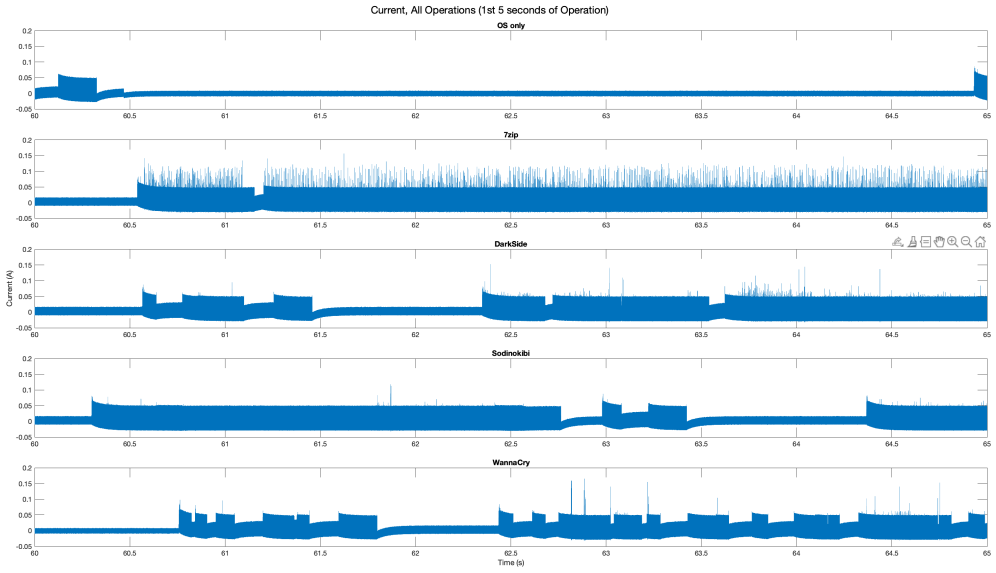


Figure 9.26: Current for a representative trial for each operation type from operation initiation at 60 second mark for 5 seconds.

9.4.1 Part 1: Power Feature Generation and Classifier Identification

To reduce the number of classifiers considered for training and testing, a separate set of training data, was collected using the same simultaneous Vtune and current collection process identified in Figure 8.3, with the parameters shown in Figure 9.27. (Note: The Vtune performance counter data was not used in this work.) This enabled process refinement for feature generation and classifier training and cross validation results, while still keeping eventual training and testing data separate.

# Trials (per operation)	Collection Method	Restore Process
10	SCRIPT	IFL

Trial Parameters		
OS idle (current only)	15	s
Vtune idle (add Vtune)	15	s
Test Operation Duration	15	s

Data Recorder Parameters		
Sampling Rate	200	kS/s
Duration	45	s
Span	4	A
Offset	-1.5	V

Operations Tested (36 total)		
Benign		
os, 7zip, 18 SPEC Workstation 3.1 CPU Benchmarks		
Ransomware	*did not encrypt files	Snatch_0422
Babuk_1022	Phobos_0122	Snatch_0521
DarkSide_Spr21	Phobos_0522	Sodinokibi_0222
DarkSide_0521	Phobos_0922	WannaCry_Spr22
Gibberish_0422	Sodinokibi_Spr21	WannaCry_0622
HiddenTear_0422	Sodinokibi_0722	WannaCry_1022*

Vtune Parameters		
HPCs	80	Custom (F1 rank)
Sampling Interval	0.5	ms
Duration	15	s
Report "bins"	600	
Effective Sampling Rate	0.005	s

Figure 9.27: Data Collection Summary for simultaneous current and Vtune HPC collection for 36-classes, used to fine-tune the feature generation and select well-performing classifiers for further exploration. One ransomware class (WannaCry_1022) was eventually dropped from the final study because it did not perform encryption operations.

9.4.1.1 Generation of Power Spectral Density Features

Periods of current data ranging from 1 to 10 seconds were converted to Power Spectral Density (PSD) in both Watts and dB using Welch's method with various (non-exhaustive) combinations

Table 9.3: Current data from trials were converted to Power Spectral Density using Welch's Method with varying combinations of data duration, window size, and maximum frequency.

Trial Duration (s)	1, 2, 3, 5, 10
Welch Window (ms)	300-1000 ms (in 100 ms increments)
Max Frequency (Hz)	5, 10, 25, 50, 100, 500, 1000

of window size and maximum frequencies shown in Table 9.3. Generated features were provided as input to the MATLAB Classification Learning Application.

9.4.1.2 Classifier Selection with MATLAB Classification Learning App

Resulting power features were loaded into the MATLAB Classification Learning App, where all models were trained with generated power features using 10-fold cross validation. Validation accuracy for each set of features and each of the 32 available non-optimized models was recorded and features were ranked from high to low for the model that generated the highest cross-validation accuracy (Coarse Tree). Results for all recorded trials are provided in Figure 9.28, and the top twenty feature combinations (based on Coarse Tree Cross Validation Accuracy rank) are shown in Figure 9.29. Based on these results, the three best performing classifiers that had been previous explored in Chapter 8 (Bagged Tree Ensemble, Linear SVM, and Fine KNN) were selected for further analysis, along with the two classifiers that generated the best overall Cross Validation Accuracy (Coarse Tree and Medium Gaussian SVM). The resulting classifiers, selected to balance best cross validation performance for power features combined with previous performance using Vtune hardware performance counter features, are listed below:

1. Bagged Tree Ensemble

2. Coarse Tree

3. Linear SVM

4. Medium Gaussian SVM

5. Fine KNN

Additional information on each of the categories of classification models available in the application are included in Section 7.5.2.1.

9.4.2 Part 2: Current-based Power side-channel Analysis for Ransomware vs. Benign Operations

This section describes how the results from Section 9.4.1.2 were applied to current data collected in conjunction with Vtune performance counter data from Chapter 8 to compare the effectiveness of side-channel analysis methods on similar data.

9.4.2.1 Data Collection Apparatus and Procedure

Current trial data for this study was collected alongside Vtune performance counter data, with the parameters shown in Figure 8.2, following the procedure described in Figure 8.3. Twenty trials each of 20 benign operations (OS "idle", 7zip Encryption of a large file, and 18 SPEC Workstation3.1 CPU benchmarks) and 15 ransomware operations (Figure 8.1) were collected in randomized order. As with the previous current studies in this Chapter, the current probe described in Section 9.1.1 was placed around the 5 power lines of a PCIe extension cable to a Samsung 970 EVO PLUS solid state drive, which was running the operating system and hardware described in Section 7.2.1, and data was collected on a Gen3i Data Recorder at 200

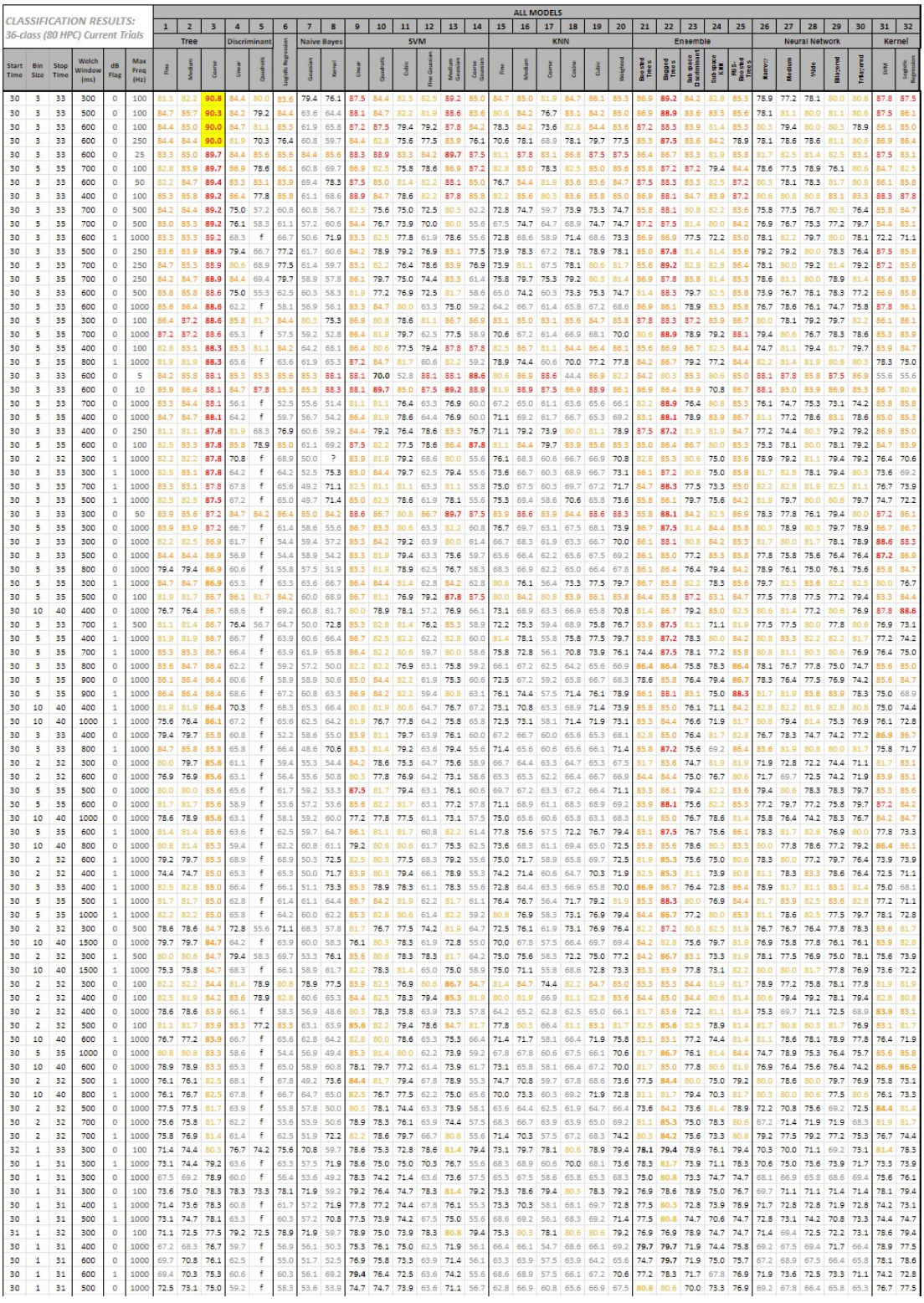


Figure 9.28: Cross Validation Accuracy for non-exhaustive combinations of trail duration, Welch Window size, maximum feature frequency, and dB units, recorded from MATLAB Classification Learning Application.

CLASSIFICATION RESULTS: 36-class (80 HPC) Current Trials							ALL MODELS																										
							1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
							Tree			Discriminant		Logistic Regression	Naive Bayes		SVM					KNN					Ensemble					Neural Network					Kernel	
Start Time	Bin Size	Stop Time	Welch Window (ms)	dB Flag	Max Freq (Hz)	fine	medium	coarse	linear	quadratic		Gaussian	kernel	linear	quadratic	cubic	fine Gaussian	medium Gaussian	coarse Gaussian	fine	medium	coarse	cosine	cubic	weighted	Boosted Trees	Bagged Trees	Subspace Discriminant	Subspace KNN	RUS-Boosted Trees	Narrow	Medium	Wide	Bilayered	Trilayered	SVM	Logistic Regression
30	3	33	300	0	100	81.1	82.2	**90.8**	84.4	80.0	83.6	79.4	76.1	87.5	84.4	82.5	82.5	89.2	85.0	84.7	85.0	81.9	84.7	86.1	85.3	86.9	**89.2**	84.2	82.8	85.3	78.9	77.2	78.1	80.0	80.8	87.8	87.5
30	3	33	500	0	100	84.7	85.7	**90.3**	84.2	79.2	84.4	63.6	64.4	88.1	84.7	82.2	81.9	88.6	83.6	80.6	84.2	76.7	83.1	84.2	85.0	86.9	**88.9**	83.6	83.3	85.6	78.1	81.1	80.0	81.1	80.6	87.5	86.1
30	3	33	600	0	100	84.4	85.0	**90.0**	84.7	81.1	85.3	61.9	65.8	87.2	87.5	79.4	79.2	87.8	84.2	78.3	84.2	73.6	82.8	84.4	83.6	87.2	88.3	83.9	81.4	85.3	80.3	79.4	80.0	80.3	78.9	86.1	85.0
30	3	33	600	0	250	84.4	84.4	**90.0**	81.9	70.3	76.4	60.8	59.7	84.4	82.8	75.6	77.5	83.9	76.1	70.6	78.1	68.9	78.1	79.7	77.5	85.3	**87.5**	83.6	84.2	78.9	78.1	78.6	78.6	81.1	80.6	86.9	86.4
30	3	33	600	0	25	83.3	85.0	**89.7**	84.4	85.6	85.6	84.4	85.6	88.3	88.9	83.3	84.2	**89.7**	87.5	81.1	87.8	83.1	86.8	87.5	87.5	86.4	86.7	83.3	81.9	85.8	81.7	82.5	81.4	82.5	83.1	87.5	83.1
30	5	35	700	0	100	82.8	83.9	**89.7**	86.9	78.6	86.1	60.8	69.7	86.9	82.5	75.8	78.6	86.9	87.2	82.8	85.0	78.3	82.5	85.0	85.6	85.8	87.2	87.2	79.4	84.4	78.6	77.5	78.9	76.1	80.6	84.7	82.5
30	3	33	600	0	50	82.2	84.7	**89.4**	83.3	83.1	83.9	69.4	78.3	87.5	85.0	81.4	82.2	88.1	85.0	76.7	84.4	81.9	83.6	83.6	84.7	87.5	88.3	83.3	82.5	87.2	80.3	78.1	78.3	81.7	80.8	86.1	85.8
30	3	33	400	0	100	85.3	85.8	**89.2**	86.4	77.8	85.8	61.1	68.6	88.9	84.7	78.6	82.2	87.8	85.8	82.2	85.6	80.3	83.6	85.8	85.0	86.9	88.1	84.7	83.9	87.2	80.6	80.8	80.8	83.1	83.3	88.3	87.8
30	3	33	700	0	500	84.2	84.4	**89.2**	75.0	57.2	60.8	60.8	56.7	82.5	75.6	75.0	72.5	80.3	62.2	72.8	74.7	59.7	73.9	73.3	74.7	85.8	88.1	80.8	82.2	83.6	75.8	77.5	76.7	80.3	76.4	85.8	84.7
30	5	35	700	0	500	85.0	85.3	**89.2**	76.1	58.3	61.1	57.2	60.6	84.4	76.7	73.9	70.0	80.0	55.6	67.5	74.7	64.7	68.9	74.7	74.7	87.2	87.5	81.4	80.0	84.2	76.9	76.7	75.3	77.2	79.7	84.4	83.1
30	3	33	600	1	1000	83.3	83.3	89.2	68.3	f	66.7	50.6	71.9	83.3	82.5	77.8	61.9	78.6	55.6	72.8	68.6	58.9	71.4	68.6	73.3	86.9	86.9	77.5	72.2	85.0	78.1	82.2	79.7	80.0	78.1	72.2	71.1
30	3	33	500	0	250	83.6	83.9	**88.9**	79.4	66.7	77.2	61.7	60.6	84.2	78.9	79.2	76.9	83.1	77.5	73.9	78.3	67.2	78.1	78.9	78.1	85.0	**87.8**	81.4	81.4	85.6	79.2	79.2	80.0	78.3	76.4	87.5	85.8
30	3	33	700	0	250	84.7	85.3	88.9	80.6	68.9	77.5	61.4	59.7	83.1	82.2	76.4	78.6	83.9	76.9	73.9	81.1	67.5	78.1	80.6	81.7	85.6	**89.2**	82.8	82.8	86.4	78.1	80.0	79.2	81.4	79.2	87.2	85.6
30	5	35	700	0	250	84.2	84.7	**88.9**	84.4	69.4	79.7	58.9	57.8	86.1	79.7	75.0	74.4	83.3	61.4	75.8	79.7	75.3	79.2	80.3	81.4	86.9	87.8	85.8	81.4	85.3	78.6	81.1	80.0	78.9	81.4	85.6	83.9
30	3	33	600	0	500	85.8	85.8	88.6	75.0	55.3	62.5	60.3	58.3	81.9	77.2	76.9	72.5	81.7	58.6	65.0	74.2	60.3	73.3	75.3	74.7	81.4	88.3	79.7	82.5	85.8	73.9	76.7	78.1	78.3	77.2	86.9	85.8
30	3	33	600	0	1000	85.6	86.4	**88.6**	62.2	f	58.1	56.9	56.1	83.3	84.7	80.0	63.3	75.0	59.2	64.2	66.7	61.4	65.8	67.2	68.6	86.9	86.1	78.9	83.3	85.8	76.7	78.6	76.1	74.7	75.8	87.8	86.1
30	5	35	300	0	100	86.4	87.2	**88.6**	85.8	81.7	84.4	80.3	75.3	86.9	80.8	78.6	81.1	86.7	86.9	83.1	85.0	83.1	85.6	84.7	85.8	87.8	88.3	87.2	83.9	86.7	80.0	78.1	79.2	79.7	82.2	86.1	86.1
30	5	35	700	0	1000	87.2	87.2	88.6	65.3	f	57.5	59.2	52.8	86.4	81.9	79.7	62.5	77.5	58.9	70.6	67.2	61.4	66.9	68.1	70.0	80.6	**88.9**	78.9	79.2	**88.1**	79.4	80.6	76.7	78.3	78.6	85.3	83.9
30	5	35	400	0	100	82.8	83.1	**88.3**	85.3	81.1	84.2	64.2	68.1	86.4	80.6	77.5	79.4	87.8	87.8	82.5	86.7	81.1	84.4	86.4	86.1	85.6	86.9	86.7	82.5	84.4	74.7	81.1	79.4	81.7	79.7	83.9	84.7
30	5	35	800	1	1000	81.9	81.9	**88.3**	65.6	f	63.6	61.9	65.3	87.2	84.7	81.7	60.6	82.2	59.2	78.9	74.4	60.6	70.0	77.2	77.8	84.2	86.7	79.2	77.2	84.4	82.2	81.4	81.9	80.8	80.3	78.3	75.0

Figure 9.29: Top 20 Cross Validation Accuracy results (based on Coarse Tree validation ranking) for non-exhaustive combinations of trail duration, Welch Window size, maximum feature frequency, and dB units, recorded from MATLAB Classification Learning Application.

Table 9.4: Current data from all training and testing trials were converted to Power Spectral Density using Welch's Method with varying combinations of data duration, window size, and maximum frequency. All power features were generated in both Watts and dBW.

Trial Duration (s)	1, 2, 3, 5, 7, 8, 9, 10, 12, 15
Welch Window (ms)	300-800 ms (in 100 ms increments)
Max Frequency (Hz)	10, 50, 100, 250

kSamples/second. The system was powered down and restored to a clean image after each ransomware trial using Image for Linux.

9.4.2.2 Training data validation

Collected data was sorted and plotted in the time domain to verify trial integrity. Full current traces, full available frequency (100 kHz) spectrogram, and low bandwidth frequency spectrogram for a selected, representative trial of each of the 20 classes of benign operations are provided in Figures 9.30-9.33, while Figures 9.34-9.36 show the respective current and frequency spectrograms for a selected representative trial for the 15 ransomware classes.

Initial observations indicate some variation in trial start time, particularly with the SPEC benchmarks

9.4.2.3 Power Feature Generation

Current trials were again converted to Power Spectral Density in both Watts/Hz and dBW/Hz using Welch's method, with the (exhaustive) combination of parameters in Table 9.4, selected based on their comparatively more promising results in the training all classification algorithms in the Classification Learning App (Section 9.4.1).

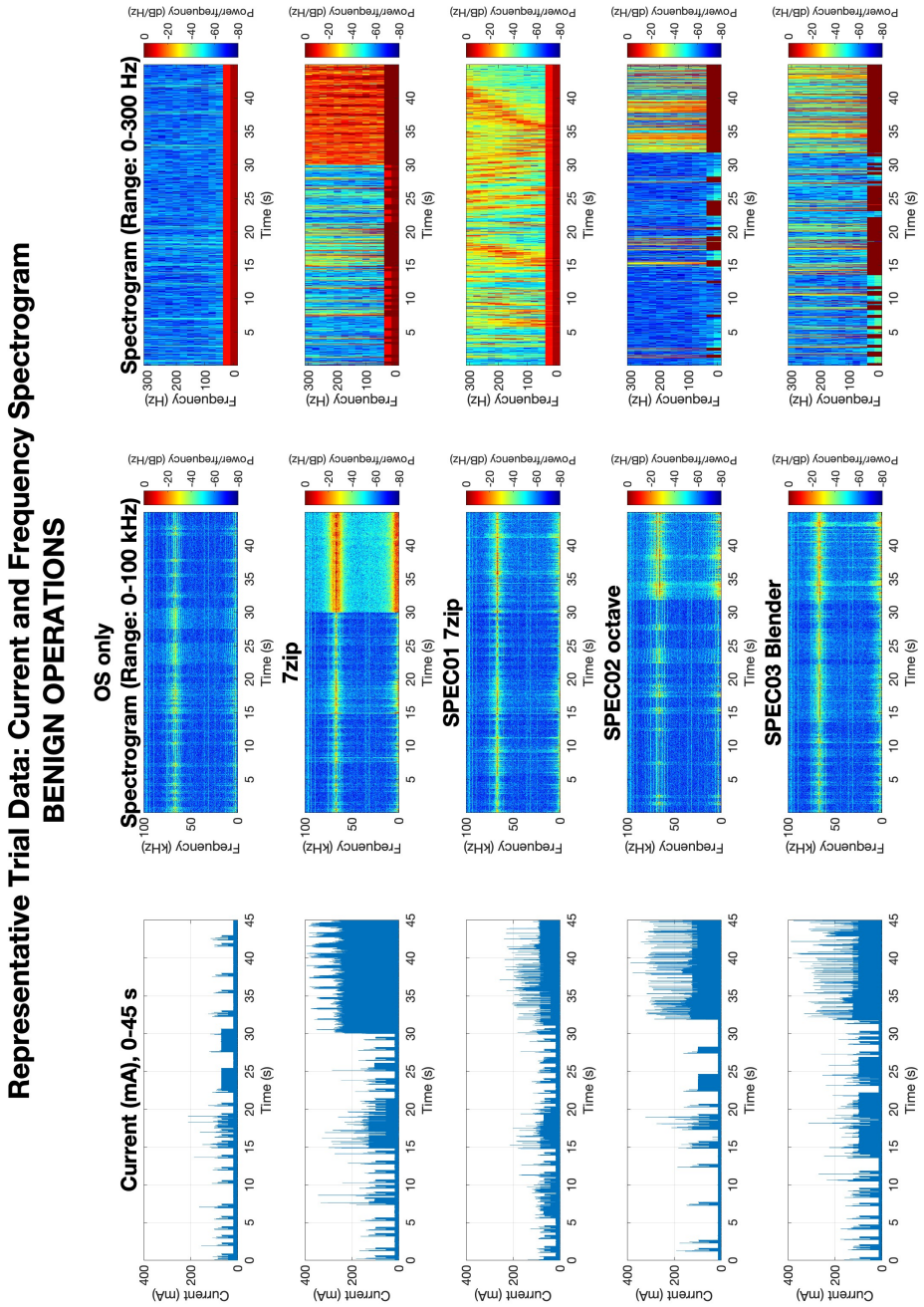


Figure 9.30: Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for benign operations: OS only, 7zip, SPEC01 7zip, SPEC02 octave, SPEC03 Blender.

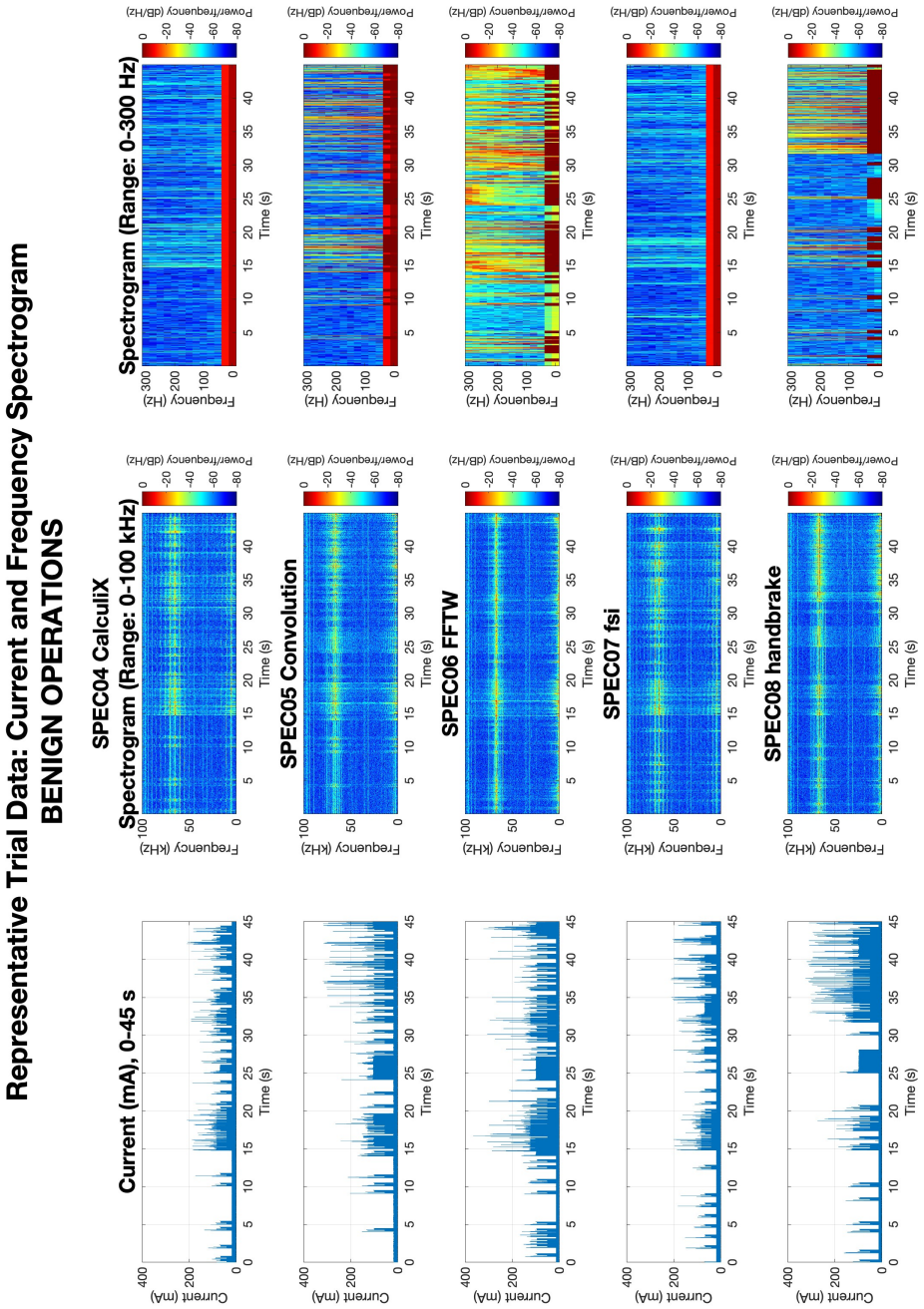


Figure 9.31: Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for benign operations: SPEC04 CalculiX, SPEC05 Convolution, SPEC06 FFTW, SPEC07 fsi, SPEC08 handbrake.

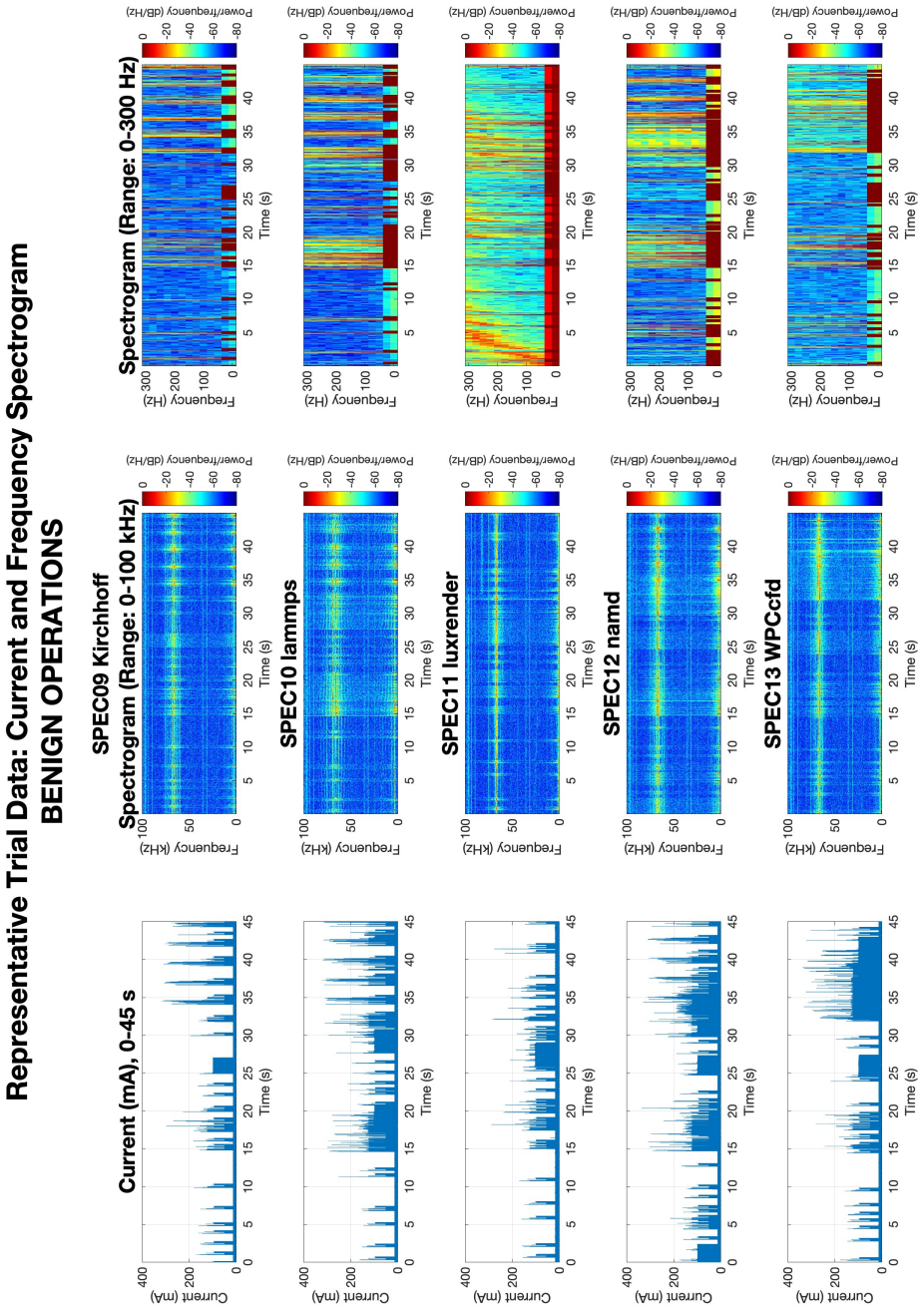


Figure 9.32: Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for benign operations: SPEC09 Kirchhoff, SPEC10 lammps, SPEC11 luxrender, SPEC12 namd, SPEC13 WPCcfd.

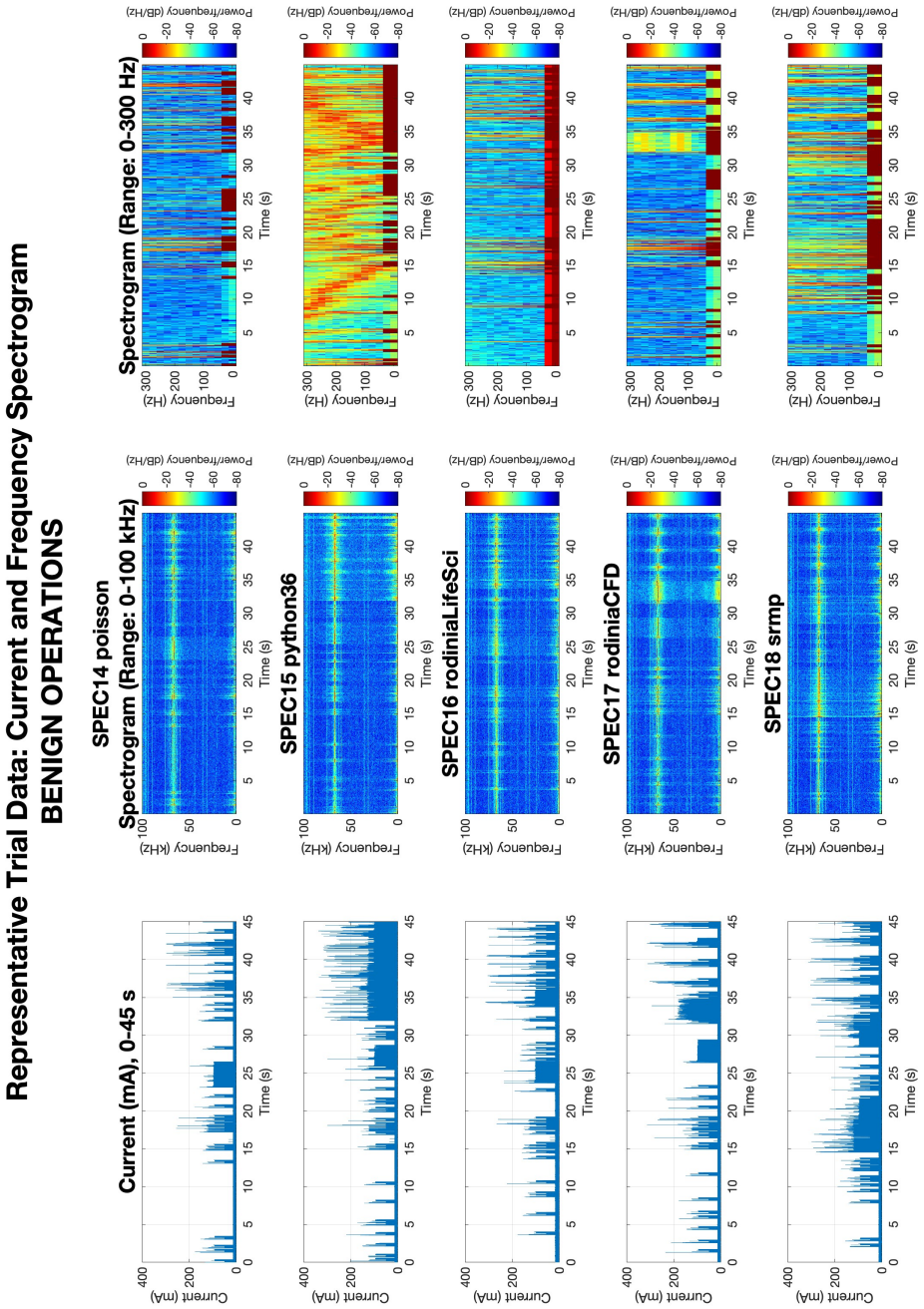


Figure 9.33: Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for benign operations: SPEC14 poisson, SPEC15 python36, SPEC16 rodinaLifeSci, SPEC17 rodiniaCFD, SPEC18 srmp.

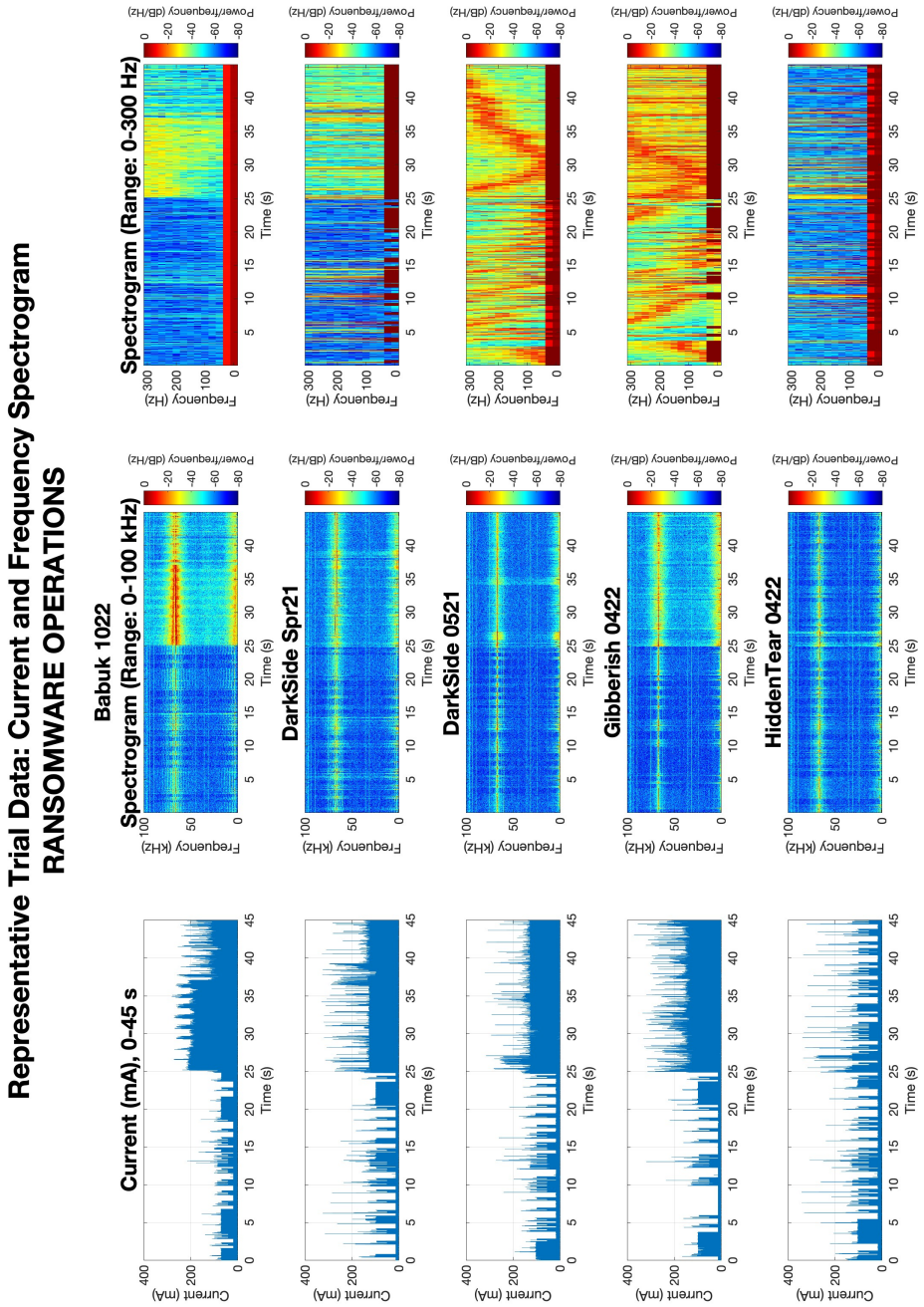


Figure 9.34: Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for ransomware operations: Babuk 1022, DarkSide Spr21, DarkSide 0521, Gibberish 0422, HiddenTear 0422.

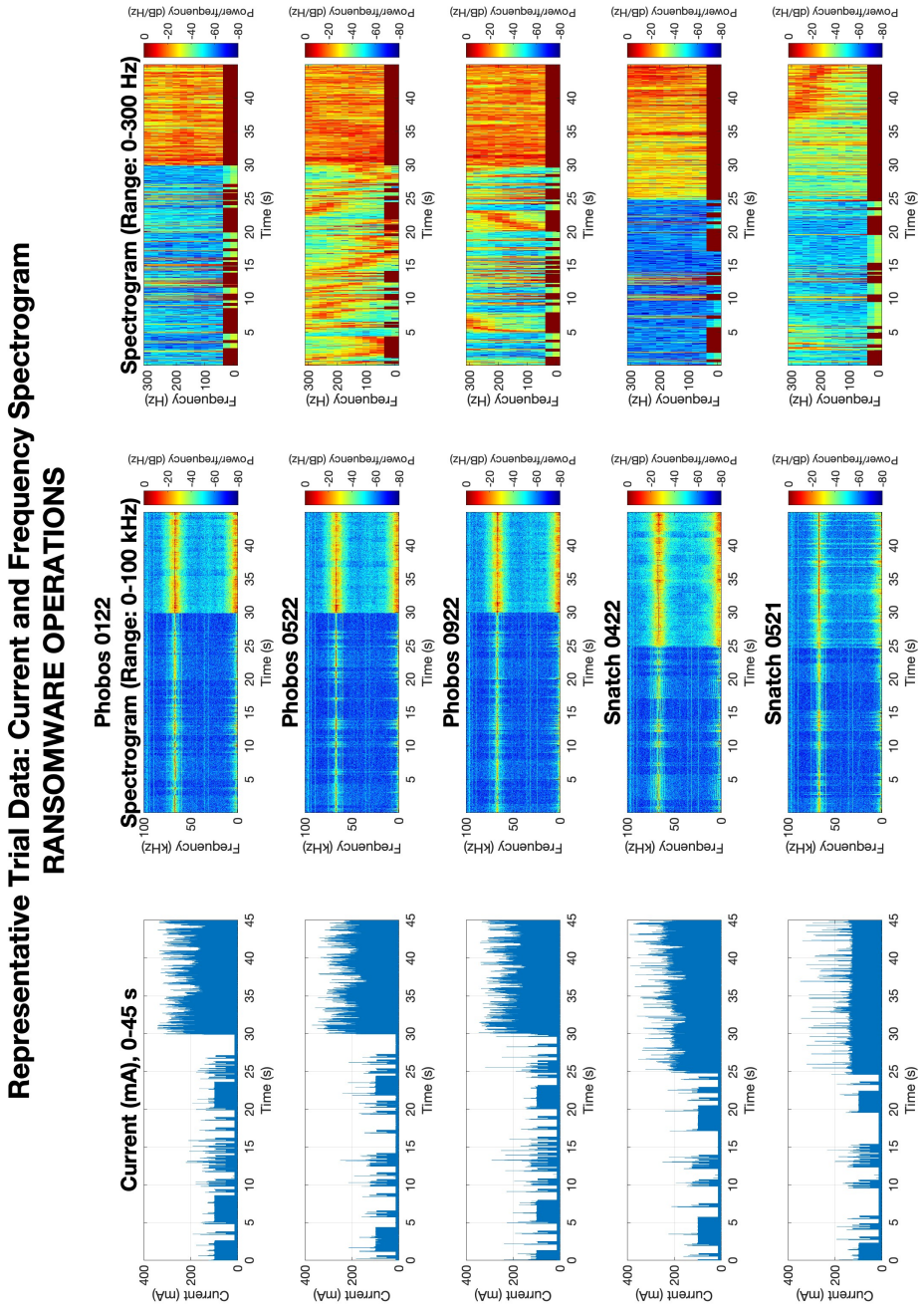


Figure 9.35: Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for ransomware operations: Phobos 0122, Phobos 0522, Phobos 0922, Snatch 0422, Snatch 0521.

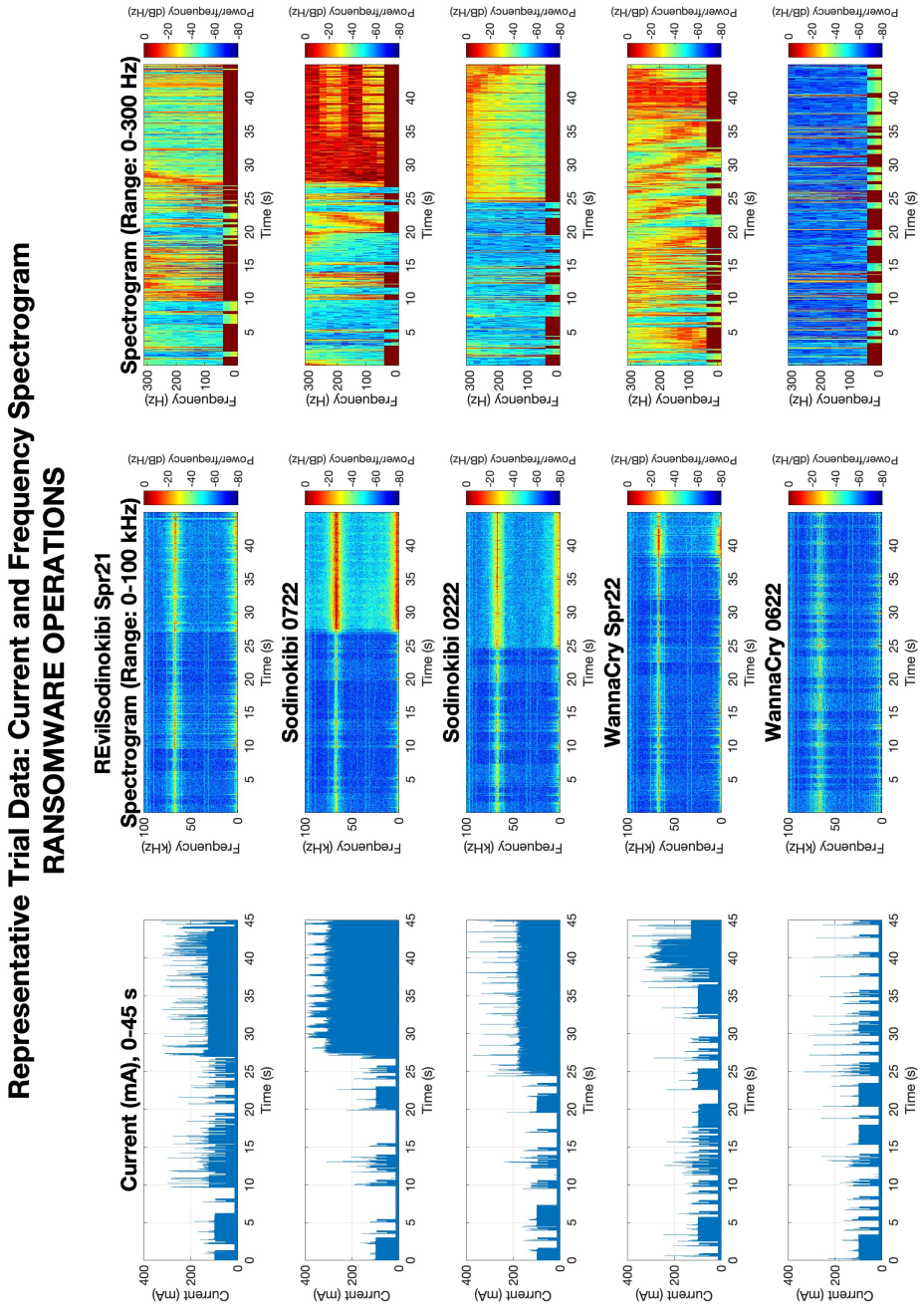


Figure 9.36: Current, Full 100 kHz Frequency Spectrogram, and Cutoff 300 Hz Frequency Spectrogram for ransomware operations: REvilSodinokibi Spr21, Sodinokibi 0722, Sodinokibi 0222, WannaCry Spr22, WannaCry 0622.

9.4.2.4 Classifier Training and Cross Validation

The five classifiers identified in Section 9.4.1.2 were trained with power spectral density features in both W/Hz and dB generated from all combinations of trial duration, Welch Window Size, and Cutoff Frequency indicated in Table 9.4. Resulting Cross Validation Accuracy for W/Hz and dB are shown in Figures 9.37 and 9.37, respectively. Figures 9.39 and 9.40 show MCC results for the corresponding sets of PSD features. In each Figure, the five classifiers trained are listed across the top of each column. Each row represents the performance at a different cutoff frequency, with 10 Hz in the top row, 50 Hz in the 2nd row, 100 Hz in the 3rd row, and 250 Hz in the 4th row. Each of the five colored lines indicates the size of the Welch Window for which PSDs were computed and then averaged together across the selected trial duration segment. Cross validation results show similar general trends for accuracy and MCC value for each (column/row) combination of classifier and frequency cutoff, with mild variation introduced with different window sizes. In all cases, accuracy and MCC increase for the first $\sim$5 seconds to approach 80% and 0.6, respectively, and then level off until the 7-8 second mark where values drop again. In most cases, values increase again at the 15 second mark, but only slightly past 80% accuracy and 0.6 MCC.

9.4.3 Study 2 Results and Initial Conclusions

This section presents classification results for test data predictions made by the trained classifiers generated in the previous section and presents conclusions regarding the viability of using current-based Power Side-channel Analysis to detect ransomware operations.

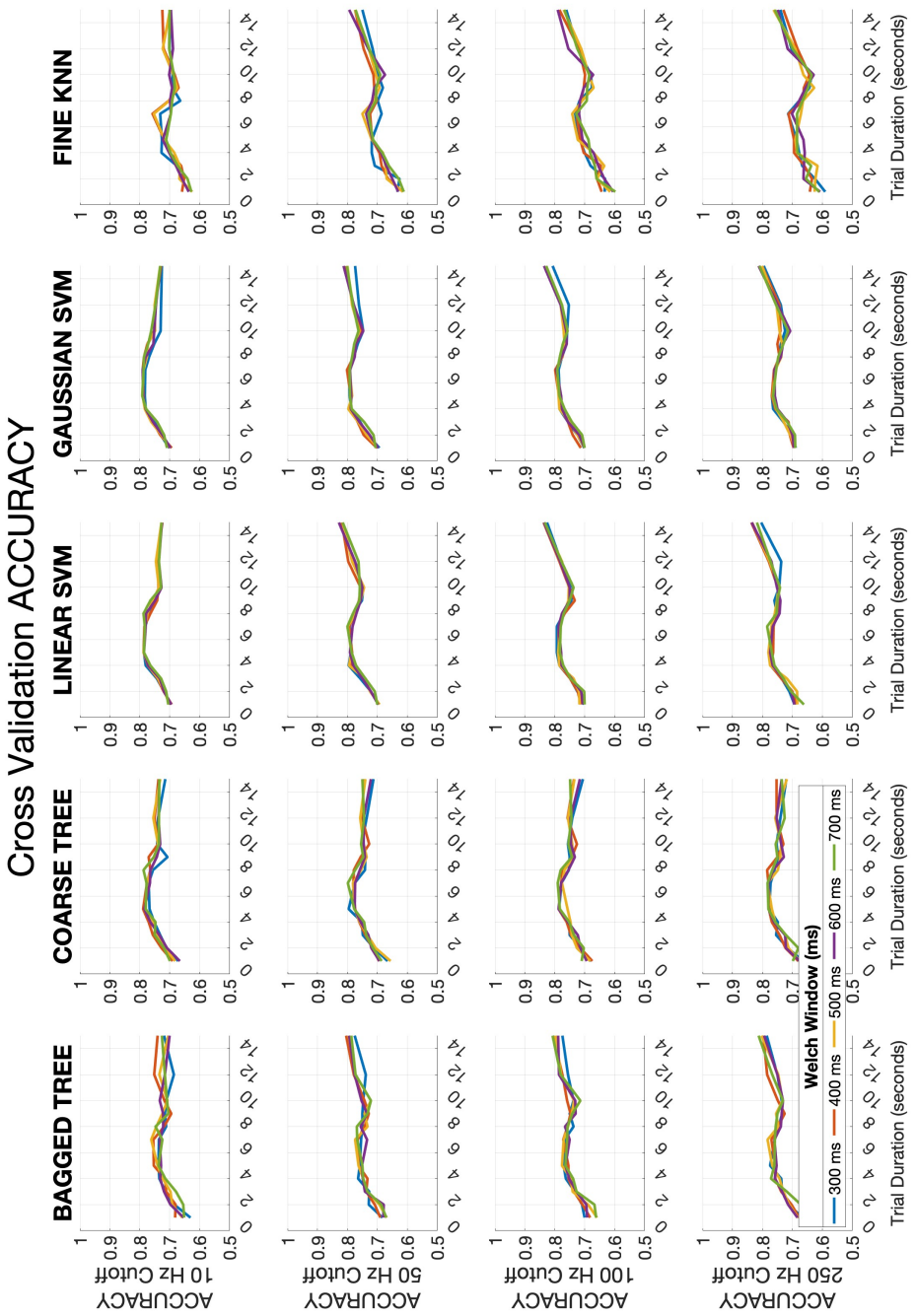


Figure 9.37: Cross Validation Accuracy for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features.

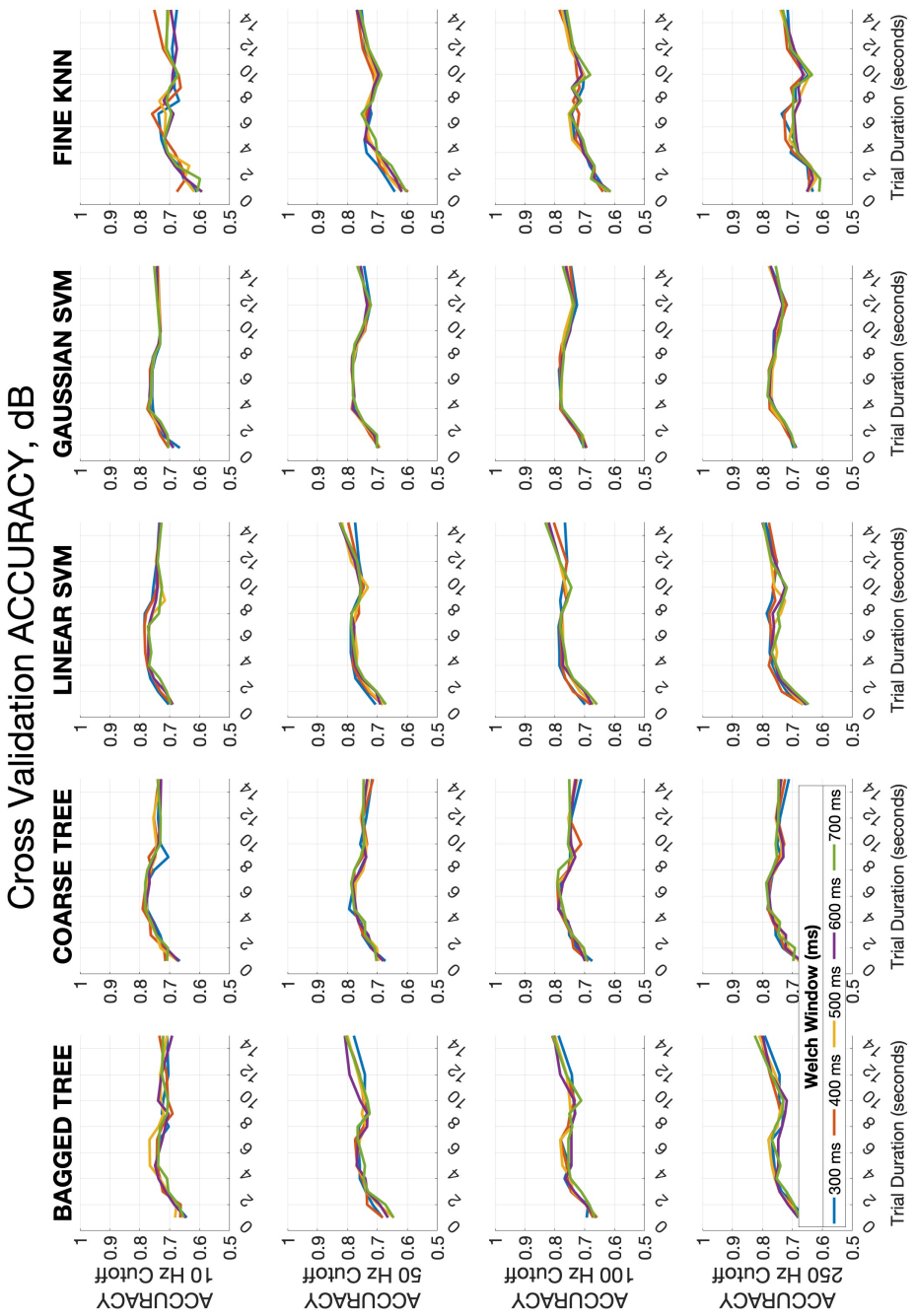


Figure 9.38: Cross Validation Accuracy for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features in dB.

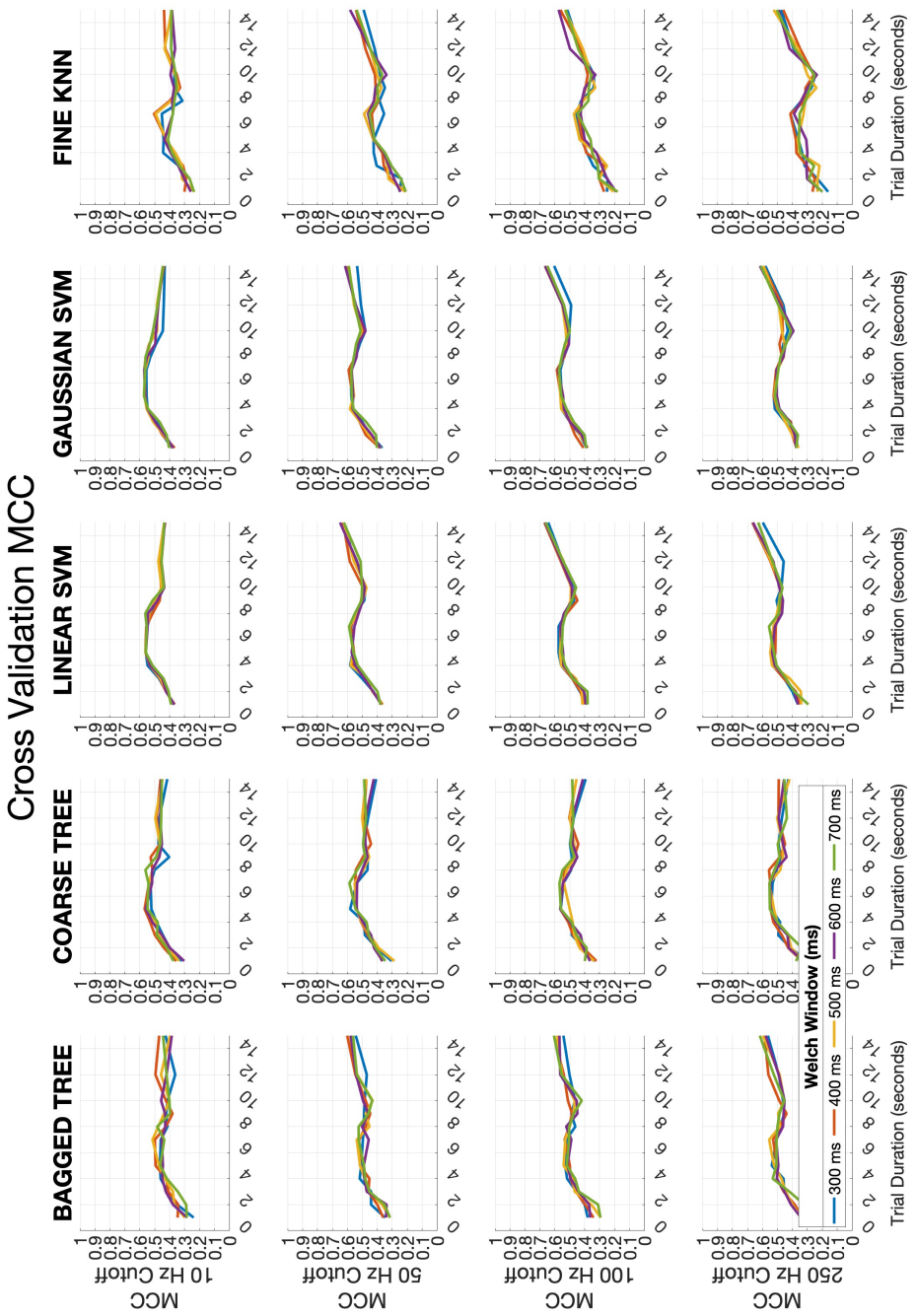


Figure 9.39: Cross Validation MCC for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features.

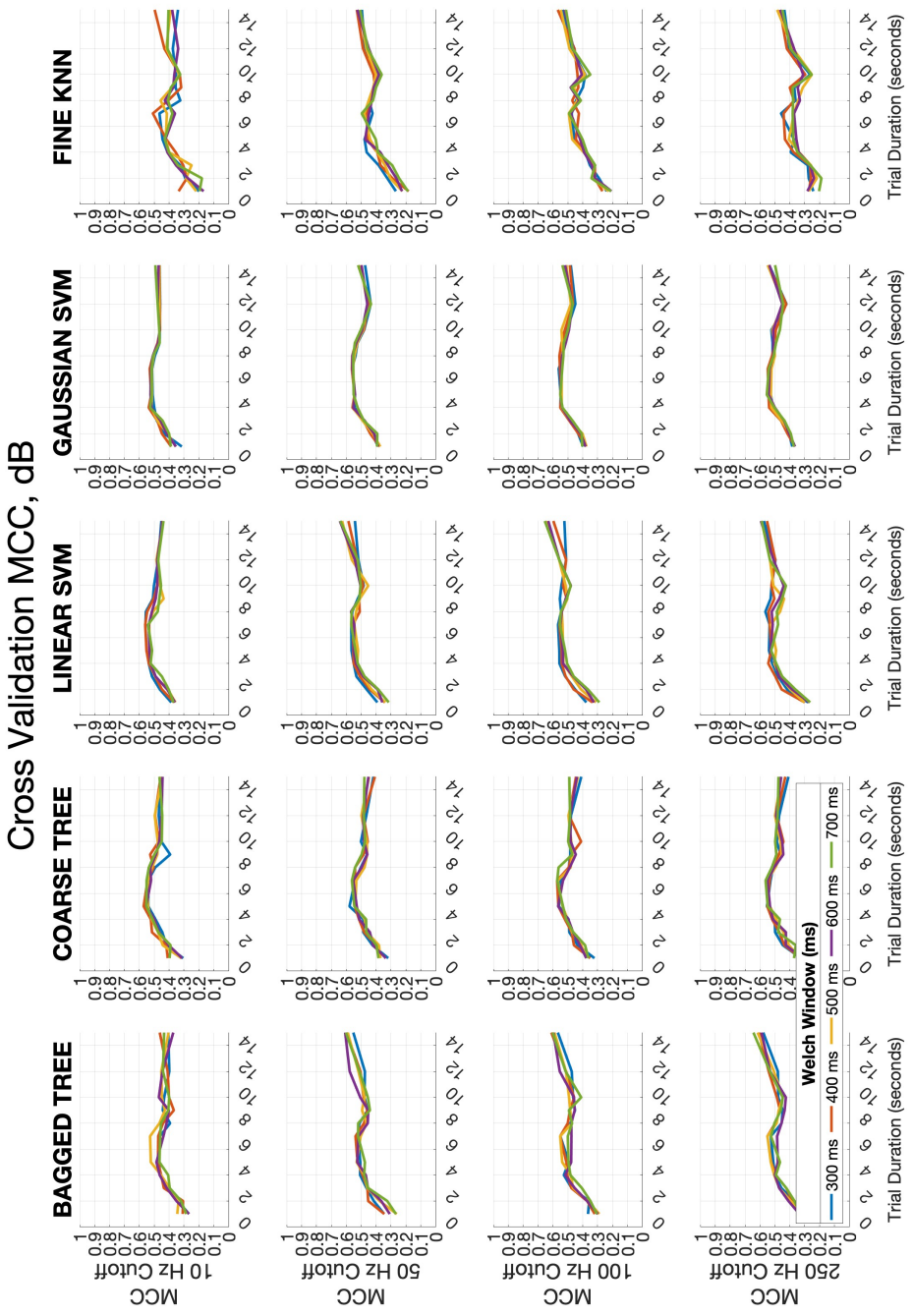


Figure 9.40: Cross Validation MCC for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features in dB.

	Bagged Tree						Coarse Tree						Fine KNN						Linear SVM						Medium Gaussian SVM					
	1	2	3	5	7	10	15	1	2	3	5	7	10	15	1	2	3	5	7	10	15	1	2	3	5	7	10	15	1	2	3	5	7	10	15
b_0S_only	0.68	0.73	0.83	0.80	0.86	0.82	0.93	0.75	0.71	0.80	0.80	0.80	0.80	0.92	0.65	0.62	0.82	0.78	0.77	0.83	0.90	0.66	0.82	0.82	0.80	0.80	0.80	0.86	0.74	0.83	0.80	0.80	0.80	0.80	0.87
b_7zip	0.81	0.85	0.80	0.83	0.86	0.69	0.62	0.96	0.96	0.87	0.83	0.86	0.50	0.62	0.79	0.84	0.75	0.92	0.85	0.68	0.62	0.95	**1.00**	0.88	0.83	0.86	0.84	0.63	0.99	**1.00**	0.93	0.80	0.80	0.80	0.60
b_SPEC01_7zip	0.85	0.91	0.89	0.99	**1.00**	0.94	0.81	0.99	0.98	**1.00**	**1.00**	0.99	0.98	0.84	0.66	0.83	0.86	0.98	**1.00**	0.83	0.86	0.97	0.98	**1.00**	**1.00**	**1.00**	0.94	0.80	0.99	**1.00**	**1.00**	**1.00**	0.95	0.90	0.80
b_SPEC02_octave	0.79	0.93	0.97	**1.00**	**1.00**	**1.00**	0.99	**1.00**	0.99	**1.00**	**1.00**	**1.00**	0.92	0.90	0.74	0.89	0.92	0.99	0.99	0.96	0.93	**1.00**	0.95	**1.00**	**1.00**	**1.00**	**1.00**	0.90	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.85
b_SPEC03_Blender	0.78	0.85	0.81	0.80	0.80	0.87	0.72	0.87	0.83	0.80	0.80	0.80	0.80	0.78	0.78	0.77	0.87	0.74	0.81	0.90	0.80	0.96	0.97	0.80	0.79	0.77	0.68	0.78	0.97	0.99	0.81	0.80	0.72	0.76	0.60
b_SPEC04_CalculiX	0.68	0.86	0.83	0.90	0.99	0.91	0.84	0.88	0.99	0.84	**1.00**	**1.00**	0.80	0.83	0.59	0.63	0.83	0.88	**1.00**	0.94	0.86	0.91	0.97	0.97	**1.00**	**1.00**	0.90	0.96	0.99	**1.00**	0.99	0.98	0.97	0.95	0.92
b_SPEC05_Convolution	0.94	0.95	**1.00**	**1.00**	0.99	**1.00**	0.90	0.97	0.97	**1.00**	**1.00**	**1.00**	**1.00**	0.85	0.84	0.91	0.92	**1.00**	**1.00**	0.93	0.80	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	0.99	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.73
b_SPEC06_FFTW	0.94	0.98	0.98	**1.00**	**1.00**	0.89	**1.00**	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.90	0.90	0.85	0.99	0.99	0.86	**1.00**	**1.00**	**1.00**	0.99	**1.00**	**1.00**	0.81	0.94	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.96	**1.00**
b_SPEC07_fsi	0.96	0.99	**1.00**	0.98	0.99	0.98	0.81	0.99	0.96	**1.00**	**1.00**	**1.00**	**1.00**	0.73	0.92	0.87	0.98	0.95	0.85	0.83	0.69	0.98	**1.00**	**1.00**	0.92	0.98	0.94	0.72	0.99	**1.00**	**1.00**	0.81	0.98	0.98	0.76
b_SPEC08_handbrake	0.82	0.96	0.90	0.87	0.81	0.80	0.98	0.97	0.98	0.90	0.81	0.80	0.80	**1.00**	0.78	0.86	0.88	0.87	0.85	0.82	0.93	0.97	0.99	0.97	0.80	0.84	0.85	**1.00**	**1.00**	**1.00**	0.99	0.80	0.80	0.80	**1.00**
b_SPEC09_Kirchhoff	0.80	0.89	0.98	0.99	**1.00**	**1.00**	**1.00**	0.79	0.83	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.83	0.83	0.94	0.92	0.97	0.96	0.91	0.80	0.81	0.93	**1.00**	**1.00**	**1.00**	**1.00**	0.79	0.81	0.95	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC10_lammps	0.68	0.92	0.99	**1.00**	**1.00**	**1.00**	0.99	0.91	0.97	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.68	0.76	0.76	0.97	**1.00**	0.99	**1.00**	0.90	0.95	0.96	**1.00**	**1.00**	**1.00**	**1.00**	0.99	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC11_luxrender	0.97	0.99	**1.00**	**1.00**	**1.00**	**1.00**	0.95	**1.00**	0.97	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.92	0.94	0.97	**1.00**	0.98	0.96	0.88	0.97	0.98	**1.00**	**1.00**	**1.00**	**1.00**	0.93	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC12_namd	0.97	0.99	0.97	**1.00**	**1.00**	**1.00**	**1.00**	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.94	0.83	0.92	0.96	**1.00**	**1.00**	**1.00**	0.93	0.91	0.97	**1.00**	**1.00**	0.98	0.97	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
b_SPEC13_WPCcfd	0.71	0.78	0.92	0.98	0.99	0.99	0.98	0.85	0.80	**1.00**	**1.00**	**1.00**	**1.00**	0.96	0.64	0.70	0.83	0.93	0.97	**1.00**	0.92	0.80	0.82	0.92	0.95	0.99	0.99	0.95	0.89	0.82	0.85	0.82	0.85	**1.00**	**1.00**
b_SPEC14_poisson	0.66	0.79	0.81	0.83	0.87	0.70	0.97	0.68	0.78	0.80	0.83	0.80	0.60	0.93	0.70	0.81	0.75	0.81	0.71	0.67	0.90	0.74	0.80	0.80	0.80	0.80	0.62	0.84	0.73	0.79	0.80	0.80	0.78	0.64	0.95
b_SPEC15_python36	0.59	0.62	0.71	0.86	0.87	0.90	0.90	0.60	0.61	0.66	0.85	0.98	0.72	0.82	0.59	0.66	0.82	0.71	0.82	0.67	0.84	0.61	0.69	0.71	0.66	0.78	0.72	0.89	0.61	0.65	0.75	0.70	0.75	0.71	0.73
b_SPEC16_rodiniaLifeSci	0.88	0.94	0.85	0.83	0.81	0.87	0.97	0.95	0.87	0.88	0.88	0.80	0.80	0.92	0.78	0.84	0.91	0.84	0.84	0.89	0.91	0.99	0.96	0.96	0.85	0.86	0.86	0.95	0.99	0.98	0.97	0.83	0.80	0.80	0.95
b_SPEC17_rodiniaCFD	0.91	0.88	0.88	0.98	0.92	0.80	0.94	0.96	0.93	0.92	0.98	0.97	0.80	0.85	0.74	0.71	0.83	0.86	0.89	0.81	0.85	0.86	0.88	0.89	0.83	0.89	0.80	0.91	0.94	0.92	0.95	0.91	0.94	0.80	0.77
b_SPEC18_smp	0.72	0.58	0.61	0.77	0.76	0.66	0.97	0.93	0.75	0.68	0.80	0.80	0.64	**1.00**	0.57	0.55	0.64	0.73	0.76	0.68	0.90	0.93	0.76	0.76	0.80	0.76	0.78	**1.00**	0.97	0.78	0.78	0.80	0.60	0.62	**1.00**

Figure 9.41: Percent of Total Correct Predictions for all combinations of Welch Window and Cutoff Frequency for 1-3s, 5s, 7s, 10s, and 15s for benign operations, with PSD conversion in dB.

9.4.3.1 Test Data

Test data for current-based side-channel analysis was collected separately from training data and consisted of the first half (only) of the test data set described in Section 8.3.1. Test data for this study was collected concurrently with hardware performance counter test data on the same system that generated the training data. Five runs each of the 35 operations were collected in randomized order, resulting in 175 test trials for classifier evaluation. One (SPEC 06) trial was found to be not viable during feature generation, so the resulting test data set consisted of 174 trials total.

Test Data Classification Accuracy and MCC Figures 9.42-9.45 show that the classifiers follow similar trends to cross validation (Figures 9.37-9.40), with classification accuracy lowest at 1 second and improving to the 3-4 second mark before levelling out until the 7-8 second mark and then either dropping off or dipping until close to 15 seconds. This is likely due to the generation of frequency features based on a window beginning at operation trigger time rather than operation start time, which means power measurements are not being compared at the same exact point in each operation. Insights from the training data verification performed in Section 8.1.1 by plotting hardware performance counter results (e.g. Figure 8.4) showed that most SPEC benchmarks could take 0-15 seconds to start the compute-intensive phase, while actual start of ransomware operations in particular could vary by as much as 15 seconds after execution. For the 7-12 second range of these results to provide any useful insight, a method to identify operation start time is required.

The higher accuracy of test data results (when compared to cross validation) may be attributed to the relatively small set of test data (5 trials of each operation) in comparison to the training data

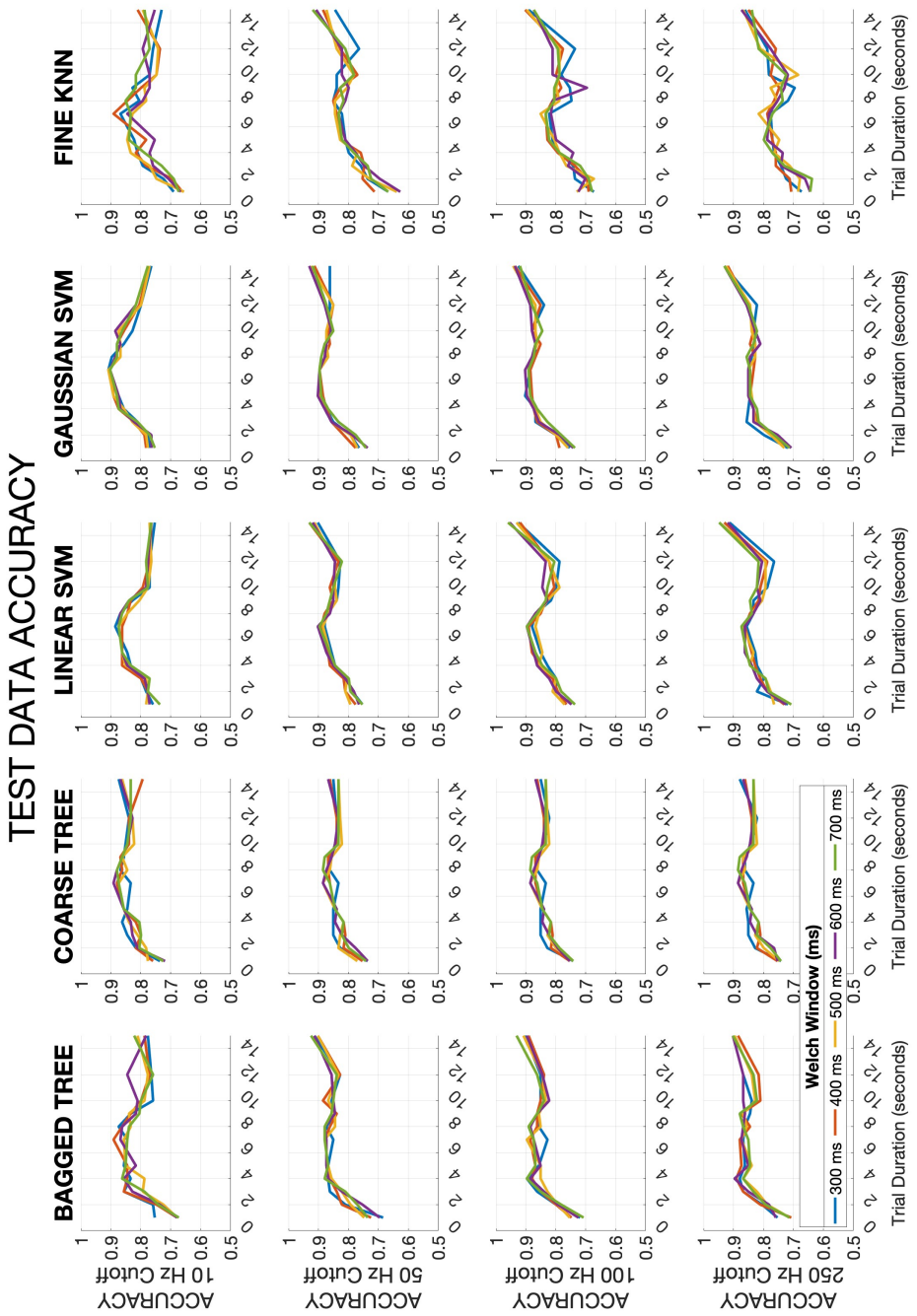


Figure 9.42: Test Data Classification Accuracy for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features.

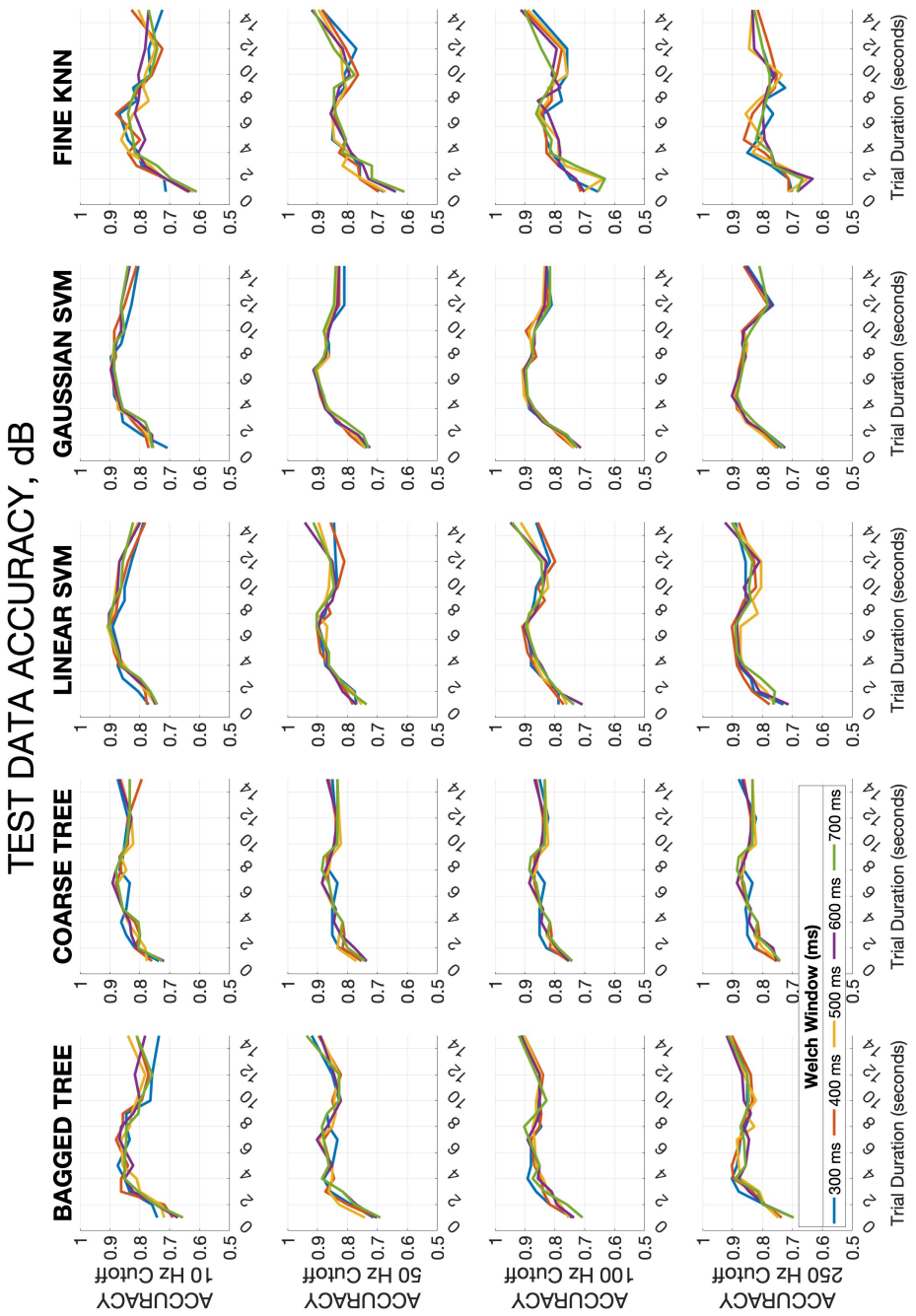


Figure 9.43: Test Data Classification Accuracy for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features in dB.

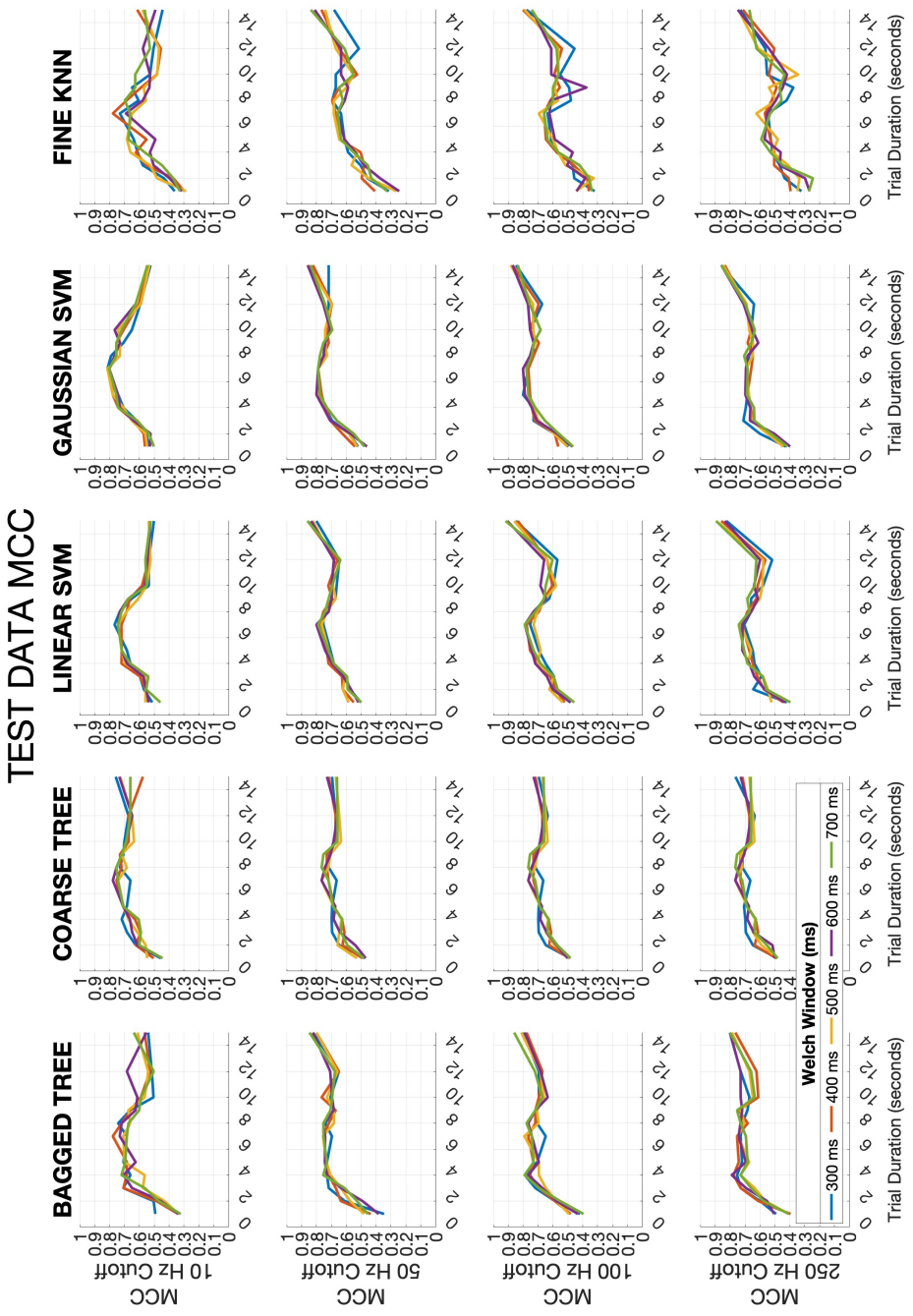


Figure 9.44: Test Data Classification MCC for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features.

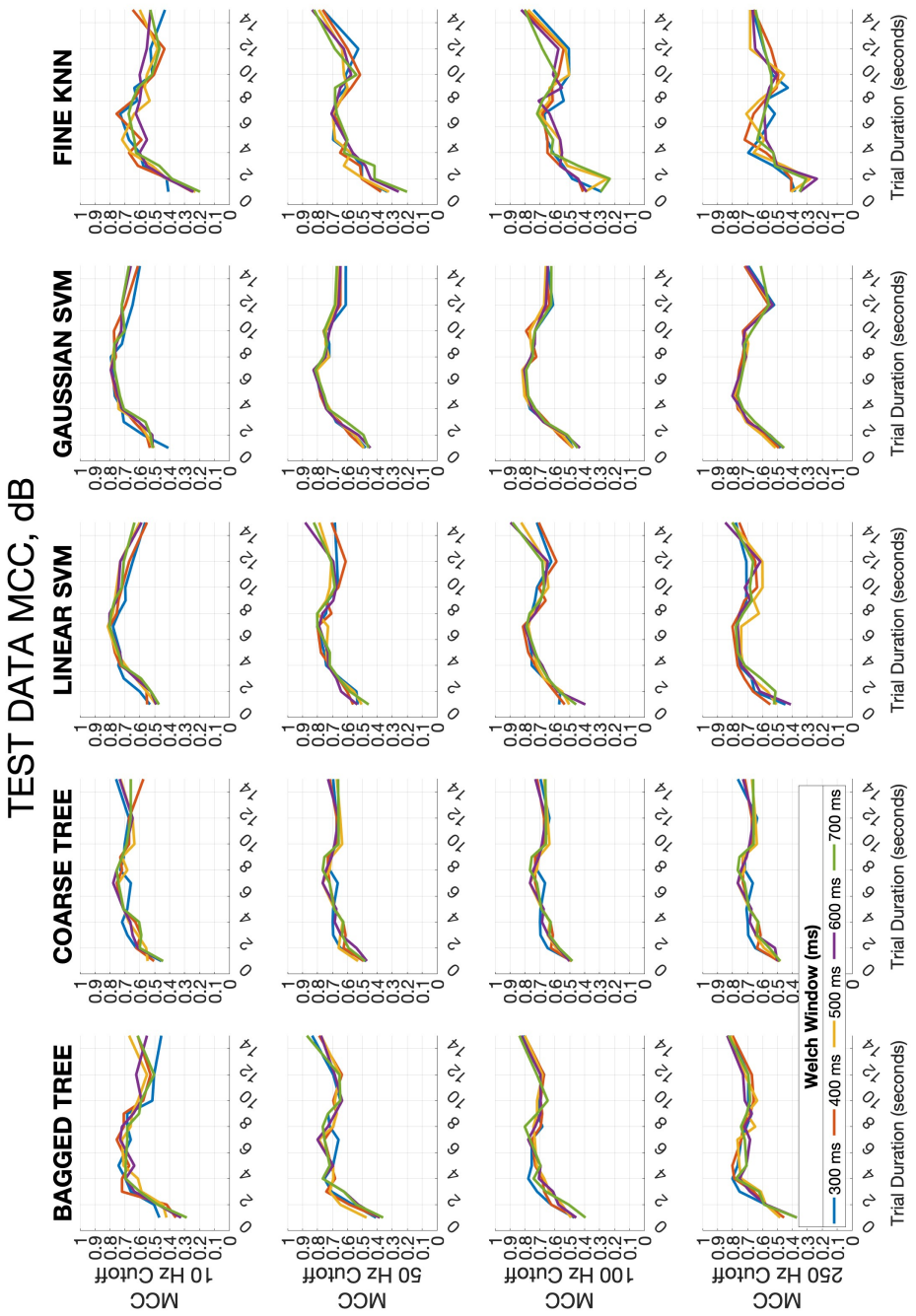


Figure 9.45: Test Data Classification MCC for all combinations of trial duration, Welch Window Size, and Cutoff Frequency Power Spectral Density features in dB.

set (20 trials of each operation). PSDs derived from current features converted to dB produced slightly better results overall, and both SVM classifiers and the Bagged Tree Ensemble achieved accuracy of 90% by 5 seconds for cutoff frequencies of 100 Hz and 250 Hz. Best case accuracy came in at 96% for a Linear SVM classifier with a 100 Hz cutoff and 700 ms Welch Window for 15 seconds of data, and all classifiers produced accuracy results $\geq$ 90% for some combination of parameters with 15 seconds of data. Figure 9.46 provides the top 3 classification results, ranked by MCC, for each trial duration segment length tested.

Linear SVM, Bagged Tree, and Medium Gaussian SVM classifiers accounted for 94% (all but 2) of the top 3 results for the eleven different trial segment durations, and dB power conversion generated the top result in 81% (9 of 11) cases. Otherwise, there was no clear best choice for Welch Window size or Cutoff Frequency. Top results were consistent with general observations from Figures 9.42-9.45.

9.5 PSCA Experiment Summary and Future Research Directions

In summary, Chapter 9 described two distinct proofs-of-concept, each of which led to a full classification study. All work in this Chapter accessed the power side-channel through current-draw obtained from a current-probe looped around the power lines directly connected to a solid state drive. The first proof-of-concept (Section 9.1) and study (Section 9.2) explored read, write, and idle operations on a variety of solid state drives, and demonstrated the ability to classify 28-ways across, drive, operation, and file size with $\sim$78% test accuracy for the same drive used to created the training data set, and $\sim$69% test accuracy for an identical, never before used SSD. The lessons learned from the first proof/study series study directly tied into the second proof-of-

Classifier	Segment Length (s)	Welch Window (ms)	Cutoff Freq (Hz)	dB	ACCURACY	PRECISION	RECALL	F1	MCC
LinearSVM	**1**	500	**50**		**0.7931**	**0.8141**	**0.7713**	**0.7921**	**0.5839**
LinearSVM	1	300	100	dB	0.7874	0.8051	0.7663	0.7852	0.5700
MedGaussianSVM	1	400	100		0.7874	0.7916	0.7727	0.7821	0.5640
LinearSVM	**2**	400	**250**	**dB**	**0.8333**	**0.8452**	**0.8180**	**0.8314**	**0.6626**
CoarseTree	2	500	50		0.8333	0.8342	0.8244	0.8293	0.6586
CoarseTree	2	500	50	dB	0.8333	0.8342	0.8244	0.8293	0.6586
BaggedTree	**3**	300	**250**	**dB**	**0.8793**	**0.8787**	**0.8745**	**0.8766**	**0.7532**
BaggedTree	3	400	50	dB	0.8736	0.8754	0.8663	0.8708	0.7416
MedGaussianSVM	3	300	100		0.8678	0.8704	0.8596	0.8650	0.7299
BaggedTree	**4**	300	**250**	**dB**	**0.9023**	**0.9000**	**0.9012**	**0.9006**	**0.8012**
BaggedTree	4	700	100		0.8966	0.8938	0.8962	0.8950	0.7900
BaggedTree	4	600	250		0.8966	0.8945	0.8945	0.8945	0.7891
MedGaussianSVM	**5**	500	**100**	**dB**	**0.9023**	**0.8994**	**0.9028**	**0.9011**	**0.8023**
BaggedTree	5	400	250	dB	0.9023	0.9066	0.8947	0.9007	0.8013
MedGaussianSVM	5	600	250	dB	0.9023	0.9000	0.9012	0.9006	0.8012
MedGaussianSVM	**7**	600	**50**	**dB**	**0.9138**	**0.9109**	**0.9162**	**0.9135**	**0.8270**
MedGaussianSVM	7	300	50	dB	0.9080	0.9052	0.9111	0.9082	0.8163
MedGaussianSVM	7	400	100	dB	0.9080	0.9052	0.9111	0.9082	0.8163
LinearSVM	**8**	700	**50**	**dB**	**0.9023**	**0.8994**	**0.9028**	**0.9011**	**0.8023**
LinearSVM	8	500	50	dB	0.9023	0.9000	0.9012	0.9006	0.8012
LinearSVM	8	600	10	dB	0.9023	0.9000	0.9012	0.9006	0.8012
MedGaussianSVM	**9**	400	**10**	**dB**	**0.8851**	**0.8821**	**0.8877**	**0.8849**	**0.7698**
MedGaussianSVM	9	700	10	dB	0.8851	0.8821	0.8877	0.8849	0.7698
LinearSVM	9	700	10	dB	0.8851	0.8819	0.8861	0.8840	0.7680
MedGaussianSVM	**10**	400	**100**	**dB**	**0.8966**	**0.8935**	**0.8978**	**0.8957**	**0.7913**
MedGaussianSVM	10	400	10	dB	0.8851	0.8827	0.8893	0.8860	0.7720
MedGaussianSVM	10	500	100	dB	0.8851	0.8822	0.8844	0.8833	0.7666
MedGaussianSVM	**12**	600	**100**		**0.8851**	**0.8927**	**0.8747**	**0.8836**	**0.7673**
MedGaussianSVM	12	700	100		0.8793	0.8882	0.8681	0.8780	0.7560
MedGaussianSVM	12	600	50		0.8793	0.8851	0.8697	0.8773	0.7546
LinearSVM	**15**	700	**100**		**0.9598**	**0.9616**	**0.9566**	**0.9591**	**0.9182**
LinearSVM	15	600	100		0.9540	0.9531	0.9531	0.9531	0.9063
LinearSVM	15	600	100	dB	0.9483	0.9459	0.9497	0.9478	0.8956

Figure 9.46: Top 3 results for each trial segment length for eleven different durations (ranging from 1-15 seconds) tested. Results are ranked from low-to-high segment length and then by high-to-low MCC.

concept and study, which examined the ability to distinguish and classify ransomware operations from benign encryption, SPEC benchmarks, and system idle.

The proof-of-concept to collect current supplied to the disk while ransomware was executing (Section 9.3) demonstrated the ability to classify 3 ransomware samples vs. benign 7zip encryption and system idle using power side-channel analysis of current-draw measurements alone, a larger study ensued (Section 9.4). After testing 32 non-optimized classification models with frequency features producing by varying a series of parameters, five were selected based on overall classification accuracy for further analysis and the scope of parameters refined. The best case results of this study using 35-class Current-draw-based Side-channel Analysis for Ransomware Classification showed Bagged Tree Ensemble, Linear Support Vector Machines, and Gaussian Support Vector Machine consistently produced the best results. The Linear Support Vector Machine produced classification accuracy over 95% for 15 seconds of data, although the time duration required to achieve those values is sub-optimal for time sensitive identification of ransomware encryption. Conversely, classification accuracy at 2 seconds was just over 83%, which indicates the power side-channel alone may provide unsatisfactory results, as many ransomware executables begin encryption operations within $\sim$2 seconds.

Four seconds of data were required to attain a best case classification accuracy greater than 90%, which improved marginally to the 7 second mark, before dropping below 90% until close to 15 seconds. This is likely due to the generation of frequency features based on a window beginning at operation trigger time, which is not necessarily equivalent to operation start time. Identifying actual operation start through current-draw is left as future work. Additionally, the current data for the main ransomware study (Section 9.4) was collected in conjunction with the micro-architectural side-channel accessed via hardware performance counters (Chapter 8), in

order to facility future multi-channel analysis, which also requires synchronization of feature start time across side-channels.

Of note, all classifiers were trained and tested without any parameter optimization, again with the assumption that results would improve with selection of a single classifier and optimization of configurable parameters. It was also noted that while modest variations in classification accuracy and MCC exited based on power spectral density feature generation parameters of dB unit, Welch window size, and cutoff frequency, the trends were consistent within each classification algorithm. Optimization of classification algorithms and frequency features to produce an ideal result is left as future work.

Overall, classification results indicate the power side-channel is not ideal as a stand-alone ransomware detection option, but could be beneficial for low overhead systems with sufficient accuracy tolerance. Another useful application could be as a honeypot-like machine in a network configuration where a machine on an edge-node could detect ransomware operations with high confidence in $\sim$15 seconds, other local machines could be notified before compromise. Additionally, classification features obtained from the power side-channel could be extremely useful as supplemental, complementary features in a multi-channel classification approach.

Chapter 10: Conclusions and Future Work

This work considered measurements from three different side-channels (temperature, power, micro-architectural) for their effectiveness in accurately differentiating between ransomware and benign operations (background OS activity, 7zip encryption, and SPEC benchmarks) on a non-virtualized system over a given time duration, with Matthews Correlation Coefficient (MCC) used to measure performance of five selected machine learning classification algorithms. Specifically, this work evaluated each side-channel analysis method based on classification results using two main criteria: (1) The highest possible accuracy in (2) lowest possible time, with $\sim$2 seconds identified as the goal threshold since ransomware encryption operations generally begin by then. Side-channel methods are particularly advantageous as a complement to traditional ransomware detection techniques as they tend to be resilient to zero-day threats. This is due to side-channel-based approaches considering overall system behavior (focusing on "normal" vs. "bad" behavior) rather than being process and signature-based, which requires the system to have previously been exposed to specific ransomware (and is, by definition, not possible in a zero-day scenario).

The temperature side-channel, accessed through (off-system) thermal imaging, showed some promise for its ability to provide insight in two dimensions with no on-device overhead, was ruled out for ransomware detection due to the low-pass filtering behavior with cutoff frequencies above the low kHz range, resulting in time delays on a scale that would preclude it from identifying

ransomware in the $\sim$2 second window before encryption operations begin.

The power side-channel, accessed through measuring current draw of the drive power supply lines, generated best-case classification accuracy results of 95% (0.92 MCC) with a Linear Support Vector Machine Classifier and accuracy $\geq$ 90% (MCC $\geq$ 0.8) for all five classifiers tested, but 15 seconds of current were required to obtain that number. Tests demonstrated at least four seconds of data were required to attain a best case classification accuracy greater than 90%, and accuracy at 2 seconds was just over 83%. While some applications may be able to tolerate just 83% ransomware detection accuracy, the power side-channel was generally not effective in a time period suitable to detect ransomware operations before encryption begins. Of note, no classification models in this work were optimized, so it is possible that parameter optimization combined with the use of time-series models could reduce the amount of data needed to accurately distinguish between benign operations and ransomware.

Of the three side-channels tested, the micro-architectural side-channel, accessed through (on-system) hardware performance counters, provided the highest accuracy results in the shortest period of time. Over 230 hardware events, an order of magnitude greater than related ransomware literature, were collected, tested, and ranked by their contribution to overall classifier performance. Fine KNN and Narrow Neural Network classifiers consistently provided the best results across all classifiers at three different stages of the operation tested. Each classification algorithm was found to have its own combination of performance counter features that generated best performance results, and these features could be further optimized by detection time. Examination of these results showed that, despite the quantity of features collected, classifier performance only marginally improved after 6 HPC features for $\leq$ 2 seconds, with MCC $\geq$ 0.9 for 1 second of data for 3 of 5 classifiers tested with just 4-6 performance counter features, and a best-case MCC of 0.98 for a

FineKNN classifier with 1 second of data and 4 HPC features. MCC results for shortest duration event window (0.1s) were found to be within 0-7% of the best case MCC result window (1-2 s) for each classifier, indicating that ransomware can be classified using only a tenth of a second of 4-6 performance event features with greater than 90% accuracy for four of five classifiers tested, which makes the implementation of this approach in a real-time ransomware detector feasible.

Preliminary results also show that using features from both the power and micro-architectural side-channels would increase classifier performance. The implementation of a multi-channel approach requires identification of actual operation start time across both side-channels, and is suggested for future work. Additional future research directions are included in Section 10.1, below. While these techniques for side-channel analysis of multiple channels were specifically applied to the classification and detection of ransomware, this approach could be applied to any number of generalized or specific applications where out-of-band measurements provide insight into system activity.

10.1 Future Work

Beginning with the implementation the aforementioned multi-channel classification approach, possible future efforts are as follows:

- Feature generation and ransomware classification based on integration of features obtained from multiple side-channels (e.g. micro-architectural, power, and temperature). This could supplement and complement existing results and requires the fusion of disparate data sources, since performance counters are measured on the CPU but power is measured at the SSD. This requires the development of a method to identify actual operation start

time across disparate side-channel collection traces and synchronize feature generation. It could also include optimizing performance counter feature rankings specifically for smaller windows of time to facilitate the development of an automated start-time detection capability across various channels. This could mitigate the poor 0.1s window "actual operation" start time results and is likely necessary to realize a real-time detection capability.

- Increase robustness of the data set, to include additional ransomware families and insertion of new, unseen/untrained ransomware in the test data set, to evaluate effectiveness in classifying potential zero-day threats. Additional "typical" benign operations should be added to the training data set as well. While SPEC benchmarks were chosen for their computational intensity, a drawback is that they may overemphasize certain performance counters due to that same computational intensity. While addition of benign samples contributes additional imbalance to the data set, it can be mitigated by considering MCC as the true indicator of classifier performance. Furthermore, generating new variants of ransomware could further delineate which features best distinguish ransomware from benign behavior, as well as help re-balance the data set.

- Additional classification model analysis, focusing on time series classification methods and collection of a reduced set of the top ranked performance counters, optimized power spectral density features, or/and multiple channel features in real-time. As existing training and testing was performed with non-optimized classification models, results are likely to improve with the selection of a single model and optimization of its parameters.

- Development of an integrated, multi-channel, real-time detection and classification capability, with a lightweight version of "VTune-like" software to collect time-series performance

counter data, integrate power measurements, and generate near-real-time classification decisions. Realization of this capability requires aforementioned Future Work items to be accomplished first.

- Consideration of a network configuration where a honeypot-like machine placed on an edge-node could detect ransomware (or other anomalous) operations with high confidence in $\sim$15 seconds. This could be particularly useful for power or temperature side-channel-based approaches for IoT configurations where devices are unlikely to have onboard resources required for constant internal monitoring. In this case, although the honeypot-like device would be compromised, it could potentially notify other local devices before the entire network is compromised.

- Leverage performance counters or SMART (Self-Monitoring, Analysis, and Reporting Technology) counters onboard SSDs to provide direct comparison of power and micro-architectural side-channel results on the same device.

- Other possible future work directions include applications to Data Version Control and Natural Language Processing.

Appendix A: Table of Vtune Hardware Performance Counter Names, Descriptions, and associated Feature Numbers for each Study, adapted from [1]

Table A.1: Overview of HPCs used in this study.

Study 2 *31-class*	**Study 1** *23-class*	**Study 3** *35-class*	**HPC Name**	**HPC Description**
1	1	-	ARITH. DIVIDER_ACTIVE	Cycles when divide unit is busy executing divide or square root operations.
2	2	-	BACLEARS. ANY	Counts the number of times the front-end is resteered when it finds a branch instruction in a fetch line.
3	3	-	BR_INST_RETIRED. ALL_BRANCHES_PS	This is a precise version of BR_INST_RETIRED. ALL_BRANCHES that counts all (macro) branch instructions retired.
5	X	-	BR_INST_RETIRED. COND_NTAKEN	This event counts not taken branch instructions retired.
4	X	1	BR_INST_RETIRED. CONDITIONAL_PS	This is a precise version of the event that counts conditional branch instructions retired.
6	X	2	BR_INST_RETIRED. FAR_BRANCH_PS	This is a precise version of the event that counts far branch instructions retired.

Table A.2: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
7	X	3	BR_INST_RETIRED. NEAR_CALL_PS	This is a precise version of the event that counts both direct and indirect near call instructions retired.
8	X	4	BR_INST_RETIRED. NEAR_RETURN_PS	This is a precise version of the event that counts return instructions retired.
9	X	5	BR_INST_RETIRED. NEAR_TAKEN_PS	This is a precise version of the event that counts taken branch instructions retired.
10	X	6	BR_INST_RETIRED. NOT_TAKEN	This event counts not taken branch instructions retired.
11	4	7	BR_MISP_RETIRED. ALL_BRANCHES_PS	This is a precise version of BR_MISP_RETIRED .ALL_BRANCHES that counts all mispredicted macro branch instructions retired.
12	X	8	BR_MISP_RETIRED. CONDITIONAL_PS	This is a precise version of the event that counts mispredicted conditional branch instructions retired.
13	X	9	BR_MISP_RETIRED. NEAR_CALL_PS	This event counts both taken and not taken retired mispredicted direct and indirect near calls, including both register and memory indirect.
14	X	10	BR_MISP_RETIRED. NEAR_TAKEN_PS	Number of near branch instructions retired that were mispredicted and taken.

Table A.3: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
15	X	11	CPU_CLK_THREAD UNHALTED. ONE_THREAD_ACTIVE	Core crystal clock cycles when this thread is unhalted and the other thread is halted.
16	X	-	CPU_CLK_THREAD UNHALTED. REF_XCLK	Core crystal clock cycles when the thread is unhalted.
17	X	-	CPU_CLK_THREAD UNHALTED. REF_XCLK_ANY	Core crystal clock cycles when at least one thread on the physical core is unhalted.
18	5	12	CPU_CLK_UNHALTED. ONE_THREAD_ACTIVE	Core crystal clock cycles when this thread is unhalted and the other thread is halted.
19	6	13	CPU_CLK_UNHALTED. REF_TSC	Counts the number of reference cycles when the core is not in a halt state.
20	7	-	CPU_CLK_UNHALTED. REF_XCLK	Core crystal clock cycles when the thread is unhalted.
21	X	-	CPU_CLK_UNHALTED. REF_XCLK_ANY	Core crystal clock cycles when at least one thread on the physical core is unhalted.
22	X	-	CPU_CLK_UNHALTED. RING0_TRANS	Counts when the Current Privilege Level (CPL) transitions to ring 0.
X	8	-	CPU_CLK_UNHALTED. THREAD	Counts the number of core cycles while the thread is not in a halt state.
23	8	-	CPU_CLK_UNHALTED. THREAD_ANY	Core cycles when at least one thread on the physical core is not in halt state.
24	9	-	CPU_CLK_UNHALTED. THREAD_P	his is an architectural event that counts the number of thread cycles while the thread is not in a halt state.

Table A.4: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
25	X	-	CPU_CLK_UNHALTED. THREAD_P_ANY	Core cycles when at least one thread on the physical core is not in halt state.
26	10	14	CYCLE_ACTIVITY. CYCLES_L1D_MISS	Cycles while L1 cache miss demand load is outstanding.
27	X	-	CYCLE_ACTIVITY. CYCLES_L2_MISS	Cycles while L2 cache miss demand load is outstanding.
28	X	-	CYCLE_ACTIVITY. CYCLES_L3_MISS	Cycles while L3 cache miss demand load is outstanding.
29	11	-	CYCLE_ACTIVITY. CYCLES_MEM_ANY	Cycles while memory subsystem has an outstanding load.
30	12	15	CYCLE_ACTIVITY. STALLS_L1D_MISS	Execution stalls while L1 cache miss demand load is outstanding.
31	13	16	CYCLE_ACTIVITY. STALLS_L2_MISS	Execution stalls while L2 cache miss demand load is outstanding.
32	14	-	CYCLE_ACTIVITY. STALLS_L3_MISS	Execution stalls while L3 cache miss demand load is outstanding.
33	15	-	CYCLE_ACTIVITY. STALLS_MEM_ANY	Execution stalls while memory subsystem has an outstanding load.
34	16	-	CYCLE_ACTIVITY. STALLS_TOTAL	Total execution stalls.
35	X	17	DSB2MITE_SWITCHES. COUNT	This event counts the number of the Decode Stream Buffer (DSB)-to-MITE switches including all misses because of missing Decode Stream Buffer (DSB) cache and u-arch forced misses.

Table A.5: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
36	17	18	DSB2MITE_SWITCHES. PENALTY_CYCLES	Counts Decode Stream Buffer (DSB)-to-MITE switch true penalty cycles.
37	X	19	DTLB_LOAD_MISSES. MISS_CAUSES_A_WALK	Counts demand data loads that caused a page walk of any page size (4K/2M/4M/1G).
38	18	-	DTLB_LOAD_MISSES. STLB_HIT	Counts loads that miss the DTLB (Data TLB) and hit the STLB (Second level TLB).
39	19	20	DTLB_LOAD_MISSES. WALK_ACTIVE	Counts cycles when at least one PMH (Page Miss Handler) is busy with a page walk for a load.
40	X	21	DTLB_LOAD_MISSES. WALK_COMPLETED	Counts completed page walks (all page sizes) caused by demand data loads.
41	X	-	DTLB_LOAD_MISSES. WALK_COMPLETED_2M_4M	Counts completed page walks (2M/4M sizes) caused by demand data loads.
42	X	-	DTLB_LOAD_MISSES. WALK_COMPLETED_4K	Counts completed page walks (4K sizes) caused by demand data loads.
43	X	22	DTLB_LOAD_MISSES. WALK_PENDING	Counts 1 per cycle for each PMH that is busy with a page walk for a load.
44	X	23	DTLB_STORE_MISSES. MISS_CAUSES_A_WALK	Counts demand data stores that caused a page walk of any page size (4K/2M/4M/1G).
45	20	-	DTLB_STORE_MISSES. STLB_HIT	Stores that miss the DTLB (Data TLB) and hit the STLB (2nd Level TLB).
46	21	24	DTLB_STORE_MISSES. WALK_ACTIVE	Counts cycles when at least one PMH (Page Miss Handler) is busy with a page walk for a store.

Table A.6: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
47	X	25	DTLB_STORE_MISSES. WALK_COMPLETED	Counts completed page walks (all page sizes) caused by demand data stores.
48	X	-	DTLB_STORE_MISSES. WALK_COMPLETED_2M_4M	Counts completed page walks (2M/4M sizes) caused by demand data stores.
49	X	-	DTLB_STORE_MISSES. WALK_COMPLETED_4K	Counts completed page walks (4K sizes) caused by demand data stores.
50	X	26	DTLB_STORE_MISSES. WALK_PENDING	Counts 1 per cycle for each PMH that is busy with a page walk for a store.
51	22	-	EXE_ACTIVITY. 1_PORTS_UTIL	Counts cycles during which a total of 1 uop was executed on all ports and Reservation Station (RS) was not empty.
52	23	-	EXE_ACTIVITY. 2_PORTS_UTIL	Counts cycles during which a total of 2 uops were executed on all ports and Reservation Station (RS) was not empty.
53	X	-	EXE_ACTIVITY. 3_PORTS_UTIL	Cycles total of 3 uops are executed on all ports and Reservation Station (RS) was not empty.
54	X	-	EXE_ACTIVITY. 4_PORTS_UTIL	Cycles total of 4 uops are executed on all ports and Reservation Station (RS) was not empty.
55	24	-	EXE_ACTIVITY. BOUND_ON_STORES	Cycles where the Store Buffer was full and no outstanding load.

Table A.7: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
56	X	-	EXE_ACTIVITY. EXE_BOUND_0_PORTS	Counts cycles during which no uops were executed on all ports and Reservation Station (RS) was not empty.
57	25	27	FP_ARITH_INST_RETIRED. 128B_PACKED_DOUBLE	Counts once for most SIMD 128-bit packed computational double precision floating-point instructions retired.
58	26	28	FP_ARITH_INST_RETIRED. 128B_PACKED_SINGLE	Counts once for most SIMD 128-bit packed computational single precision floating-point instructions retired.
59	27	29	FP_ARITH_INST_RETIRED. 256B_PACKED_DOUBLE	Counts once for most SIMD 256-bit packed double computational precision floating-point instructions retired.
60	28	30	FP_ARITH_INST_RETIRED. 256B_PACKED_SINGLE	Counts once for most SIMD 256-bit packed single computational precision floating-point instructions retired.
61	29	31	FP_ARITH_INST_RETIRED. SCALAR_DOUBLE	Counts once for most SIMD scalar computational double precision floating-point instructions retired.
62	30	-	FP_ARITH_INST_RETIRED. SCALAR_SINGLE	Counts once for most SIMD scalar computational single precision floating-point instructions retired
63	X	-	FP_ASSIST. ANY	Counts cycles with any input and output SSE or x87 FP assist.

Table A.8: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
X	31	-	FRONTEND_RETIRED. ANY_DSB_MISS	Counts retired Instructions that experienced DSB (Decode stream buffer i.e. the decoded instruction-cache) miss.
64	32	-	FRONTEND_RETIRED. DSB_MISS_PS	Number of retired Instructions that experienced a critical DSB (Decode stream buffer i.e. the decoded instruction-cache) miss.
65	X	-	FRONTEND_RETIRED. ITLB_MISS_PS	Counts retired Instructions that experienced iTLB (Instruction TLB) true miss.
66	X	32	FRONTEND_RETIRED. L1I_MISS_PS	Retired Instructions who experienced Instruction L1 Cache true miss. Precise Event.
67	33	33	FRONTEND_RETIRED. L2_MISS_PS	Retired Instructions who experienced Instruction L2 Cache true miss. Precise Event.
68	X	-	FRONTEND_RETIRED. LATENCY_GE_1	Retired instructions that are fetched after an interval where the front-end delivered no uops for a period of at least 1 cycle which was not interrupted by a back-end stall.
69	X	-	FRONTEND_RETIRED. LATENCY_GE_128_PS	Retired instructions that are fetched after an interval where the front-end delivered no uops for a period of 128 cycles which was not interrupted by a back-end stall.

Table A.9: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
70	34	34	FRONTEND_RETIRED. LATENCY_GE_16_PS	Counts retired instructions that are delivered to the back-end after a front-end stall of at least 16 cycles.
71	X	35	FRONTEND_RETIRED. LATENCY_GE_2	Retired instructions that are fetched after an interval where the front-end delivered no uops for a period of 2 cycles which was not interrupted by a back-end stall.
73	35	36	FRONTEND_RETIRED. LATENCY GE_2_BUBBLES_GE_1_PS	Counts retired instructions that are delivered to the back-end after the front-end had at least 1 bubble-slot for a period of 2 cycles.
74	X	-	FRONTEND_RETIRED. LATENCY GE_2_BUBBLES_GE_2_PS	Retired instructions that are fetched after an interval where the front-end had at least 2 bubble-slots for a period of 2 cycles which was not interrupted by a back-end stall.
75	X	-	FRONTEND_RETIRED. LATENCY_GE_2_PS	Retired instructions that are fetched after an interval where the front-end delivered no uops for a period of 2 cycles which was not interrupted by a back-end stall.
72	X	-	FRONTEND_RETIRED. LATENCY_GE_256_PS	Retired instructions that are fetched after an interval where the front-end delivered no uops for a period of 256 cycles which was not interrupted by a back-end stall.

Table A.10: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
76	X	-	FRONTEND_RETIRED. LATENCY_GE_32_PS	Counts retired instructions that are delivered to the back-end after a front-end stall of at least 32 cycles.
77	36	-	FRONTEND_RETIRED. LATENCY_GE_4_PS	Retired instructions that are fetched after an interval where the front-end delivered no uops for a period of 4 cycles which was not interrupted by a back-end stall.
78	X	-	FRONTEND_RETIRED. LATENCY_GE_512_PS	Retired instructions that are fetched after an interval where the front-end delivered no uops for a period of 512 cycles which was not interrupted by a back-end stall.
79	X	-	FRONTEND_RETIRED. LATENCY_GE_64_PS	Retired instructions that are fetched after an interval where the front-end delivered no uops for a period of 64 cycles which was not interrupted by a back-end stall.
80	X	-	FRONTEND_RETIRED. LATENCY_GE_8_PS	Counts retired instructions that are delivered to the back-end after a front-end stall of at least 8 cycles.
81	37	37	FRONTEND_RETIRED. STLB_MISS_PS	Counts retired Instructions that experienced STLB (2nd level TLB) true miss
82	X	-	HW_INTERRUPTS. RECEIVED	Counts the number of hardware interruptions received by the processor.
83	38	38	ICACHE_16B. IFDATA_STALL	Cycles where a code line fetch is stalled due to an L1 instruction cache miss.

Table A.11: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
X	39		ICACHE_16B. IFDATA_STALL:cmask=1:e=yes	Cycles where a code line fetch is stalled due to an L1 instruction cache miss.
84	X	39	ICACHE_64B. IFTAG_HIT	Instruction fetch tag lookups that hit in the instruction cache (L1I).
85	X	40	ICACHE_64B. IFTAG_MISS	Instruction fetch tag lookups that miss in the instruction cache (L1I).
86	40	41	ICACHE_64B. IFTAG_STALL	Cycles where a code fetch is stalled due to L1 instruction cache tag miss.
100	49		IDQ_UOPS_NOT_DELIVERED. CORE	Counts the number of uops not delivered to Resource Allocation Table (RAT) per thread.
101	50		IDQ_UOPS_NOT_DELIVERED. CYCLES_0_UOPS_DELIV. CORE	Counts cycles when no uops are delivered to Resource Allocation Table (RAT).
102	X	47	IDQ_UOPS_NOT_DELIVERED. CYCLES_FE_WAS_OK	Counts cycles FE delivered 4 uops or Resource Allocation Table (RAT) was stalling FE.
103	X		IDQ_UOPS_NOT_DELIVERED. CYCLES_LE_1_UOP_DELIV. CORE	Counts cycles when less than 1 uop is delivered to Resource Allocation Table (RAT).
104	X		IDQ_UOPS_NOT_DELIVERED. CYCLES_LE_2_UOP_DELIV. CORE	Cycles with less than 2 uops delivered by the front-end.
105	X		IDQ_UOPS_NOT_DELIVERED. CYCLES_LE_3_UOP_DELIV. CORE	Cycles with less than 3 uops delivered by the front-end.
87	41	-	IDQ. ALL_DSB_CYCLES_4_UOPS	Counts the number of cycles 4 uops were delivered to Instruction Decode Queue (IDQ) from the Decode Stream Buffer (DSB) path.

Table A.12: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
88	42	-	IDQ. ALL_DSB_CYCLES_ANY_UOPS	Counts the number of cycles uops were delivered to Instruction Decode Queue (IDQ) from the Decode Stream Buffer (DSB) path.
89	43	42	IDQ. ALL_MITE_CYCLES_4_UOPS	Counts the number of cycles 4 uops were delivered to the Instruction Decode Queue (IDQ) from the MITE (legacy decode pipeline) path.
90	44	43	IDQ. ALL_MITE_CYCLES_ANY_UOPS	Counts the number of cycles uops were delivered to the Instruction Decode Queue (IDQ) from the MITE (legacy decode pipeline) path.
91	X	-	IDQ. DSB_CYCLES	Counts cycles during which uops are being delivered to Instruction Decode Queue (IDQ) from the Decode Stream Buffer (DSB) path.
92	45	44	IDQ. DSB_UOPS	Counts the number of uops delivered to Instruction Decode Queue (IDQ) from the Decode Stream Buffer (DSB) path.
93	X	-	IDQ. MITE_CYCLES	Counts cycles during which uops are being delivered to Instruction Decode Queue (IDQ) from the MITE path.

Table A.13: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
94	46	-	IDQ. MITE_UOPS	Counts the number of uops delivered to Instruction Decode Queue (IDQ) from the MITE path.
95	X	-	IDQ. MS_CYCLES	Counts cycles during which uops are being delivered to Instruction Decode Queue (IDQ) while the Microcode Sequencer (MS) is busy.
96	X	-	IDQ. MS_DSB_CYCLES	Counts cycles during which uops initiated by Decode Stream Buffer (DSB) are being delivered to Instruction Decode Queue (IDQ) while the Microcode Sequencer (MS) is busy.
97	X	45	IDQ. MS_MITE_UOPS	Counts the number of uops initiated by MITE and delivered to Instruction Decode Queue (IDQ) while the Microcode Sequencer (MS) is busy.
98	47	-	IDQ. MS_SWITCHES	Number of switches from DSB (Decode Stream Buffer) or MITE (legacy decode pipeline) to the Microcode Sequencer.
99	48	46	IDQ. MS_UOPS	Counts the total number of uops delivered by the Microcode Sequencer (MS).

Table A.14: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
106	51	-	ILD_STALL. LCP	Counts cycles that the Instruction Length decoder (ILD) stalls occurred due to dynamically changing prefix length of the decoded instruction.
107	52	48	INST_RETIRED. ANY	Counts the number of instructions retired from execution.
108	X	-	INST_RETIRED. ANY_P	Counts the number of instructions (EOMs) retired.
X	53	-	INST_RETIRED. NOP	Number of all retired NOP instructions.
109	54	-	INST_RETIRED. PREC_DIST	A version of INST_RETIRED that allows for a more unbiased distribution of samples across instructions retired.
110	X	49	INST_RETIRED. TOTAL_CYCLES_PS	Number of cycles using an always true condition applied to PEBS instructions retired event.
111	55	50	INT_MISC. CLEAR_RESTEER_CYCLES	Cycles the issue-stage is waiting for front-end to fetch from resteered path following branch misprediction or machine clear events.
112	56	51	INT_MISC. RECOVERY_CYCLES	Core cycles the Resource allocator was stalled due to recovery from an earlier branch misprediction or machine clear event.

Table A.15: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
113	X	-	INT_MISC. RECOVERY_CYCLES_ANY	Core cycles the allocator was stalled due to recovery from earlier clear event for any thread running on the physical core.
115	X	52	ITLB_MISSES. MISS_CAUSES_A_WALK	Counts page walks of any page size (4K/2M/4M/1G) caused by a code fetch.
116	X	-	ITLB_MISSES. STLB_HIT	Instruction fetch requests that miss the ITLB and hit the STLB.
117	X	53	ITLB_MISSES. WALK_ACTIVE	Cycles when at least one PMH is busy with a page walk for code (instruction fetch) request.
118	X	54	ITLB_MISSES. WALK_COMPLETED	Counts completed page walks (all page sizes) caused by a code fetch.
119	X	-	ITLB_MISSES. WALK_COMPLETED_2M_4M	Counts completed page walks (2M/4M page sizes) caused by a code fetch.
120	X	-	ITLB_MISSES. WALK_COMPLETED_4K	Counts completed page walks (4K page sizes) caused by a code fetch.
121	X	55	ITLB_MISSES. WALK_PENDING	Counts 1 per cycle for each PMH (Page Miss Handler) that is busy with a page walk for an instruction fetch request.
114	X		ITLB. ITLB_FLUSH	Counts the number of flushes of the big or small ITLB pages.
123	57	57	L1D_PEND_MISS. FB_FULL	Number of times a request needed a FB (Fill Buffer) entry but there was no entry available for it.

Table A.16: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
X	57		L1D_PEND_MISS. FB_FULL:cmask=1	Number of times a request needed a FB (Fill Buffer) entry but there was no entry available for it.
124	58	58	L1D_PEND_MISS. PENDING	Counts duration of L1D miss outstanding.
125	X	-	L1D_PEND_MISS. PENDING_CYCLES	Counts duration of L1D miss outstanding in cycles.
126	X	-	L1D_PEND_MISS. PENDING_CYCLES_ANY	Cycles with L1D load Misses outstanding from any thread on physical core.
122	X	56	L1D. REPLACEMENT	Counts L1D data line replacements including opportunistic replacements and replacements that require stall-for-replace or block-for-replace.
127	X	59	L2_LINES_IN. ALL	Counts the number of L2 cache lines filling the L2. Counting does not cover rejects.
128	X	60	L2_LINES_OUT. NON_SILENT	Counts the number of lines that are evicted by L2 cache when triggered by an L2 cache fill.
129	X	-	L2_LINES_OUT. SILENT	Counts the number of lines that are silently dropped by L2 cache when triggered by an L2 cache fill.
130	X	-	L2_LINES_OUT. USELESS_HWPF	Counts the number of lines that have been hardware prefetched but not used and now evicted by L2 cache
131	X	61	L2_LINES_OUT. USELESS_PREF	This event is deprecated.

Table A.17: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
132	X	-	L2_RQSTS. ALL_CODE_RD	Counts the total number of L2 code requests.
133	X	-	L2_RQSTS. ALL_DEMAND_DATA_RD	Counts the number of demand Data Read requests (including requests from L1D hardware prefetchers).
134	X	-	L2_RQSTS. ALL_DEMAND_MISS	Demand requests that miss L2 cache.
135	X	62	L2_RQSTS. ALL_DEMAND_REFERENCES	Demand requests to L2 cache.
136	X	63	L2_RQSTS. ALL_PF	Counts the total number of requests from the L2 hardware prefetchers.
137	59	-	L2_RQSTS. ALL_RFO	Counts the total number of RFO (read for ownership) requests to L2 cache.
138	X	-	L2_RQSTS. CODE_RD_HIT	Counts L2 cache hits when fetching instructions.
139	X	64	L2_RQSTS. CODE_RD_MISS	Counts L2 cache misses when fetching instructions.
140	X	65	L2_RQSTS. DEMAND_DATA_RD_HIT	Counts the number of demand Data Read requests that are initiated by load instructions that hit L2 cache
141	X	-	L2_RQSTS. DEMAND_DATA_RD_MISS	Counts the number of demand Data Read requests that miss L2 cache.
142	X	66	L2_RQSTS. MISS	All requests that miss L2 cache.

Table A.18: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
143	X	67	L2_RQSTS. PF_HIT	Counts requests from the L1/L2/L3 hardware prefetchers or Load software prefetches that hit L2 cache.
144	X	68	L2_RQSTS. PF_MISS	Counts requests from the L1/L2/L3 hardware prefetchers or Load software prefetches that miss L2 cache.
145	X	69	L2_RQSTS. REFERENCES	All L2 requests.
146	60	70	L2_RQSTS. RFO_HIT	Counts the RFO (Read-for-Ownership) requests that hit L2 cache.
147	X	71	L2_RQSTS. RFO_MISS	Counts the RFO (Read-for-Ownership) requests that miss L2 cache.
148	X	72	L2_TRANS. L2_WB	Counts L2 writebacks that access L2 cache.
151	63	-	LD_BLOCKS_PARTIAL. ADDRESS_ALIAS	Counts false dependencies in MOB when the partial comparison upon loose net check and dependency was resolved by the Enhanced Loose net mechanism.
149	61	-	LD_BLOCKS. NO_SR	The number of times that split load operations are temporarily blocked because all resources for handling the split accesses are in use.
150	62	-	LD_BLOCKS. STORE_FORWARD	Counts the number of times where store forwarding was prevented for a load operation.

Table A.19: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
152	X	-	LOAD_HIT_PRE. SW_PF	Counts all not software-prefetch load dispatches that hit the fill buffer (FB) allocated for the software prefetch.
153	X	73	LONGEST_LAT_CACHE. MISS	Counts core-originated cacheable requests that miss the L3 cache (Longest Latency cache).
154	X	74	LONGEST_LAT_CACHE. REFERENCE	Counts core-originated cacheable requests to the L3 cache (Longest Latency cache).
155	65	-	MACHINE_CLEARS. COUNT	Number of machine clears (nukes) of any type.
156	X	75	MACHINE_CLEARS. MEMORY_ORDERING	Counts the number of memory ordering Machine Clears detected.
157	X	76	MACHINE_CLEARS. SMC	Counts self-modifying code (SMC) detected.
158	X	77	MEM_INST_RETIRED. ALL_LOADS_PS	Stores that miss the DTLB (Data TLB) and hit the STLB (2nd Level TLB).
159	66	-	MEM_INST_RETIRED. ALL_STORES_PS	All retired store instructions.
X	67	-	MEM_INST_RETIRED. ANY	Counts all retired memory instructions - loads and stores.
160	68	-	MEM_INST_RETIRED. LOCK_LOADS_PS	Retired load instructions with locked access. (Precise Event)
161	69	78	MEM_INST_RETIRED. SPLIT_LOADS_PS	Retired load instructions that split across a cacheline boundary. (Precise Event)
162	70	79	MEM_INST_RETIRED. SPLIT_STORES_PS	Retired store instructions that split across a cacheline boundary. (Precise Event)

Table A.20: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
163	71	-	MEM_INST_RETIRED. STLB_MISS_LOADS_PS	Number of retired load instructions that (start a) miss in the 2nd-level TLB (STLB).
164	72	-	MEM_INST_RETIRED. STLB_MISS_STORES_PS	Number of retired store instructions that (start a) miss in the 2nd-level TLB (STLB).
166	74	-	MEM_LOAD_L3_HIT_RETIRED. XSNP_HIT_PS	Retired load instructions which data sources were L3 and cross-core snoop hits in on-pkg core cache.
165	73	-	MEM_LOAD_L3_HIT_RETIRED. XSNP_HITM_PS	Retired load instructions which data sources were HitM responses from shared L3.
167	X	-	MEM_LOAD_L3_HIT_RETIRED. XSNP_MISS_PS	Retired load instructions which data sources were L3 hit and cross-core snoop missed in on-pkg core cache.
168	X	-	MEM_LOAD_L3_HIT_RETIRED. XSNP_NONE_PS	Retired load instructions which data sources were hits in L3 without snoops required.
X	75	-	MEM_LOAD_L3_MISS_RETIRED. LOCAL_DRAM_PS	Retired load instructions which data sources missed L3 but serviced from local dram.
169	X	-	MEM_LOAD_MISC_RETIRED. UC_PS	Retired instructions with at least 1 uncacheable load or lock.
170	76	-	MEM_LOAD_RETIRED. FB_HIT_PS	Counts retired load instructions with at least one uop was load missed in L1 but hit FB (Fill Buffers) due to preceding miss to the same cache line with data not ready.

Table A.21: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
171	77	80	MEM_LOAD_RETIRED. L1_HIT_PS	Counts retired load instructions with at least one uop that hit in the L1 data cache.
172	78	-	MEM_LOAD_RETIRED. L1_MISS_PS	Counts retired load instructions with at least one uop that missed in the L1 cache.
173	79	81	MEM_LOAD_RETIRED. L2_HIT_PS	Retired load instructions with L2 cache hits as data sources.
174	X	-	MEM_LOAD_RETIRED. L2_MISS_PS	Retired load instructions missed L2 cache as data sources.
175	80	-	MEM_LOAD_RETIRED. L3_HIT_PS	Retired load instructions with L3 cache hits as data sources.
176	81	-	MEM_LOAD_RETIRED. L3_MISS_PS	Retired load instructions missed L3 cache as data sources.
X	83		OCR. DEMAND_DATA_RD. L3_HIT. HIT_OTHER_CORE_FWD	(no details provided on site)
X	82		OCR. DEMAND_DATA_RD. L3_HIT. HITM_OTHER_CORE	Counts demand data reads.
X	84		OCR. DEMAND_RFO. L3_HIT. HITM_OTHER_CORE	Counts all demand data writes (RFOs).
X	85		OCR. PF_L2_RFO. L3_HIT. HITM_OTHER_CORE	Counts all prefetch (that bring data to L2) RFOs.
X	86		OFFCORE_REQUESTS_BUFFER. SQ_FULL	Counts the number of cases when the offcore requests buffer cannot take more entries for the core.
X	87		OFFCORE_REQUESTS OUTSTANDING. ALL_DATA_RD:cmask=4	Counts the number of offcore outstanding cacheable Core Data Read transactions in the super queue every cycle.

Table A.22: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
X	88		OFFCORE_REQUESTS OUTSTANDING. CYCLES_WITH_DATA_RD	Counts cycles when offcore outstanding cacheable Core Data Read transactions are present in the super queue.
180	X		OFFCORE_REQUESTS OUTSTANDING. CYCLES_WITH_DEMAND_DATA_RD	Counts cycles when offcore outstanding Demand Data Read transactions are present in the super queue (SQ).
181	89		OFFCORE_REQUESTS OUTSTANDING. CYCLES_WITH_DEMAND_RFO	Counts the number of offcore outstanding demand rfo Reads transactions in the super queue every cycle.
177	X		OFFCORE_REQUESTS. ALL_DATA_RD	Counts the demand and prefetch data reads.
178	X		OFFCORE_REQUESTS. ALL_REQUESTS	Counts memory transactions reached the super queue including requests initiated by the core, all L3 prefetches, page walks, etc..
179	X	82	OFFCORE_REQUESTS. DEMAND_DATA_RD	Counts the Demand Data Read requests sent to uncore.
214	X	90	OPS_ISSUED. SLOW_LEA	Number of slow LEA uops being allocated.
182	X	-	OTHER_ASSISTS. ANY	Number of times a microcode assist is invoked by HW other than FP-assist.
183	90	-	PARTIAL_RAT_STALLS. SCOREBOARD	This event counts cycles during which the microcode scoreboard stalls happen.
184	X	-	RESOURCE_STALLS. ANY	Counts resource-related stall cycles.

Table A.23: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
185	X	83	RESOURCE_STALLS. SB	Counts allocation stall cycles caused by the store buffer (SB) being full.
186	X	-	ROB_MISC_EVENTS. PAUSE_INST	Number of retired PAUSE instructions (that do not end up with a VMExit to the VMM
187	X	-	RS_EVENTS. EMPTY_CYCLES	Counts cycles during which the reservation station (RS) is empty for the thread.
188	X	-	RS_EVENTS. EMPTY_END	Counts end of periods where the Reservation Station (RS) was empty.
189	X	-	SQ_MISC. SPLIT_LOCK	Counts the number of cache line split locks sent to the uncore.
190	X	-	TLB_FLUSH. DTLB_THREAD	Counts the number of DTLB flush attempts of the thread-specific entries.
191	X	-	TLB_FLUSH. STLB_ANY	Counts the number of any STLB flush attempts (such as entire, VPID, PCID, InvPage, CR3 write, etc.).
192	91	84	UOPS_DISPATCHED_PORT. PORT_0	Counts, on the per-thread basis, cycles during which at least one uop is dispatched from the Reservation Station (RS) to port 0.
193	92	-	UOPS_DISPATCHED_PORT. PORT_1	Counts, on the per-thread basis, cycles during which at least one uop is dispatched from the Reservation Station (RS) to port 1.

Table A.24: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
194	93	-	UOPS_DISPATCHED_PORT. PORT_2	Counts, on the per-thread basis, cycles during which at least one uop is dispatched from the Reservation Station (RS) to port 2.
195	94	-	UOPS_DISPATCHED_PORT. PORT_3	Counts, on the per-thread basis, cycles during which at least one uop is dispatched from the Reservation Station (RS) to port 3.
196	95	-	UOPS_DISPATCHED_PORT. PORT_4	Counts, on the per-thread basis, cycles during which at least one uop is dispatched from the Reservation Station (RS) to port 4.
197	96	-	UOPS_DISPATCHED_PORT. PORT_5	Counts, on the per-thread basis, cycles during which at least one uop is dispatched from the Reservation Station (RS) to port 5.
198	97	-	UOPS_DISPATCHED_PORT. PORT_6	Counts, on the per-thread basis, cycles during which at least one uop is dispatched from the Reservation Station (RS) to port 6.
199	98	85	UOPS_DISPATCHED_PORT. PORT_7	Counts, on the per-thread basis, cycles during which at least one uop is dispatched from the Reservation Station (RS) to port 7.
200	X	-	UOPS_EXECUTED. CORE	Number of uops executed from any thread.
201	99	-	UOPS_EXECUTED. CORE_CYCLES_GE_1	Cycles at least 1 micro-op is executed from any thread on physical core.

Table A.25: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
202	100	-	UOPS_EXECUTED. CORE_CYCLES_GE_2	Cycles at least 2 micro-op is executed from any thread on physical core.
203	101	-	UOPS_EXECUTED. CORE_CYCLES_GE_3	Cycles at least 3 micro-op is executed from any thread on physical core.
204	X	-	UOPS_EXECUTED. CORE_CYCLES_GE_4	Cycles at least 4 micro-op is executed from any thread on physical core.
205	102	-	UOPS_EXECUTED. CORE_CYCLES_NONE	Cycles with no micro-ops executed from any thread on physical core.
206	X	86	UOPS_EXECUTED. CYCLES_GE_1_UOP_EXEC	Cycles where at least 1 uop was executed per-thread.
207	X	87	UOPS_EXECUTED. CYCLES_GE_2_UOPS_EXEC	Cycles where at least 2 uops were executed per-thread.
208	X	-	UOPS_EXECUTED. CYCLES_GE_3_UOPS_EXEC	Cycles where at least 3 uops were executed per-thread.
209	X	-	UOPS_EXECUTED. CYCLES_GE_4_UOPS_EXEC	Cycles where at least 4 uops were executed per-thread.
210	X	-	UOPS_EXECUTED. STALL_CYCLES	Counts cycles during which no uops were dispatched from the Reservation Station (RS) per thread.
211	103	-	UOPS_EXECUTED. THREAD	Cycles with no micro-ops executed from any thread on physical core.
212	104	88	UOPS_EXECUTED. X87	Counts the number of x87 uops executed.
213	105	89	UOPS_ISSUED. ANY	Counts the number of uops that the Resource Allocation Table (RAT) issues to the Reservation Station (RS).

Table A.26: Overview of HPCs used in this study.

Study 2 (31-class)	Study 1 (23-class)	Study 3 (35-class)	HPC Name	HPC Description
215	X	-	UOPS_ISSUED. STALL_CYCLES	Counts cycles during which the Resource Allocation Table (RAT) does not issue any Uops to the reservation station (RS) for the current thread.
216	106	-	UOPS_ISSUED. VECTOR_WIDTH_MISMATCH	Counts the number of Blend Uops issued by the Resource Allocation Table (RAT) to the reservation station (RS) in order to preserve upper bits of vector registers.
217	107	-	UOPS_RETIRED. MACRO_FUSED	Counts the number of macro-fused uops retired. (non precise)
218	108	-	UOPS_RETIRED. RETIRE_SLOTS	Counts the retirement slots used.
219	X	-	UOPS_RETIRED. STALL_CYCLES	This event counts cycles without actually retired uops.
220	X	-	UOPS_RETIRED. TOTAL_CYCLES	Number of cycles using always true condition applied to non PEBS uops retired event.

Bibliography

[1] Intel perfmon events. https://perfmon-events.intel.com/. Accessed: 2022-01-16.

[2] Electromagnetic energy facts, and a qualitative view. https://www.ices-emfsafety.org/electromagnetic-energy/. Accessed: 2022-03-03.

[3] The engineering toolbox: Electromagnetic spectrum. https://www.engineeringtoolbox.com/electromagnetic-spectrum-d_1929.html. Accessed: 2022-03-03.

[4] Mathworks choose classifier options. https://www.mathworks.com/help/stats/choose-a-classifier.html. Accessed: 2023-02-01.

[5] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

[6] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[7] Tempest: A signal problem. https://www.nsa.gov/portals/75/documents/news-features/declassified-documents/cryptologic-spectrum/tempest.pdf. Accessed: 2022-03-02.

[8] Dakshi Agrawal, Bruce Archambeault, Josyula R Rao, and Pankaj Rohatgi. The em side—channel (s). In *International workshop on cryptographic hardware and embedded systems*, pages 29–45. Springer, 2002.

[9] François-Xavier Standaert. Introduction to side-channel attacks. In *Secure integrated circuits and systems*, pages 27–42. Springer, 2010.

[10] YongBin Zhou and DengGuo Feng. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *Cryptology ePrint Archive*, 2005.

[11] Martin Hutle and Markus Kammerstetter. Resilience against physical attacks. In *Smart Grid Security*, pages 79–112. Elsevier, 2015.

[12] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 443–461. Springer, 2009.

[13] Hailong Zhang and Yongbin Zhou. On the exact relationship between the mutual information metric and the success rate metric. *Information Sciences*, 435:15–25, 2018.

[14] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Mutual information analysis: how, when and why? In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 429–443. Springer, 2009.

[15] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *Journal of Cryptology*, 24(2):269–291, 2011.

[16] Amir Moradi, Nima Mousavi, Christof Paar, and Mahmoud Salmasizadeh. A comparative study of mutual information analysis under a gaussian assumption. In *International Workshop on Information Security Applications*, pages 193–205. Springer, 2009.

[17] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 426–442. Springer, 2008.

[18] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1–19. IEEE, 2019.

[19] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *arXiv preprint arXiv:1801.01207*, 2018.

[20] John R Hoehn. Defense primer: Electronic warfare. Library Of Congress, Congressional Research Service, 2020.

[21] MT Thurbon. The origins of electronic warfare. *The RUSI Journal*, 122(3):56–63, 1977.

[22] Alfred Price. *Instruments of darkness: the history of electronic warfare, 1939–1945*. Pen and Sword, 2017.

[23] Joint publication 3-85: Joint electromagnetic spectrum operations. `https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp3_85.pdf`, 2020. Accessed: 2022-03-02.

[24] Zachary Johnson, Alex Varon, Justin Blanco, Ryan Rakvic, James Shey, Hau Ngo, Dane Brown, and Owens Walker. Classifying solid state drive firmware via side-channel current draw analysis. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 943–948. IEEE, 2018.

[25] Dane Brown, Owens Walker, Ryan Rakvic, Robert W Ives, Hau Ngo, James Shey, and Justin Blanco. Towards detection of modified firmware on solid state drives via side channel analysis. In *Proceedings of the International Symposium on Memory Systems*, pages 315–320, 2018.

[26] T Owens Walker, Justin A Blanco, Ryan Rakvic, Ann Vanleer, Dane Brown, James Shey, Gregory L Sinsley, Hau T Ngo, and Robert W Ives. Classifying proprietary firmware on a solid state drive using idle state current draw measurements. *IEEE Access*, 8:31869–31881, 2020.

[27] Gas hits highest price in 6 years, fuel outages persist despite colonial pipeline restart. https://abcnews.go.com/US/gas-hits-highest-price-years-fuel-outages-persist/story?id=77735010. Accessed: 2023-02-21.

[28] Cyber attack suspected in german woman's death. https://www.nytimes.com/2020/09/18/world/europe/cyber-attack-germany-ransomeware-death.html. Accessed: 2023-02-21.

[29] U.s. department of the treasure press release: Remarks by deputy secretary of the treasury wally adeyemo at the international counter ransomware initiative summit. https://home.treasury.gov/news/press-releases/jy1067. Accessed: 2023-02-21.

[30] Fincen analysis reveals ransomware reporting in bsa filings increased significantly during the second half of 2021. https://www.fincen.gov/news/news-releases/fincen-analysis-reveals-ransomware-reporting-bsa-filings-increased-si
Accessed: 2023-02-21.

[31] The white house fact sheet: The second international counter ransomware initiative summit. https://www.whitehouse.gov/briefing-room/statements-releases/2022/11/01/fact-sheet-the-second-international-counter-ransomware-initiative-sum
Accessed: 2023-02-21.

[32] Global ransomware damages to exceed $30bn by 2023. https://www.infosecurity-magazine.com/news/ransomware-exceed-30bn-dollars-2023/. Accessed: 2023-02-21.

[33] Mark Randolph and William Diehl. Power side-channel attack analysis: A review of 20 years of study for the layman. *Cryptography*, 4(2).

[34] Mottaqiallah Taouil, Abdullah Aljuffri, and Said Hamdioui. Power side channel attacks: Where are we standing? In *2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–6. IEEE, 2021.

[35] Asanka Sayakkara, Nhien-An Le-Khac, and Mark Scanlon. A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics. *Digital Investigation*, 29:43–54, 2019.

[36] Jiaji He, Xiaolong Guo, Mark Tehranipoor, Apostol Vassilev, and Yier Jin. Em side channels in hardware security: Attacks and defenses. *IEEE Design & Test*, 2021.

[37] Luong N Nguyen, Chia-Lin Cheng, Milos Prvulovic, and Alenka Zajić. Creating a backscattering side channel to enable detection of dormant hardware trojans. *IEEE transactions on very large scale integration (VLSI) systems*, 27(7):1561–1574, 2019.

[38] Luong N Nguyen, Chia-Lin Cheng, Frank T Werner, Milos Prvulovic, and Alenka Zajic. A comparison of backscattering, em, and power side-channels and their performance in detecting software and hardware intrusions. *Journal of Hardware and Systems Security*, 4(2):150–165, 2020.

[39] GM Deepa, G SriTeja, and S Venkateswarlu. An overview of acoustic side-channel attack. *International Journal of Computer Science & Communication Networks*, 3(1):15, 2013.

[40] Li Zhuang, Feng Zhou, and J Doug Tygar. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):1–26, 2009.

[41] Tzipora Halevi and Nitesh Saxena. Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios. *International Journal of Information Security*, 14(5):443–456, 2015.

[42] S Abhishek Anand and Nitesh Saxena. A sound for a sound: Mitigating acoustic side channel attacks on password keystrokes with active sounds. In *International Conference on Financial Cryptography and Data Security*, pages 346–364. Springer, 2016.

[43] Ilia Shumailov, Laurent Simon, Jeff Yan, and Ross Anderson. Hearing your touch: A new acoustic side channel on smartphones. *arXiv preprint arXiv:1903.11137*, 2019.

[44] Michael Backes, Markus Dürmuth, Sebastian Gerling, Manfred Pinkal, Caroline Sporleder, et al. Acoustic {Side-Channel} attacks on printers. In *19th USENIX Security Symposium (USENIX Security 10)*, 2010.

[45] Mohammad Abdullah Al Faruque, Sujit Rokka Chhetri, Arquimedes Canedo, and Jiang Wan. Acoustic side-channel attacks on additive manufacturing systems. In *2016 ACM/IEEE 7th international conference on Cyber-Physical Systems (ICCPS)*, pages 1–10. IEEE, 2016.

[46] Chen Song, Feng Lin, Zhongjie Ba, Kui Ren, Chi Zhou, and Wenyao Xu. My smartphone knows what you print: Exploring smartphone-based side-channel attacks against 3d printers. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 895–907, 2016.

[47] Adi Shamir and Eran Tromer. Acoustic cryptanalysis, 2004.

[48] Daniel Genkin, Adi Shamir, and Eran Tromer. Rsa key extraction via low-bandwidth acoustic cryptanalysis. In *Annual cryptology conference*, pages 444–461. Springer, 2014.

[49] Daniel Genkin, Adi Shamir, and Eran Tromer. Acoustic cryptanalysis. *Journal of Cryptology*, 30(2):392–443, 2017.

[50] Peng Cheng, Ibrahim Ethem Bagci, Utz Roedig, and Jeff Yan. Sonarsnoop: Active acoustic side-channel attacks. *International Journal of Information Security*, 19(2):213–228, 2020.

[51] Julie Ferrigno and M Hlaváč. When aes blinks: introducing optical side channel. *IET Information Security*, 2(3):94–98, 2008.

[52] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. Simple photonic emission analysis of aes. *Journal of cryptographic engineering*, 3(1):3–15, 2013.

[53] Juliane Krämer, Dmitry Nedospasov, Alexander Schlösser, and Jean-Pierre Seifert. Differential photonic emission analysis. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 1–16. Springer, 2013.

[54] Elad Carmon, Jean-Pierre Seifert, and Avishai Wool. Photonic side channel attacks against rsa. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 74–78. IEEE, 2017.

[55] Sergei Skorobogatov. Using optical emission analysis for estimating contribution to power analysis. In *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 111–119. IEEE, 2009.

[56] Mordechai Guri. Optical covert channel from air-gapped networks via remote orchestration of router/switch leds. In *2018 European Intelligence and Security Informatics Conference (EISIC)*, pages 54–60. IEEE, 2018.

[57] Isabel Wagner and David Eckhoff. Technical privacy metrics: a systematic survey. *ACM Computing Surveys (CSUR)*, 51(3):1–38, 2018.

[58] Jean-Sébasticn Coron, Paul Kocher, and David Naccache. Statistics and secret leakage. In *International Conference on Financial Cryptography*, pages 157–173. Springer, 2000.

[59] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[60] Timothy DelSole. Predictability and information theory. part i: Measures of predictability. *Journal of Atmospheric Sciences*, 61(20):2425–2440, 2004.

[61] Boris Köpf and David Basin. An information-theoretic model for adaptive side-channel attacks. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 286–296, 2007.

[62] Peng Gu, Dylan Stow, Russell Barnes, Eren Kursun, and Yuan Xie. Thermal-aware 3d design for side-channel information leakage. In *2016 IEEE 34th International Conference on Computer Design (ICCD)*, pages 520–527. IEEE, 2016.

[63] James L Massey. Guessing and entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory*, page 204. IEEE, 1994.

[64] Emmanuel Prouff and Matthieu Rivain. Theoretical and practical aspects of mutual information based side channel analysis. In *International Conference on Applied Cryptography and Network Security*, pages 499–518. Springer, 2009.

[65] Konstantinos Chatzikokolakis, Tom Chothia, and Apratim Guha. Statistical measurement of information leakage. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 390–404. Springer, 2010.

[66] Tom Chothia, K Chatzikokolakiz, and A Guha. Statistical measurement of information leakage. *Technische Universiteit Eindhoven, Netherlands and University of Birmingham, UK*.

[67] Dan Walters, Andrew Hagen, and Eric Kedaigle. Sleak: A side-channel leakage evaluator and analysis kit. Technical report, MITRE CORP BEDFORD MA BEDFORD United States, 2014.

[68] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu. A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 73–96, 2020.

[69] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011.

[70] T Schnneider and A Moradi. Leakage assessment methodology—a clear roadmap for side-channel evaluations. *Proc. Cryptographic Hardware Embedded Syst*, pages 495–513, 2015.

[71] V.S. Bokharaie and A. Jahanian. Side-channel leakage assessment metrics and methodologies at design cycle: A case study for a cryptosystem. *Journal of Information Security and Applications*, 2020.

[72] François-Xavier Standaert. How (not) to use welch's t-test in side-channel security evaluations. In *International Conference on Smart Card Research and Advanced Applications*, pages 65–79. Springer, 2018.

[73] John Demme, Robert Martin, Adam Waksman, and Simha Sethumadhavan. Side-channel vulnerability factor: A metric for measuring information leakage. In *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, pages 106–117. IEEE, 2012.

[74] John Demme, Robert Martin, Adam Waksman, and Simha Sethumadhavan. A quantitative, experimental approach to measuring processor side-channel security. *IEEE Micro*, 33(3):68–77, 2013.

[75] John Demme and Simha Sethumadhavan. Side-channel vulnerability metrics: Svf vs. csv. In *Proc. of 11th Annual Workshop on Duplicating, Deconstructing and Debunking (WDDD)*, 2014.

[76] Tianwei Zhang, Fangfei Liu, Si Chen, and Ruby B Lee. Side channel vulnerability metrics: the promise and the pitfalls. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, pages 1–8, 2013.

[77] Robert Callan, Alenka Zajic, and Milos Prvulovic. A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 242–254. IEEE, 2014.

[78] Somdip Dey, Amit Kumar Singh, Xiaohang Wang, and Klaus Dieter McDonald-Maier. Date: Defense against temperature side-channel attacks in dvfs enabled mpsocs. *arXiv preprint arXiv:2007.01377*, 2020.

[79] Ibrahim Issa, Sudeep Kamath, and Aaron B Wagner. Maximal leakage minimization for the shannon cipher system. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 520–524. IEEE, 2016.

[80] Ibrahim Issa, Sudeep Kamath, and Aaron B Wagner. An operational measure of information leakage. In *2016 Annual Conference on Information Science and Systems (CISS)*, pages 234–239. IEEE, 2016.

[81] Ibrahim Issa and Aaron B Wagner. Operational definitions for some common information leakage metrics. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 769–773. IEEE, 2017.

[82] Ibrahim Issa, Aaron B Wagner, and Sudeep Kamath. An operational approach to information leakage. *IEEE Transactions on Information Theory*, 66(3):1625–1657, 2019.

[83] Benjamin Wu, Aaron B Wagner, and G Edward Suh. A case for maximal leakage as a side channel leakage metric. *arXiv preprint arXiv:2004.08035*, 2020.

[84] Jiachun Liao, Oliver Kosut, Lalitha Sankar, and Flavio P Calmon. A tunable measure for information leakage. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 701–705. IEEE, 2018.

[85] Momoka Kasuya, Takanori Machida, and Kazuo Sakiyama. New metric for side-channel information leakage: Case study on em radiation from aes hardware. In *2016 URSI Asia-Pacific Radio Science Conference (URSI AP-RASC)*, pages 1288–1291. IEEE, 2016.

[86] Jonathan Cruz, Prabhat Mishra, and Swarup Bhunia. The metric matters: The art of measuring trust in eiectronics. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–4. IEEE, 2019.

[87] Alric Althoff, Jeremy Blackstone, and Ryan Kastner. Holistic power side-channel leakage assessment: Towards a robust multidimensional metric. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2019.

[88] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):1–29, 2019.

[89] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 348–375, 2020.

[90] Kevin Skadron, Tarek Abdelzaher, and Mircea R Stan. Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management. In *Proceedings Eighth International Symposium on High Performance Computer Architecture*, pages 17–28. IEEE, 2002.

[91] Ryan Cochran, Abdullah Nazma Nowroz, and Sherief Reda. Post-silicon power characterization using thermal infrared emissions. In *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, pages 331–336. IEEE, 2010.

[92] Michael Hutter and Jörn-Marc Schmidt. The temperature side channel and heating fault attacks. In *International Conference on Smart Card Research and Advanced Applications*, pages 219–235. Springer, 2013.

[93] Johann Knechtel and Ozgur Sinanoglu. On mitigation of side-channel attacks in 3d ics: Decorrelating thermal patterns from power and activity. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2017.

[94] Abdullah Aljuffri, Marc Zwalua, Cezar Rodolfo Wedig Reinbrecht, Said Hamdioui, and Mottaqiallah Taouil. Applying thermal side-channel attacks on asymmetric cryptography. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(11):1930–1942, 2021.

[95] Open hardware monitor. `https://openhardwaremonitor.org/`. Last accessed 14-February-2022.

[96] Wei Huang, Shougata Ghosh, Sivakumar Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R Stan. Hotspot: A compact thermal modeling methodology for

early-stage vlsi design. *IEEE Transactions on very large scale integration (VLSI) systems*, 14(5):501–513, 2006.

[97] Ramya Jayaram Masti, Devendra Rai, Aanjhan Ranganathan, Christian Müller, Lothar Thiele, and Srdjan Capkun. Thermal covert channels on multi-core platforms. In *24th USENIX security symposium (USENIX security 15)*, pages 865–880, 2015.

[98] Zijun Long, Xiaohang Wang, Yingtao Jiang, Guofeng Cui, Li Zhang, and Terrence Mak. Improving the efficiency of thermal covert channels in multi-/many-core systems. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1459–1464. IEEE, 2018.

[99] Joonho Kong, Johnsy K John, Eui-Young Chung, Sung Woo Chung, and Jie Hu. On the thermal attack in instruction caches. *IEEE Transactions on Dependable and Secure Computing*, 7(2):217–223, 2010.

[100] Sherief Reda, Kapil Dev, and Adel Belouchrani. Blind identification of thermal models and power sources from thermal measurements. *IEEE Sensors Journal*, 18(2):680–691, 2017.

[101] David Werner, Kyle Juretus, Ioannis Savidis, and Mark Hempstead. Machine learning on the thermal side-channel: Analysis of accelerator-rich architectures. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*, pages 83–91. IEEE, 2018.

[102] Chongxi Bao and Ankur Srivastava. A secure algorithm for task scheduling against side-channel attacks. In *Proceedings of the 4th International Workshop on Trustworthy Embedded Devices*, pages 3–12, 2014.

[103] Domenic Forte, Chongxi Bao, and Ankur Srivastava. Temperature tracking: An innovative run-time approach for hardware trojan detection. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 532–539. IEEE, 2013.

[104] Abdullah Nazma Nowroz, Kangqiao Hu, Farinaz Koushanfar, and Sherief Reda. Novel techniques for high-sensitivity hardware trojan detection using thermal and power maps. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(12):1792–1805, 2014.

[105] Mordechai Guri, Matan Monitz, Yisroel Mirski, and Yuval Elovici. Bitwhisper: Covert signaling channel between air-gapped computers using thermal manipulations. In *2015 IEEE 28th Computer Security Foundations Symposium*, pages 276–289. IEEE, 2015.

[106] Davide B Bartolini, Philipp Miedl, and Lothar Thiele. On the capacity of thermal covert channels in multicores. In *Proceedings of the Eleventh European Conference on Computer Systems*, pages 1–16, 2016.

[107] David Paul-Pena, Prashanth Krishnamurthy, Ramesh Karri, and Farshad Khorrami. Process-aware side channel monitoring for embedded control system security. In *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6. IEEE, 2017.

[108] Somdip Dey, Amit Kumar Singh, and Klaus McDonald-Maier. Thermalattacknet: Are cnns making it easy to perform temperature side-channel attack in mobile edge devices? *Future Internet*, 13(6):146, 2021.

[109] Eduardo Boemo and Sergio López-Buedo. Thermal monitoring on fpgas using ring-oscillators. In *International Workshop on Field Programmable Logic and Applications*, pages 69–78. Springer, 1997.

[110] Sergio Lopez-Buedo, Javier Garrido, and Eduardo I Boemo. Dynamically inserting, operating, and eliminating thermal sensors of fpga-based systems. *IEEE Transactions on components and packaging technologies*, 25(4):561–566, 2002.

[111] John J León Franco, Eduardo Boemo, Encarnación Castillo, and Luis Parrilla. Ring oscillators as thermal sensors in fpgas: Experiments in low voltage. In *2010 VI Southern Programmable Logic Conference (SPL)*, pages 133–137. IEEE, 2010.

[112] Ji Sun, Ray Bittner, and Ken Eguro. Fpga side-channel receivers. In *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, pages 267–276, 2011.

[113] Martin Gag, Tim Wegner, Ansgar Waschki, and Dirk Timmermann. Temperature and on-chip crosstalk measurement using ring oscillators in fpga. In *2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 201–204. IEEE, 2012.

[114] Christoph Ruething, Andreas Agne, Markus Happe, and Christian Plessl. Exploration of ring oscillator design space for temperature measurements on fpgas. In *22nd International Conference on Field Programmable Logic and Applications (FPL)*, pages 559–562. IEEE, 2012.

[115] Wenjuan Yu, Shiwei Feng, Yamin Zhang, and Bangbing Shi. Temperature distribution measurement based on field-programmable gate array embedded ring oscillators. *Solid-State Electronics*, 158:16–21, 2019.

[116] Luciana De Micco, Carlos Minchola, John J Leon-Franco, Eduardo Boemo, and Maximiliano Antonelli. An annotated guide to utilize ring-oscillators as thermal sensor in fpga technology. In *2020 Argentine Conference on Electronics (CAE)*, pages 1–7. IEEE, 2020.

[117] Sheng Wang, Shiwei Feng, Yuxuan Xiao, Chaoxu Hu, and Shijie Pan. Build-in compact and efficient temperature sensor array on field programmable gate array. *Microelectronics Journal*, 111:105018, 2021.

[118] Julien Brouchier, Tom Kean, Carol Marsh, and David Naccache. Temperature attacks. *IEEE Security & Privacy*, 7(2):79–82, 2009.

[119] Taras Iakymchuk, Maciej Nikodem, and Krzysztof Kepa. Temperature-based covert channel in fpga systems. In *6th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pages 1–7. IEEE, 2011.

[120] Shanquan Tian and Jakub Szefer. Temporal thermal covert channels in cloud fpgas. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 298–303, 2019.

[121] Daniel-Ciprian Vasile, Paul Svasta, and Mihaela Pantazică. Preventing the temperature side channel attacks on security circuits. In *2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pages 244–247. IEEE, 2019.

[122] Puyan Dadvar and Kevin Skadron. Potential thermal security risks. In *Semiconductor Thermal Measurement and Management IEEE Twenty First Annual IEEE Symposium, 2005.*, pages 229–234. IEEE, 2005.

[123] Jakub Szefer. Survey of microarchitectural side and covert channels, attacks, and defenses. *Journal of Hardware and Systems Security*, 3(3):219–234, 2019.

[124] Xiaoxuan Lou, Tianwei Zhang, Jun Jiang, and Yinqian Zhang. A survey of microarchitectural side-channel vulnerabilities, attacks, and defenses in cryptography. *ACM Computing Surveys (CSUR)*, 54(6):1–37, 2021.

[125] Arnab Kumar Biswas, Dipak Ghosal, and Shishir Nagaraja. A survey of timing channels and countermeasures. *ACM Computing Surveys (CSUR)*, 50(1):1–39, 2017.

[126] Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering*, 8(1):1–27, 2018.

[127] Sandeep Saxena, Goutam Sanyal, et al. Cache based side channel attack: A survey. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 278–284. IEEE, 2018.

[128] Yangdi Lyu and Prabhat Mishra. A survey of side-channel attacks on caches and countermeasures. *Journal of Hardware and Systems Security*, 2(1):33–50, 2018.

[129] Xinliang Miao, Liehui Jiang, and Rui Chang. Survey of access-driven cache-based side channel attack. *Journal of Computer Research and Development*, 57(4):824, 2020.

[130] Sanjeev Das, Jan Werner, Manos Antonakakis, Michalis Polychronakis, and Fabian Monrose. Sok: The challenges, pitfalls, and perils of using hardware performance counters for security. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 20–38. IEEE, 2019.

[131] Philip J Mucci, Shirley Browne, Christine Deane, and George Ho. Papi: A portable interface to hardware performance counters. In *Proceedings of the department of defense HPCMP users group conference*, volume 710, 1999.

[132] Intel®vtune™ profiler. `https://www.intel.com/content/dam/develop/external/us/en/documents/vtune-profiler-user-guide.pdf`. Accessed: 2022-06-13.

[133] Vincent M Weaver and Sally A McKee. Can hardware performance counters be trusted? In *2008 IEEE International Symposium on Workload Characterization*, pages 141–150. IEEE, 2008.

[134] Sai Manoj Pudukotai Dinakarrao, Sairaj Amberkar, Sahil Bhat, Abhijitt Dhavlle, Hossein Sayadi, Avesta Sasan, Houman Homayoun, and Setareh Rafatirad. Adversarial attack on microarchitectural events based malware detectors. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[135] Boyou Zhou, Anmol Gupta, Rasoul Jahanshahi, Manuel Egele, and Ajay Joshi. Hardware performance counters can detect malware: Myth or fact? In *Proceedings of the 2018 on Asia conference on computer and communications security*, pages 457–468, 2018.

[136] Boyou Zhou, Anmol Gupta, Rasoul Jahanshahi, Manuel Egele, and Ajay Joshi. A cautionary tale about detecting malware using hardware performance counters and machine learning. *IEEE Design & Test*, 38(3):39–50, 2021.

[137] Benjamin Serebrin and Daniel Hecht. Virtualizing performance counters. In *European Conference on Parallel Processing*, pages 223–233. Springer, 2011.

[138] Rana Elnaggar, Lorenzo Servadei, Shubham Mathur, Robert Wille, Wolfgang Ecker, and Krishnendu Chakrabarty. Accurate and robust malware detection: Running xgboost on runtime data from performance counters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(7):2066–2079, 2021.

[139] Maria Mushtaq, Pascal Benoit, and Umer Farooq. Challenges of using performance counters in security against side-channel leakage. In *5th International Conference on Cyber-Technologies and Cyber-Systems (CYBER 2020)*, 2020.

[140] Charalambos Konstantinou, Xueyang Wang, Prashanth Krishnamurthy, Farshad Khorrami, Michail Maniatakos, and Ramesh Karri. Hpc-based malware detectors actually work: Transition to practice after a decade of research. *IEEE Design & Test*, 2022.

[141] The cost of cybercrime. `https://www.accenture.com/_acnmedia/pdf-96/accenture-2019-cost-of-cybercrime-study-final.pdf`. Accessed: 2022-06-16.

[142] The 11 most common types of malware. `https://www.crowdstrike.com/cybersecurity-101/malware/types-of-malware/`. Accessed: 2022-06-12.

[143] Bernard Knox et al. *The Odyssey*. Penguin, 1997.

[144] What are ransomware families? `https://businessinsights.bitdefender.com/what-are-ransomware-families-and-why-knowing-them-can-help-your-b` Accessed: 2022-11-01.

[145] Ransomware as a service (raas) explained. `https://www.crowdstrike.com/cybersecurity-101/ransomware/ransomware-as-a-service-raas/`. Accessed: 2022-11-01.

[146] Nwokedi Idika and Aditya P Mathur. A survey of malware detection techniques. *Purdue University*, 48(2):32–46, 2007.

[147] Ekta Gandotra, Divya Bansal, and Sanjeev Sofat. Malware analysis and classification: A survey. *Journal of Information Security*, 2014, 2014.

[148] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 3–14, 2011.

[149] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni. Survey of machine learning techniques for malware analysis. *Computers & Security*, 81:123–147, 2019.

[150] P Vinod, R Jaipur, V Laxmi, and M Gaur. Survey on malware detection methods. In *Proceedings of the 3rd Hackers' Workshop on computer and internet security (IITKHACK'09)*, pages 74–79, 2009.

[151] Imtithal A Saeed, Ali Selamat, and Ali MA Abuagoub. A survey on malware and malware detection systems. *International Journal of Computer Applications*, 67(16), 2013.

[152] Quoc-Dung Ngo, Huy-Trung Nguyen, Van-Hoang Le, and Doan-Hieu Nguyen. A survey of iot malware and detection methods based on static features. *ICT Express*, 6(4):280–286, 2020.

[153] Daniel Gibert, Carles Mateu, and Jordi Planes. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications*, 153:102526, 2020.

[154] Ömer Aslan Aslan and Refik Samet. A comprehensive review on malware detection approaches. *IEEE Access*, 8:6249–6271, 2020.

[155] Jungmin Park, Xiaolin Xu, Yier Jin, Domenic Forte, and Mark Tehranipoor. Power-based side-channel instruction-level disassembler. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018.

[156] Haider Adnan Khan, Monjur Alam, Alenka Zajic, and Milos Prvulovic. Detailed tracking of program control flow using analog side-channel signals: a promise for iot malware detection and a threat for many cryptographic implementations. In *Cyber Sensing 2018*, volume 10630, pages 23–36. SPIE, 2018.

[157] Haider Adnan Khan, Nader Sehatbakhsh, Luong N Nguyen, Milos Prvulovic, and Alenka Zajić. Malware detection in embedded systems using neural network model for electromagnetic side-channel signals. *Journal of Hardware and Systems Security*, 3(4):305–318, 2019.

[158] Nikhil Chawla, Harshit Kumar, and Saibal Mukhopadhyay. Machine learning in wavelet domain for electromagnetic emission based malware analysis. *IEEE Transactions on Information Forensics and Security*, 16:3426–3441, 2021.

[159] Corey Malone, Mohamed Zahran, and Ramesh Karri. Are hardware performance counters a cost effective way for integrity checking of programs. In *Proceedings of the sixth ACM workshop on Scalable trusted computing*, pages 71–76, 2011.

[160] Rian Quinn. *Detection of malware via side channel information*. State University of New York at Binghamton, 2012.

[161] Xueyang Wang and Ramesh Karri. Numchecker: Detecting kernel control-flow modifying rootkits by using hardware performance counters. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–7. IEEE, 2013.

[162] Alberto Garcia-Serrano. Anomaly detection for malware identification using hardware performance counters. *arXiv preprint arXiv:1508.07482*, 2015.

[163] John Demme, Matthew Maycock, Jared Schmitz, Adrian Tang, Adam Waksman, Simha Sethumadhavan, and Salvatore Stolfo. On the feasibility of online malware detection with performance counters. *ACM SIGARCH Computer Architecture News*, 41(3):559–570, 2013.

[164] Adrian Tang, Simha Sethumadhavan, and Salvatore J Stolfo. Unsupervised anomaly-based malware detection using hardware features. In *International Workshop on Recent Advances in Intrusion Detection*, pages 109–129. Springer, 2014.

[165] Mikhail Kazdagli, Vijay Janapa Reddi, and Mohit Tiwari. Quantifying and improving the efficiency of hardware-based mobile malware detectors. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13. IEEE, 2016.

[166] Kanad Basu, Prashanth Krishnamurthy, Farshad Khorrami, and Ramesh Karri. A theoretical study of hardware performance counters-based malware detection. *IEEE Transactions on Information Forensics and Security*, 15:512–525, 2019.

[167] Xueyang Wang, Sek Chai, Michael Isnardi, Sehoon Lim, and Ramesh Karri. Hardware performance counter-based malware identification and detection with adaptive compressive sensing. *ACM Transactions on Architecture and Code Optimization (TACO)*, 13(1):1–23, 2016.

[168] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. A low-level hybrid intrusion detection system based on hardware performance counters. In *Trust, Security and Privacy for Big Data*, pages 174–192. CRC Press, 2022.

[169] Mohammad Bagher Bahador, Mahdi Abadi, and Asghar Tajoddin. Hlmd: a signature-based approach to hardware-level behavioral malware detection and classification. *The Journal of Supercomputing*, 75(8):5551–5582, 2019.

[170] Harshit Kumar, Biswadeep Chakraborty, Sudarshan Sharma, Nikhil Chawla, and Saibal Mukhopadhyay. Xmd: An expansive hardware-telemetry based malware detector to enhance endpoint detection. *arXiv preprint arXiv:2206.12447*, 2022.

[171] Sai Praveen Kadiyala, Akella Kartheek, and Tram Truong-Huu. Program behavior analysis and clustering using performance counters. In *2020 Workshop in DYnamic and Novel Advances in Machine Learning and Intelligent Cyber Security*, pages 1–9, 2020.

[172] Vitor Ramos, Carlos Valderrama, Samuel Xavier de Souza, and Pierre Manneback. An accurate tool for modeling, fingerprinting, comparison, and clustering of parallel applications based on performance counters. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 797–804. IEEE, 2019.

[173] Mohammad Bagher Bahador, Mahdi Abadi, and Asghar Tajoddin. Hpcmalhunter: Behavioral malware detection using hardware performance counters and singular value decomposition. In *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 703–708. IEEE, 2014.

[174] Nisarg Patel, Avesta Sasan, and Houman Homayoun. Analyzing hardware based malware detectors. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2017.

[175] Hossein Sayadi, Amir Houmansadr, Setareh Rafatirad, and Houman Homayoun. Comprehensive assessment of run-time hardware-supported malware detection using general and ensemble learning. In *Proceedings of the 15th ACM International Conference on Computing Frontiers*, pages 212–215, 2018.

[176] Sai Praveen Kadiyala, Pranav Jadhav, Siew-Kei Lam, and Thambipillai Srikanthan. Hardware performance counter-based fine-grained malware detection. *ACM Transactions on Embedded Computing Systems (TECS)*, 19(5):1–17, 2020.

[177] Zhangying He, Amin Rezaei, Houman Homayoun, and Hossein Sayadi. Deep neural network and transfer learning for accurate hardware-based zero-day malware detection. In *Proceedings of the Great Lakes Symposium on VLSI 2022*, pages 27–32, 2022.

[178] Abraham Peedikayil Kuruvila, Xingyu Meng, Shamik Kundu, Gaurav Pandey, and Kanad Basu. Explainable machine learning for intrusion detection via hardware performance counters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[179] Zhixin Pan, Jennifer Sheldon, and Prabhat Mishra. Hardware-assisted malware detection and localization using explainable machine learning. *IEEE Transactions on Computers*, 2022.

[180] Umara Urooj, Bander Ali Saleh Al-rimy, Anazida Zainal, Fuad A Ghaleb, and Murad A Rassam. Ransomware detection using the dynamic analysis and machine learning: A survey and research directions. *Applied Sciences*, 12(1):172, 2021.

[181] Sajad Homayoun, Ali Dehghantanha, Marzieh Ahmadzadeh, Sattar Hashemi, and Raouf Khayami. Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence. *IEEE transactions on emerging topics in computing*, 8(2):341–351, 2017.

[182] Chris Moore. Detecting ransomware with honeypot techniques. 2016.

[183] Saiyed Kashif Shaukat and Vinay J Ribeiro. Ransomwall: A layered defense system against cryptographic ransomware attacks using machine learning. In *2018 10th international conference on communication systems & networks (COMSNETS)*, pages 356–363. IEEE, 2018.

[184] Rakshit Agrawal, Jack W Stokes, Karthik Selvaraj, and Mady Marinescu. Attention in recurrent neural networks for ransomware detection. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3222–3226. IEEE, 2019.

[185] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In *International conference on detection of intrusions and malware, and vulnerability assessment*, pages 3–24. Springer, 2015.

[186] Amin Kharraz and Engin Kirda. Redemption: Real-time protection against ransomware at end-hosts. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 98–119. Springer, 2017.

[187] Amin Kharraz, William Robertson, and Engin Kirda. Protecting against ransomware: A new line of research or restating classic ideas? *IEEE Security & Privacy*, 16(3):103–107, 2018.

[188] Nolen Scaife, Henry Carter, Patrick Traynor, and Kevin RB Butler. Cryptolock (and drop it): stopping ransomware attacks on user data. In *2016 IEEE 36th international conference on distributed computing systems (ICDCS)*, pages 303–312. IEEE, 2016.

[189] Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, and Emil C Lupu. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. *arXiv preprint arXiv:1609.03020*, 2016.

[190] Mehmet Sabir Kiraz, Ziya Alper Genç, and Erdinç Öztürk. Detecting large integer arithmetic for defense against crypto ransomware. *Cryptology ePrint Archive*, 2017.

[191] Routa Moussaileb, Benjamin Bouget, Aurélien Palisse, Hélène Le Bouder, Nora Cuppens, and Jean-Louis Lanet. Ransomware's early mitigation mechanisms. In *Proceedings of the 13th international conference on availability, reliability and security*, pages 1–10, 2018.

[192] Nicoló Andronio, Stefano Zanero, and Federico Maggi. Heldroid: Dissecting and detecting mobile ransomware. In *international symposium on recent advances in intrusion detection*, pages 382–404. Springer, 2015.

[193] Shagufta Mehnaz, Anand Mudgerikar, and Elisa Bertino. Rwguard: A real-time detection system against cryptographic ransomware. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 114–136. Springer, 2018.

[194] Fei Tang, Boyang Ma, Jinku Li, Fengwei Zhang, Jipeng Su, and Jianfeng Ma. Ransomspector: An introspection-based approach to detect crypto ransomware. *Computers & Security*, 97:101997, 2020.

[195] Sun-Jin Lee, Hye-Yeon Shim, Yu-Rim Lee, Tae-Rim Park, So-Hyun Park, and Il-Gu Lee. Study on systematic ransomware detection techniques. In *2022 24th International Conference on Advanced Communication Technology (ICACT)*, pages 297–301. IEEE, 2022.

[196] Manaar Alam, Sarani Bhattacharya, Swastika Dutta, Sayan Sinha, Debdeep Mukhopadhyay, and Anupam Chattopadhyay. Ratafia: ransomware analysis using time and frequency informed autoencoders. In *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 218–227. IEEE, 2019.

[197] Manaar Alam, Sayan Sinha, Sarani Bhattacharya, Swastika Dutta, Debdeep Mukhopadhyay, and Anupam Chattopadhyay. Rapper: Ransomware prevention via performance counters. *arXiv preprint arXiv:2004.01712*, 2020.

[198] Nitin Pundir, Mark Tehranipoor, and Fahim Rahman. Ranstop: A hardware-assisted runtime crypto-ransomware detection technique. *arXiv preprint arXiv:2011.12248*, 2020.

[199] John Podolanko. *Effective Crypto Ransomware Detection Using Hardware Performance Counters*. PhD thesis, 2019.

[200] Michael A Taylor, Kaitlin N Smith, and Mitchell A Thornton. Sensor-based ransomware detection. In *Future Technologies Conference*, pages 794–801, 2017.

[201] Michael A Taylor, Eric C Larson, and Mitchell A Thornton. Rapid ransomware detection through side channel exploitation. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 47–54. IEEE, 2021.

[202] Sana Aurangzeb, Rao Naveed Bin Rais, Muhammad Aleem, Muhammad Arshad Islam, and Muhammad Azhar Iqbal. On the classification of microsoft-windows ransomware using hardware profile. *PeerJ Computer Science*, 7:e361, 2021.

[203] Gaddisa Olani, Chun-Feng Wu, Yuan-Hao Chang, and Wei-Kuan Shih. Deepware: Imaging performance counters with deep learning to detect ransomware. *IEEE Transactions on Computers*, 2022.

[204] Olivia Shapiro, Jennie Hill, Robert Ives, Ryan Rakvic, Hau Ngo, and James Shey. Toward thermal imaging analysis to characterize operations of solid-state drives via the temperature side-channel. In *2022 Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2022.

[205] Nakul Garg, Irtaza Shahid, Erin Avllazagaj, Jennie Hill, Jun Han, and Nirupam Roy. Thermware: Toward side-channel defense for tiny iot devices. In *Proceedings of the 24th International Workshop on Mobile Computing Systems and Applications*, pages 81–88, 2023.

[206] Erin Avllazagaj, Ziyun Zhu, Leyla Bilge, Davide Balzarotti, and Tudor Dumitras. When malware changed its mind: An empirical study of variable program behaviors in the real world. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3487–3504. USENIX Association, August 2021.

[207] Maik Morgenstern and Hendrik Pilz. Useful and useless statistics about viruses and anti-virus programs. In *Proceedings of the CARO Workshop*, 2010.

[208] Image for linux - terabyte unlimited. `https://www.terabyteunlimited.com/image-for-linux/`. Accessed: 2022-04-16.

[209] Mathworks feature selection and feature transformation using classification learner app. `https://www.mathworks.com/help/stats/feature-selection-and-feature-transformation.html`. Accessed: 2022-09-16.

[210] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21:1–13, 2020.

[211] JonPaul Canclini, James McMasters, James Shey, Owens Walker, Ryan Rakvic, Hau Ngo, and Kevin D Fairbanks. Inferring read and write operations of solid-state drives based on energy consumption. In *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 1–6. IEEE, 2016.

[212] James Shey, Ryan Rakvic, Hau Ngo, Owens Walker, Thomas Tedesso, Justin A Blanco, and Kevin Fairbanks. Inferring trimming activity of solid-state drives based on energy consumption. In *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, pages 1–6. IEEE, 2016.

[213] James Shey, Justin A Blanco, Owens Walker, Thomas W Tedesso, Hau T Ngo, Ryan Rakvic, and Kevin D Fairbanks. Monitoring device current to characterize trim operations of solid-state drives. *IEEE Transactions on Information Forensics and Security*, 14(5):1296–1306, 2018.

[214] Peter Welch. The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. *IEEE Transactions on audio and electroacoustics*, 15(2):70–73, 1967.