# Selection of Best Fit Hardware Performance Counters to Detect Cache Side-Channel Attacks

Melis Kapotoglu Koc
Department of Computer Engineering
Istanbul Technical University
Istanbul, Turkey
kapotoglu@itu.edu.tr

Deniz Turgay Altilar
Department of Computer Engineering
Istanbul Technical University
Istanbul, Turkey
altilar@itu.edu.tr

## ABSTRACT

Cache side-channel attack is a common threat in cloud environments where caches are shared across co-located tenants. Detection of such attacks in real-time before the attack procedure is completed can enable cloud users to come up with a countermeasure and protect their privacy against these kinds of vulnerabilities. In this work, a real-time cache side-channel attack detection system for cloud systems is presented which leverages hardware performance counters. The combination of two neural networks is trained with long-term time sequences collected via hardware performance counters to learn the normal behavior of benign applications so that anomalies caused by attackers can be detected. This paper primarily examines the selection of best fit hardware performance counters for this purpose. Initial experiments are performed and time series feature extraction and selection methods are applied to preliminary results for the analysis.

## CCS CONCEPTS

• **Security and privacy** → **Intrusion detection systems**; **Distributed systems security**; *Virtualization and security*.

## KEYWORDS

cache side-channel attacks; real-time attack detection; hardware performance counters

**ACM Reference Format:**
Melis Kapotoglu Koc and Deniz Turgay Altilar. 2023. Selection of Best Fit Hardware Performance Counters to Detect Cache Side-Channel Attacks. In *Proceedings of the 2023 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (SaT-CPS '23), April 26, 2023, Charlotte, NC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3579988.3585052

## 1 INTRODUCTION

Cache is one of the side channels which can be leveraged by measuring and evaluating the time of memory accesses to get some information about the utilization of the shared cache. A cache side-channel attack is one of the most well-known threats that can be performed in cloud systems as well as non-virtualized environments. Since the attacker does not need to access the device physically to conduct this type of attack, the cache can be used as the source of vulnerability when co-residency is achieved in a multi-tenant cloud system. In several prior researches such as [1, 12, 14, 24]; it is proved that cache side-channel attacks are practically applicable across virtual machine (VM) boundaries. Therefore it is crucial to develop powerful detection systems to make cloud environments more secure.

The main challenge of a detection system against cache side-channel attacks is to be capable of detecting all attack types including the ones not yet discovered. Besides the core challenges of cache side-channel attacks, additional constraints specific to cloud environments should also be considered to propose a practical detection system for cloud systems.

In the scope of this work, a real-time anomaly-based detection system is proposed for cross-VM cache side-channel attacks. Any type of malicious activity in cache side-channel attacks leaves abnormal alteration in the exploited shared resources. Therefore anomaly-based detection techniques can be applied to the data collected by monitoring the victim VM's own cache utilization instead of co-located VMs. Hardware performance counters exposed by the Performance Monitor Unit in many modern processors can be used to monitor changes on the victim VM over time. The anomaly-based detection technique will be developed by combining Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM) network because of their capabilities at feature extraction and temporal modeling respectively [20]. A unified neural network will be trained with time-based benign execution patterns collected via hardware events. In summary, the contribution of the proposed system can be listed as follows:

- The first usage of the unified architecture of CNN and LSTM network in the detection of cross-VM cache side-channel attacks
- Detection of all types of cache side-channel attacks in real-time
- Diversity in scenarios including both virtualized and non-virtualized environments

In the proposed detection system, a unified network model is trained after determining best fit hardware performance counters and collecting a sufficient number of benign execution patterns in the format of time series data. This paper primarily focuses on the selection of hardware performance counters that yield the most useful results for the proposed anomaly detection system. The motivation of the paper is to study the characteristics of hardware performance counters with respect to time during the execution of comprehensive scenarios.



*SaT-CPS '23, April 26, 2023, Charlotte, NC, USA*

ACM ISBN 979-8-4007-0100-9/23/04.
https://doi.org/10.1145/3579988.3585052

## 2 BACKGROUND

In this section cache side-channel attacks and hardware performance counters are introduced for a better understanding of the proposed system.

### 2.1 Cache Side-Channel Attacks

The adversary tries to extract sensitive information such as the secret key in a cryptographic application, by peeking at shared CPU caches in the cache side-channel attacks. Several techniques supported by most modern CPUs for optimization purposes such as hierarchical cache, memory deduplication, Intel TSX, etc. are leveraged by the attacker to manipulate the cached data. Data flows or control flows that can reveal secrecy-related information are monitored with the help of cache use patterns. Two common cache manipulation techniques for side-channel attacks are Prime-Probe and Flush-Reload attacks which are explained as follows:

- **Prime-Probe Attack:** First proposal of this attack is seen in [19]. In the beginning, an array of memory blocks corresponding to the concerned cache sets are allocated by the adversary. Then execution of the attack phases starts. In the *PRIME* phase, allocated memory blocks are accessed to be placed in the cache. In this way, memory blocks which belong to the victim are evicted from the cache. The adversary becomes idle for some time interval after *PRIME* phase to let the victim perform sensitive operations using cache. Then *PROBE* phase is executed in which the memory blocks allocated in the beginning are read again to measure the time needed for accessing them. If the accessed memory block is residing in the cache, the required time to read it should be shorter as a result of the cache hit. If the measured time is longer, the cache has been filled by the victim during the idle period of the adversary. These operations are repeated until an amount of meaningful execution traces are collected.
- **Flush-Reload Attack:** This attack is first proposed in [9] and assumes that the victim and adversary share the same cache lines. This actually means that the same memory pages should be shared by different parties. This assumption can be met with the existence of memory deduplication technique. Critical memory blocks from shared pages are determined by the adversary as the initialization of this attack. In the *FLUSH* phase, selected memory blocks are removed from cache memory with the help of clflush[18] instruction. Similar to the previous attack, the adversary waits for some time to have the victim perform sensitive operations using the cache. Then *RELOAD* phase is executed in which selected memory blocks are read again to measure the access time. If the access time is short because of a cache hit, this means the memory block has been used by the victim in the idle time of the adversary.

### 2.2 Hardware Performance Counters

Hardware performance counters (HPCs) are special registers available in most modern processors aiming to monitor certain events primarily for the purpose of software debugging, system performance analysis, and tuning [11]. In addition to the main purposes, they are also being used for intrusion detection with negligible performance overhead in recent researches. It is possible to program HPCs according to the needs with the help of the Performance Monitor Unit (PMU). While measurable events seem to provide simple data, valuable information can be obtained with the right combination of different performance counters assuming that the processor supports multiple counters running simultaneously.

## 3 RELATED WORK

There are various studies on cache side-channel attack detection in the literature. However, in this paper, we consider it appropriate to mention a few studies that provide comparable information about their feature selection methods.

CacheShield [2] is an anomaly-based side-channel attack detection tool for cloud systems, in which a self-monitoring mechanism is applied similar to our system. CacheShield proposes an unsupervised detection algorithm using Cumulative Sum (CUSUM) detection [17] to make the system adaptive for different attack patterns. The use of different machine learning algorithms in the detection of various types of cache side-channel attacks is commonly seen in recent researches [5, 16, 21]. In [4], three methods, two of them based on machine learning, are introduced to detect a spy process conducting a cache side-channel attack. The strengths and weaknesses of each are discussed in real-world scenarios. As a different approach, CloudRadar [23] proposes a real-time cache side-channel attack detection and mitigation system for multi-tenant cloud systems, combining two detection techniques, namely signature-based detection and anomaly-based detection. Signature-based detection is used to recognize the execution pattern of sensitive applications running in the victim VM and anomaly-based detection is used to detect anomalous cache use patterns of co-located VMs as is typically seen during cache side-channel attacks. The authors build their detection system on the premise that consecutive occurrence of cryptographic application execution on the victim VM and abnormal cache usage on co-located VMs is an important indicator of side-channel attacks.

In FortuneTeller [10], authors use the LSTM network and Gated Recurrent Unit (GRU) network individually to model time-dependent behaviors of benign applications to detect microarchitectural attacks in real-world environments. Both network models are trained in an unsupervised way with long sequences of data. In this work, authors feed the LSTM network model with raw time series data. Therefore, the shortcoming of their system is the lack of temporal representation of input data. Their system can be improved by adding a feature extraction layer before the LSTM network. In our research, this issue is resolved by combining CNN and LSTM networks.

In all of the aforementioned studies, HPCs are leveraged to monitor certain hardware events in the detection of cache side-channel attacks. However, they use different methods to determine optimal HPCs for similar purposes. Table 1 shows experiment parameters and selected events in these studies. In our opinion, it would be more effective to select best fit HPCs by applying a feature extraction method which evaluates temporal aspects since studied data consists of a sequence of data points collected over an interval of time.

**Table 1: State-of-the-art Researches on HPC Selection for Cache Side-Channel Attack Detection**

| Name | Algorithm | Sampling Interval | Number of HPCs | Sample Size | Chosen HPCs |
|---|---|---|---|---|---|
| CloudRadar [23] (in signature-based detection) | Fisher Score [7] | 100 $\mu$s | 16 | 15000 | Branch instructions |
| CacheShield [2] | Relief Algorithm [13] | 100 $\mu$s | 30 | 1 million | L3 Cache Misses |
| NIGHTs-WATCH [16] | No specific algorithm (Empirically determined) | N/A | 12 | Unknown number of samples collected through 15000 RSA encryption rounds | Branch Miss-Predictions<br>Total Execution Cycles<br>L1 Instruction Cache Misses<br>L3 Instruction Cache Accesses |
| HybridShield [21] | Greedy Forward Selection [3] Pearson Correlation | 50 $\mu$s | 16 | N/A | L1 Cache Hits |
| Cho et al. [5] | No specific algorithm (Empirically determined) | N/A | N/A | N/A | L1 Cache Misses<br>L2 Cache Misses<br>L3 Cache Misses<br>Speculative and Retired Branches<br>Instruction Per Cycle |
| Chiappetta et al. [4] | No specific algorithm (Empirically determined) | N/A | N/A | N/A | Total Instructions<br>Total CPU Cycles<br>L2 Cache Hits<br>L3 Cache Misses<br>L3 Cache Accesses |
| FortuneTeller [10] | LSTM [8] | 1 ms | 36 | N/A | L1 Instruction Misses<br>L1 Instruction Hits<br>L3 Cache Misses |

## 4 DETECTION SYSTEM

In this section, we propose a real-time cross-VM cache side-channel attack detection system that monitors the anomalies in the values of HPCs. Since cache side-channel attacks cause deviations in the cache usage of the victim's execution, detection can be accomplished by monitoring the HPCs on the victim's own VM, instead of those of co-located VMs. Figure 1 illustrates the architecture diagram of the proposed system. The initial component of the system architecture consists of an offline phase, during which a time series feature extraction and selection method is applied to data gathered from various scenario settings to determine the best fit HPCs. For this purpose, the output of the HPCs is leveraged to describe the execution traces of a benign VM in the presence or absence of an attack in the cloud system. Once the best fit HPCs are determined, a network model that combines a CNN and LSTM network is trained using the supervised data collected in the first step from selected HPCs of several benign applications.

The proposed model utilizes the capability of CNN to extract local features by examining the time series of simultaneous readings from potentially multiple HPCs selected in the previous step. Subsequently, the extracted features are processed through the pooling procedure to diminish dimensionality within the convolution layer. An LSTM layer which is a modified version of Recurrent Neural Networks (RNN) is employed after the convolution layer to enable learning of long-term temporal relationships in sequential data through its internal memory. The dropout layer follows both the convolution and LSTM layers during the training phase of the model to reduce overfitting by randomly deactivating some of the neurons in the network. The fully connected layer, situated at the end of the network just before the output layer, functions to connect and integrate all neurons derived from the preceding layer, thereby generating a binary output that indicates the presence or absence of an attack. Once the proposed hybrid model is trained with the supervised HPC dataset in time series format, it can be employed in the online phase to monitor real-time execution traces of a victim VM to detect any cache side-channel attacks.

Architectures designed for defeating side-channel attacks in the cloud mostly require modifications in the hardware, hypervisors, or guest VMs. However, alterations in the underlying infrastructure are unacceptable for already-built cloud systems. This kind of limitation becomes the main obstacle against most existing detection mechanisms for not being adopted in cloud systems. Therefore the proposed system is designed so as not to require any modification in the underlying hardware, hypervisor, or guest operating systems. It can be run by cloud users individually without any need from the hypervisor except access permission to performance counters. Victim VM can activate the detection system when needed (e.g. in the execution of a sensitive operation) so that performance overhead can be minimized.

There are over one hundred HPCs available, which can vary based on the processor architecture. Additionally, the number of HPCs that can be simultaneously read is also dependent on the processor architecture. Thus, when selecting the best fit HPCs for the proposed system, both their availability and the number that can be read simultaneously should be carefully considered since these constraints have a direct impact on the success of the proposed system.

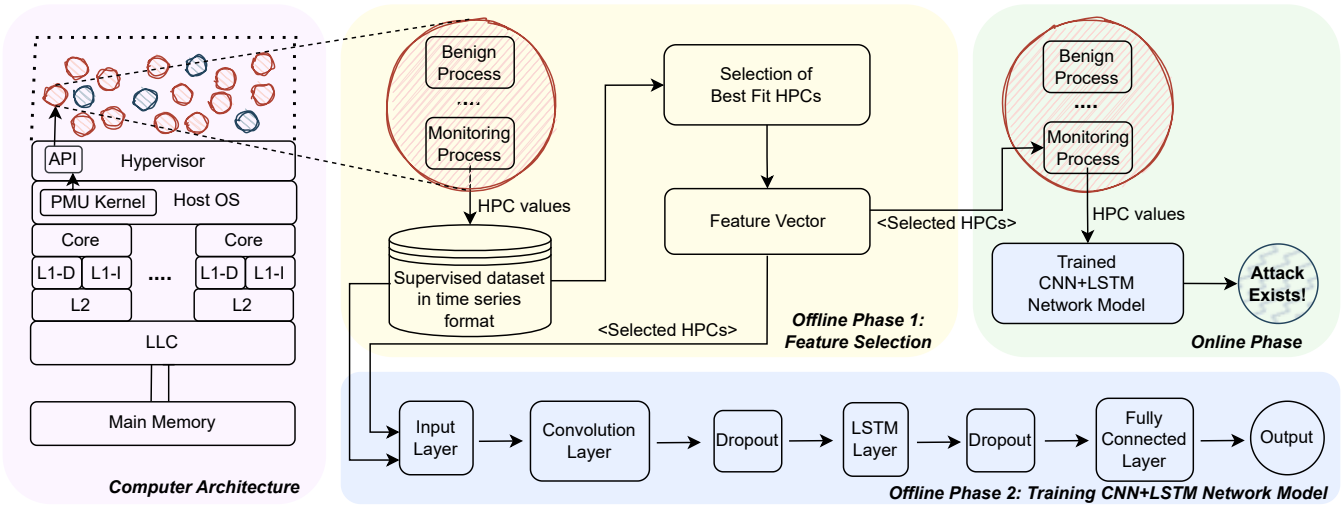


**Figure 1: The architecture diagram of the proposed cache side-channel attack detection system. The red and blue circles on top of the hypervisor indicate benign and attacker VMs, respectively, which are co-located on the same physical cloud server. In the first step of the offline phase, HPC values are collected in various scenarios and the best fit HPCs are selected. The collected data of the selected HPCs are used to train a CNN+LSTM network model. In the online phase, the trained model is used to monitor the executions of victim VMs and detect cache side-channel attacks in real time.**

## 5 EXPERIMENTS

The first step in building the proposed system is the determination of best fit HPCs. Therefore, experiments should be performed to collect data and implement a suitable approach for feature selection. In this section, the experimental setup and scenarios are explained and then preliminary results are analyzed.

### 5.1 Experimental Setup

The experiments are performed on a computer equipped with an Intel(R) Core(TM) i7-11800H CPU having 16 cores, a 24 MB 12-way set associative non-inclusive L3 cache, and 16GB of RAM. The operating system is Ubuntu 20.04 LTS. PAPI (Performance Application Programming Interface) [15] is used as a profiling tool to configure and read HPCs. A data collection framework is developed on top of PAPI to gather data from HPCs. Additional supplementary libraries, namely OpenSSL for including full-featured implementations of common cryptographic algorithms and Mastik [22] for running practical implementation of micro-architectural side-channel attacks are integrated into the data collection framework. For the analysis of initial experiments, tsfresh [6] which is a Python package providing several time series characterization methods, is used to apply feature extraction and selection techniques to the collected data.

### 5.2 Scenarios

In order to select the most suitable HPCs for the proposed detection system, it is required to have a clear understanding of how counter values change while cache attacks are being conducted. Therefore, HPC data should be collected for various real-world scenarios. Initially, two basic categories are determined: *No-attack* and *Attack* scenarios. One benign application and one monitoring process run in *No-attack* scenario. While benign application executes an operation, monitoring process periodically reads counter values which are specifically registered for the benign process. In *Attack* scenario, there should be an extra process running simultaneously along with the others and conducting cache side-channel attack against the benign application.

**Table 2: Top five HPCs having the highest F-Scores and corresponding extraction methods**

| Name | Description | F-Score | Method |
|---|---|---|---|
| UNHALTED_CORE_CYCLES | Number of core clock cycles in unhalted state | 0.937491 | absolute_maximum |
| INSTRUCTION_RETIRED | Number of instructions at retirement | 0.921544 | absolute_maximum |
| UNHALTED_REFERENCE_CYCLES | Number of reference clock cycles (fixed frequency) | 0.897037 | mean |
| BRANCH_INSTRUCTIONS_RETIRED | Number of branch instructions at retirement | 0.852241 | absolute_maximum |
| LLC_REFERENCES | Number of each request to reference a cache line in LLC | 0.805889 | root_mean_square |

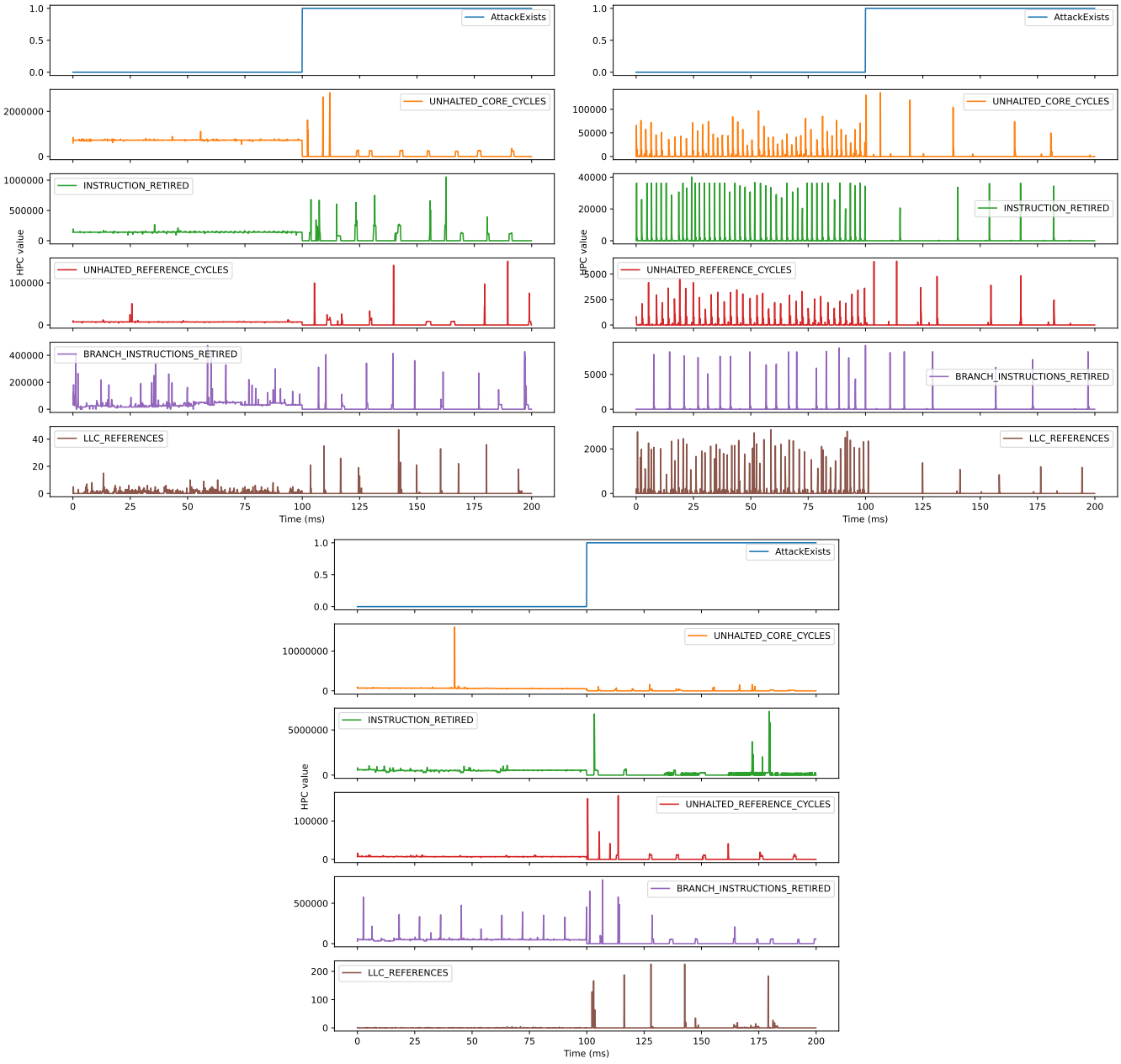


**Figure 2: Change of values of top five HPCs over time for prime number calculation (top left), file operation (top right), and AES-128's t-table execution (bottom). Topmost of each subfigure: 0 for *No-attack* and 1 for *Attack* scenarios. HPCs from top to bottom: UNHALTED_CORE_CYCLES, INSTRUCTION_RETIRED, UNHALTED_REFERENCE_CYCLES, BRANCH_INSTRUCTIONS_RETIRED, and LLC_REFERENCES.**

Three types of benign applications and one cache side-channel attack are selected for the experiments. Benign applications are:

- a computation-intensive application calculating the total number of prime numbers up to a big integer value,
- a file-based operation which reads the content of a text file,
- a cryptographic application that runs AES-128's t-table implementation in OpenSSL.

In *Attack* scenario set, a Flush-Reload attack is performed against each benign application. In this way, the changes in HPC values can be observed in multiple scenarios. For example, in the *Attack* scenario of the prime calculator application, the attacker fails to obtain any sensitive information. On the other hand, it is possible to leak critical information in the *Attack* scenario of file-based operation and cryptographic application.

### 5.3 Preliminary Results

The monitoring, attacker, and benign processes are run simultaneously on the same operating system without any virtualization in the given scenarios. Each benign application is executed individually and the monitoring process is attached to the benign process (not the attacker) since our purpose is to catch the anomalies in the normal execution behavior of benign applications as explained in the system architecture. The total number of HPCs observed during experiments is 37. The sampling interval of the monitoring application is 100 $\mu$s. 1000 samples in time series format are collected for each HPC in every scenario which means the total sample size gathered in the given six scenario variations (*Attack* and *No-attack* scenarios with three benign applications) is more than 200000.

In the analysis of our data, we apply the rolling mechanism in tsfresh package before feature extraction to create multiple consecutive sub-time series from a big single time series by sliding a window over it. In this way, it is possible to evaluate smaller time series data within a specified window size. Then feature extraction is applied by using several operations such as sum, median, absolute maximum, and variance. Eventually, the feature selection module of tsfresh (which operates on p-values) is used to decide the most relevant features based on the importance of extracted features.

The presented results in Table 2 show the top five HPCs that yield the highest F-scores, along with the corresponding extraction methods. Here a basic classifier based on a decision tree is used to evaluate the results. Figure 2 illustrates how the values of the top five HPCs change over time for the given three benign applications under *Attack* and *No-attack* scenarios. The results are shown side by side for *Attack* and *No-attack* scenarios, highlighting the differences in values between the two scenarios. In our experiments, time measurement is converted to index numbers for simplicity since it does not hold any meaning beyond representing the data sequence.

## 6 CONCLUSION AND FUTURE WORK

In this work, an anomaly-based real-time detection system is introduced for cross-VM cache side-channel attacks. Before training the proposed model that combines CNN and LSTM networks, best fit HPCs providing the most useful data for our purpose should be determined. In this paper, preliminary experimental results for determining the best fit HPCs are analyzed. A feature extraction method evaluating temporal aspects is executed on time series data collected from multiple scenario variations. Since the selection of best fit HPCs is so crucial for the success of the proposed detection system, our plan is to enhance scenario variations, especially by including cloud scenarios. A comparison of the experimental results in virtualized and non-virtualized environments is our future work.

Upon completing the proposed detection system, an evaluation of the proposed detection system will be conducted based on significant criteria such as detection latency, the accuracy of attack detection, and performance overhead on the monitored VM. Moreover, a comparison will be performed with a state-of-the-art research that uses a similar approach, such as FortuneTeller, with respect to these criteria. Our objective with the proposed detection system is to demonstrate that it is possible to construct more secure cloud platforms with robust security mechanisms and minimal performance overhead.

## ACKNOWLEDGMENTS

This work was supported by the Scientific Research Projects Department of Istanbul Technical University (Project number: MGA-2021-42887).

## REFERENCES

[1] Gorka Irazoqui Apecechea, Mehmet Sinan Inci, Thomas Eisenbarth, and Berk Sunar. 2014. Fine grain Cross-VM Attacks on Xen and VMware are possible! *Cryptology ePrint Archive* (2014).
[2] Samira Briongos, Gorka Irazoqui, Pedro Malagón, and Thomas Eisenbarth. 2018. Cacheshield: Detecting Cache Attacks through Self-Observation. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. 224–235.
[3] Rich Caruana and Dayne Freitag. 1994. Greedy Attribute Selection. In *Machine Learning Proceedings 1994*. Elsevier, 28–36.
[4] Marco Chiappetta, Erkay Savas, and Cemal Yilmaz. 2016. Real time detection of cache-based side-channel attacks using hardware performance counters. *Applied Soft Computing* 49 (2016), 1162–1174.
[5] Jonghyeon Cho, Taehun Kim, Soojin Kim, Miok Im, Taehyun Kim, and Youngjoo Shin. 2020. Real-Time Detection for Cache Side Channel Attack using Performance Counter Monitor. *Applied Sciences* 10, 3 (2020), 984.
[6] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. 2018. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing* 307 (2018), 72–77.
[7] Richard O Duda, Peter E Hart, and David G Stork. 2000. *Pattern Classification (2nd Edition)*. Wiley-Interscience.
[8] Alex Graves. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks* (2012), 37–45.
[9] David Gullasch, Endre Bangerter, and Stephan Krenn. 2011. Cache Games - Bringing Access-Based Cache Attacks on AES to Practice. In *2011 IEEE Symposium on Security and Privacy*. IEEE, 490–505.
[10] Berk Gulmezoglu, Ahmad Moghimi, Thomas Eisenbarth, and Berk Sunar. 2019. Fortuneteller: Predicting Microarchitectural Attacks via Unsupervised Deep Learning. *arXiv preprint arXiv:1907.03651* (2019).
[11] Clint Huffman. 2014. *Windows Performance Analysis Field Guide*. Elsevier.
[12] Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar. 2015. S$A: A Shared Cache Attack that Works Across Cores and Defies VM Sandboxing–and its Application to AES. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 591–604.
[13] Kenji Kira, Larry A Rendell, et al. 1992. The Feature Selection Problem: Traditional Methods and a New Algorithm. In *Aaai*, Vol. 2. 129–134.
[14] Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B Lee. 2015. Last-Level Cache Side-Channel Attacks are Practical. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 605–622.
[15] Philip J Mucci, Shirley Browne, Christine Deane, and George Ho. 1999. PAPI: A Portable Interface to Hardware Performance Counters. In *Proceedings of the department of defense HPCMP users group conference*, Vol. 710. Citeseer.
[16] Maria Mushtaq, Ayaz Akram, Muhammad Khurram Bhatti, Maham Chaudhry, Vianney Lapotre, and Guy Gogniat. 2018. NIGHTs-WATCH: A Cache-Based Side-Channel Intrusion Detector using Hardware Performance Counters. In *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*. 1–8.
[17] Ewan S Page. 1954. Continuous Inspection Schemes. *Biometrika* 41, 1/2 (1954), 100–115.
[18] Salvador Palanca, Stephen A Fischer, and Subramaniam Maiyuran. 2003. CLFLUSH micro-architectural implementation method and system. US Patent 6,546,462.
[19] Colin Percival. 2005. Cache missing for fun and profit. http://www.daemonology.net/papers/htt.pdf.
[20] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. 2015. Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4580–4584.
[21] Han Wang, Hossein Sayadi, Avesta Sasan, Setareh Rafatirad, and Houman Homayoun. 2020. Hybrid-shield: Accurate and Efficient Cross-Layer Countermeasure for Run-Time Detection and Mitigation of Cache-Based Side-Channel Attacks. In *Proceedings of the 39th International Conference on Computer-Aided Design*. 1–9.
[22] Yuval Yarom. 2016. Mastik: A Micro-Architectural Side-Channel Toolkit. https://cs.adelaide.edu.au/~yval/Mastik/Mastik.pdf.
[23] Tianwei Zhang, Yinqian Zhang, and Ruby B Lee. 2016. Cloudradar: A Real-Time Side-Channel Attack Detection System in Clouds. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 118–140.
[24] Yinqian Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2012. Cross-VM Side Channels and Their Use to Extract Private Keys. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. 305–316.