Cache side-channel attacks detection based on machine learning

Zhongkai Tong
Institute of Information Engineering
Chinese Academy of Sciences
School of Cyber Security
University of Chinese Academy of Sciences
Beijing, China
tongzhongkai@iie.ac.cn

Ziyuan Zhu
Institute of Information Engineering
Chinese Academy of Sciences
School of Cyber Security
University of Chinese Academy of Sciences
Beijing, China
zhuziyuan@iie.ac.cn

Zhanpeng Wang
Institute of Information Engineering
Chinese Academy of Sciences
School of Cyber Security
University of Chinese Academy of Sciences
Beijing, China
wangzhanpeng@iie.ac.cn

Limin Wang
Institute of Information Engineering
Chinese Academy of Sciences
School of Cyber Security
University of Chinese Academy of Sciences
Beijing, China
wanglimin@iie.ac.cn

Yusha Zhang
Institute of Information Engineering
Chinese Academy of Sciences
School of Cyber Security
University of Chinese Academy of Sciences
Beijing, China
zhangyusha@iie.ac.cn

Yuxin Liu
Institute of Information Engineering
Chinese Academy of Sciences
School of Cyber Security
University of Chinese Academy of Sciences
Beijing, China
liuyuxin@iie.ac.cn

***Abstract*—Security has always been one of the main concerns in the field of computer architecture and cloud computing. Cache-based side-channel attacks pose a threat to almost all existing architectures and cloud computing. Especially in the public cloud, the cache is shared among multiple tenants, and cache attacks can make good use of this to extract information. Cache side-channel attacks are a problem to be solved for security, in which how to accurately detect cache side-channel attacks has been a research hotspot. Because the cache side-channel attack does not require the attacker to physically contact the target device and does not need additional devices to obtain the side channel information, the cache-side channel attack is efficient and hidden, which poses a great threat to the security of cryptographic algorithms. Based on the AES algorithm, this paper uses hardware performance counters to obtain the features of different cache events under Flush + Reload, Prime + Probe, and Flush + Flush attacks. Firstly, the random forest algorithm is used to filter the cache features, and then the support vector machine algorithm is used to model the system. Finally, high detection accuracy is achieved under different system loads. The detection accuracy of the system is 99.92% when there is no load, the detection accuracy is 99.85% under the average load, and the detection accuracy under full load is 96.57%.**

Keywords— Cache side-channel attack, Hardware performance counters, Machine learning, Detection

I. Introduction

With the development of science and technology, information technology has become an indispensable part of people's production and life. More and more fields have begun to use information technology to improve work efficiency and create more value. With the popularization of information technology, in addition to higher requirements for the efficiency and quality of information processing, people have gradually begun to attach importance to information security.

Modern information technologies like cloud computing are based on shared hardware resources, which are opaquely accessed by independent users and protected by isolation technology. However, although this isolation is reliable at the logical level and can ensure that users cannot access each other's memory, this isolation technique cannot correctly prevent information leakage from shared hardware resources (such as caches).

The side-channel attack uses the physical changes related to the calculation process unintentionally leaked from the environment to infer secret information, and it has a variety of classification methods [1]. Cache attack is a kind of side-channel attack, which uses the time information leaked at the interface of computer software and hardware: the delay of physical memory access. Compared with the throughput of the processor, the memory access speed is slower and may become a performance bottleneck, so a cache is introduced inside the processor to reduce memory access. As a result, if the data of a memory address is cached by the processor, its access latency will be very low; otherwise, it will be very high. Using a high-precision timer, the program can accurately measure the time-consuming difference of memory access and determine whether the memory access returns from the processor's cache. Besides, due to performance considerations, some processor caches are shared by multiple cores, or even shared globally by the system. Using the cache may bypass some common security isolation mechanisms, such as process virtual memory space, user mode, and kernel[2.3]. Virtual machine isolation [4], etc., to obtain the initially inaccessible sensitive information and decrypt the key of the cryptographic algorithm.

To prevent cache-based side-channel attacks, a variety of detection and mitigation techniques have been proposed, which are mainly based on software and hardware. The hardware-based proposals all try to mitigate cache-based-side channel attacks by redesigning the cache hardware (for example, [5 6 7]). Unfortunately, there is little evidence that CPU manufacturers will deploy these defenses in the foreseeable future because even the so-called low overhead can still cause a lot of damage to performance.

A most important aspect of defending against attacks requires the attention of the research community, and that is detection. Attack detection technology is a more flexible protection measure based on software. Detection-based solutions [8], [9], [10], [11] monitor system functions (such as timing, access behavior, and instruction execution tracking) with the help of hardware performance monitoring to determine whether the system is safe or Has been attacked. Detection technology is the first line of defense that plays a major role in resisting side-channel attacks. When the attack is known, analyzing the characteristics of the attack can provide an effective early detection mechanism. These methods are a solid foundation for building an attack detection mechanism that can detect new attacks or variants of known attacks by monitoring the behavior of the microarchitecture at runtime. Moreover, the detection technology does not need to change the existing system structure and has stronger universality.

In this paper, the problem we are addressing is how to accurately detect whether there is a side-channel attack based on the cache in the system, and we propose a detection mechanism based on machine learning modeling. The model uses a variety of data obtained by the hardware performance counter (HPC) to analyze the different memory access patterns generated by the data-dependent encryption operations performed by the underlying hardware in the presence and absence of attacks. In summary, the main contributions of this article are as follows:

- This paper uses hardware performance counters to obtain 24 different hardware counter features related to the cache side-channel attack under different attacks and different load conditions, and then uses the random forest algorithm to perform feature selection on the feature set, and finally obtains four features as the optimal feature subset (because the existing system only provides about 4 performance counter interfaces). It provides a theoretical basis for establishing the measurement channel detection model.
- As far as we know, compared with other papers that analyze and model different attacks in isolation, this paper is the first one that has established a unified detection model for three different attacks (Flush + Reload, Flush + Flush and Prime + Probe), and explained that under actual system load conditions, namely under no load, average load and full load, it successfully detects cache side-channel attacks with high detection accuracy. By testing under actual load conditions, the deployability of applying the model to real-time detection can be improved.
- For the selected machine learning model, we provide experimental results and discussions on detection accuracy and classification performance. Also, we further analyzed some connections between hardware events and attacks.

The various parts of this paper are organized as follows. In the second part, we analyze the principle of different cache side channel attacks and introduce the hardware performance counter and related work. The third part is to make a specific explanation of the establishment of the machine learning model in this paper. The fourth part is the experimental part of this paper, which introduces the machine learning algorithms related to this paper, and optimizes the features obtained by the hardware performance counter to get the optimal feature subset. The fifth part is about the results of the model under different system loads and the performance of the model. Finally, the sixth section summarizes and discusses this paper.

II. BACKGROUND AND RELATED WORK

A. Cache side attacks

All modern processors work at high clock frequencies. The performance of the CPU depends not only on the clock frequency but also on the speed of the main memory. Main memory access time is slow. The main memory cannot meet the memory bandwidth requirements of high-speed processors. The introduction of high-speed cache is to fill the speed gap between high-speed processors and slow main memory. The cache is located between the main memory and the processor. A cache running at a speed close to the processor helps solve this problem by storing the most commonly used data; therefore, the cost of accessing this data is reduced [12]. The modern cache structure is mainly divided into three levels of cache, including L1 level one cache (including L1 Data and L1 Instruction), L2 level two cache, and LLC level three cache. Among them, L1 and L2 belong to the private cache, and LLC belongs to the shared cache. The shared structure of the cache brings performance improvements but also brings serious security issues. The cache holds a copy of the most recently used data, and the processor can request it in a very short time. If the data requested by the processor is not in the cache (cache miss), the data will be loaded from the main memory and stored in the cache. If it exists in the cache (cache hit), there is no need to access the main memory. Such an obvious time difference can easily be exploited by attackers.

In the past few years, in cache-based side-channel attacks, the attacker obtained sensitive information from the victim through the shared CPU cache. Cache attacks monitor the utilization of the cache to retrieve information about co-resident victims. Sensitive information is usually related to encryption operations (such as signing or decrypting), but can also be extended to other applications [13]. Although the meltdown and spectre attacks in recent years are not directly related to the shared structure of the cache, an indispensable part of these two types of attacks is the cache side-channel attack. If the cache side channel-attack can be prevented, it can affect the success rate of attacks such as Meltdown and Spectre from the side.

Among the existing cache side-channel attacks, there are three types of attacks with very high attack accuracy and concealment: Flush+Flush, Flush+Reload, and Prime+Probe. Flush+Reload was first introduced in [14] and later extended to search for cipher keys for LLC, TLS protocol session messages, or keyboard keystrokes via VM [15 16]. By measuring the time difference between the response cache hit loading data and the response cache miss loading data, the slow loading indicates that the attacker is the only process that touches the cache line, so each reloads will cause the cache miss and will take more time Get data from main memory. However, a faster reload indicates that the victim process has also touched the attacker's cache line, so the attacker has a cache hit and it takes less time to obtain the data.

Compared with the Flush+Reload attack, Prime+Probe is not only applicable to systems with memory deduplication mechanisms, it can be applied to almost any system. The attacker will evict a specific data set from the cache and wait for the victim to evict the monitored data set; when comparing access times, fast access means that the cache line has been touched. For details, see [17].

The Flush +Flush attack proposed in [18] is similar to the Flush+Reload attack, except that the Flush+Flush attack has stronger concealment. Unlike the other two cache attacks, Flush+Flush attack does not perform any memory access, it only relies on accurate cache time, which makes it more concealed and efficient attack capabilities. The attacker process applies the clflush instruction on the shared cache line every time to measure the execution time of the instruction. If victim access occurs, the instruction execution time will increase.

B. Hardware Performance counters

Hardware performance counters are a set of system-specific registers built into x86 (such as Intel and AMD) and ARM processors. It was originally used to count a large number of low-level hardware events related to the execution of the system. They are used with event selectors, which specify specific hardware events and update the counter after the hardware event occurs.

The hardware performance counters were originally designed for software debugging and system performance analysis. In recent studies, more and more people use hardware performance counters to detect security risks and program vulnerabilities in the system [19 20 21]. Because the hardware performance counter can discover the features during the execution of different programs by collecting different events. Using this principle, it can be applied to reflect the security status of the program. The hardware performance counter detection will bring almost negligible performance overhead to the program, and it also makes it an almost perfect detection tool for cache side-channel attacks.

C. Related Work on Detections

In this section, we summarize the latest developments in cache side-channel attack detection mechanisms (with or without machine learning methods) and discuss their advantages and disadvantages. In 2016, the author in [22] established a detection model by analyzing side-channel attacks based on cross-virtual machine caching, namely Flush+Reload and Prime+Probe. The detection method they proposed is to correlate the execution of encrypted applications with the cache miss/hit rate of untrusted virtual machines. But they did not consider whether it would affect their detection success rate under different system loads. Moreover, only using the ratio of hits and misses of cahce as an indicator can provide too little information, and it is easy to misjudge some benign programs. In [23], a three-step detection method is proposed to deal with cache side channel attacks and attacks based on branch prediction. This method helps to eliminate false alarms of attacks in the system and more accurately detects attacks based on branch prediction and DRAM attacks. However, this detection mechanism cannot be used effectively under actual system load conditions (that is, any background noise), which is likely to cause huge errors. Also, this technology has a large detection overhead in the system and is poor in practicability. In [24], the researchers established machine learning models to detect FR attacks and PP attacks by extracting the events of four hardware performance counters and the values of three fixed-function registers. Under no-load conditions, the detection success rate can reach 97%, but under load conditions, the detection success rate drops significantly, even lower than 70%. So far, [9] is the only research work I know that adds Flush+Flush attack detection. Analyzing FF, FR, and PP attacks by counting the different characteristics of hardware counters, the highest detection accuracy can reach 99.51%. However, 12 different hardware performance events are used for modeling in [9], which is significantly different from the actual 4 hardware counter interfaces and cannot be applied to real-time monitoring of cache attacks. Moreover, the above papers model each type of attack separately, which loses the significance of detection as the first door of protection in the actual application process. The advantage of this paper is that using as few hardware events as possible to establish a unified model, the final model can accurately find out whether there is any one of the three cache side-channel attacks.

III. MODEL IMPLEMENTATION

This section is an introduction to the establishment of the detection model of this paper. Before establishing the model, the data under different loads are randomly divided into two parts, separated by a ratio of 8:2. Among them, 80% of the data set is used as the training set, and then 20% of the data set is used as the test set. In this article, tenfold cross-validation is to randomly divide the model ten times to calculate the average accuracy. Overfitting is an unavoidable problem in machine learning. The above operations are also to alleviate possible model overfitting. By completely separating the training set and the test set, the result of the detection model obtained is more objective, accurate, and more convincing.

The following figure1 shows the research flow chart of this paper.

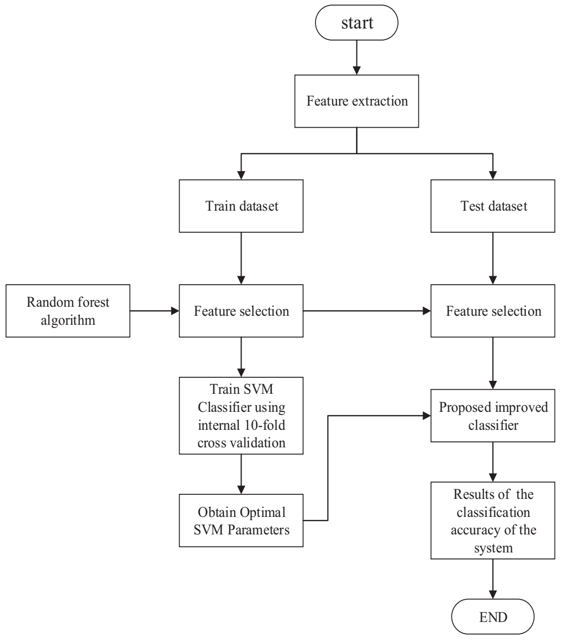
start
Feature extraction
Train dataset
Test dataset
Random forest algorithm
Feature selection
Feature selection
Train SVM Classifier using internal 10-fold cross validation
Proposed improved classifier
Obtain Optimal SVM Parameters
Results of the classification accuracy of the system
END

Fig 1 The flowchart of classification system modeling

The steps of the machine learning algorithm are listed as follows:

- Data extraction: Under different loads of NL, AL, FL, the hardware performance events of FF, FR and PP attack and non-attack are collected through the hardware performance counter at a frequency of 100ms.
- Data dividing: The data in NL, AL, and FL are divided into two parts randomly according to 8:2. Among them, 80% of the dataset is used as a train set, and then 20% of the dataset is used as a test set.
- Model training: The SVM algorithm is used to classify and model the data of the train set, and obtain the classification accuracy of the train set.
- Model testing: Adjust the key parameters in the algorithm and repeat step 3. Until the accuracy is high

and stable. The model established by the train set is verified by using the data of the test set, and finally, the classification accuracy of the model is obtained

- Classification results: The final accuracy of the classification model was obtained by using 10 times cross-validated parameters and finally validating with the test set

IV. Experiment

The experiment was conducted on Lenovo, which has an Intel Core I5-7200U 2.50GHz processor and 8GB RAM, running Ubuntu 16.04. All simulated environmental load tests use SPEC cpu2006. The three different attacks used in this paper are all against the T table in the AES encryption algorithm. The collected hardware counter data is provided by Linux with a performance monitoring tool called perf [25] to monitor hardware and software events. We use the Linux perf event kernel API to call the performance counter. Perf provides a function perf_event_open to retrieve the value of the performance counter in real-time. The sampling interval of the performance counter in this article is 100ms because the lower sampling frequency brings less overhead and has less impact on system performance.

A. *Feature extraction*

To meet the differences between attack and no attack in different system environments, this paper collects performance counter data in three different environments, one is no load, the other is average load, and the third is a full load. In the first scenario, high-resolution data is ensured by running only the victim and the attacking program. In the second scenario, we run two of the test sets in SPEC2006, using two processes to proceed at the same time. This article selects gcc in the Int test set and bwaves in the Float test set. In the case of full load, in addition to the two test sets in the second case, we also added two processes to run bzip2 in the Int test set and dealII in the Float test set. Collect the hardware events of the three types of attacks under three conditions. Due to the imbalance of samples, the prediction tends to be biased towards the category with more samples. This will greatly reduce the model's normalization ability. Such a model often results in high accuracy, but low AUC (Area Under Curve). In order to balance the positive and negative samples as much as possible and improve the stability of the system, the sample size in the attack and non-attack situations in this article is almost the same. The following table1 shows the sample size in different situations. Although running the same program, as the load increases, the running time is significantly longer.

TABLE I. Samples collected for different attacks under different loads.

Attack Type	**System Load**		
	NL	***AL***	***FL***
FF	Attack 1084 No Attack 1117	Attack 1309 No Attack 1522	Attack 1875 No Attack 1904
FR	Attack 1153 No Attack 1117	Attack 1199 No Attack 1231	Attack 1950 No Attack 1978
PP	Attack 772 No Attack 661	Attack 955 No Attack 720	Attack 1362 No Attack 1127

By viewing the perf_event_open function and analyzing the hardware performance events of the system that may be affected by the three attacks, Table 2 shows all the features collected in this paper. It includes 24 haderware performance events related to different attacks, such as L1D, L1I, LLC, ITLB, DTLB, BPU, and CACHE_NODE(local memory access).

TABLE II. Collected features.

Category	**Perf_event_open**	
	PERF_TYPE_HARDWARE	***PERF_TYPE_HW_CACHE***
Event	1.PERF_COUNT_HW_CACHE_REFERENCES, 2.PERF_COUNT_HW_CACHE_MISSES 3.PERF_COUNT_HW_CPU_CYCLES 4.PERF_COUNT_HW_INSTRUCTIONS 5.PERF_COUNT_HW_REF_CPU_CYCLES 6.PERF_COUNT_HW_BUS_CYCLES	7.PERF_COUNT_HW_CACHE_LL READ 8.PERF_COUNT_HW_CACHE_LL WRITE 9.PERF_COUNT_HW_CACHE_L1D MISS 10.PERF_COUNT_HW_CACHE_L1I MISS 11.PERF_COUNT_HW_CACHE_ITLB MISS 12.PERF_COUNT_HW_CACHE_ITLB READ 13.PERF_COUNT_HW_CACHE_L1D ACCESS 14.PERF_COUNT_HW_CACHE_NODE 15.PERF_COUNT_HW_CACHE_ITLB ACCESS 16.PERF_COUNT_HW_CACHE_DTLB ACCESS 17.PERF_COUNT_HW_CACHE_DTLB READ 18.PERF_COUNT_HW_CACHE_DTLB WRITE 19.PERF_COUNT_HW_CACHE_DTLB MISS 20.PERF_COUNT_HW_CACHE_NODE READ 21.PERF_COUNT_HW_CACHE_NODE WRITE 22.PERF_COUNT_HW_CACHE_BPU ACCESS 23.PERF_COUNT_HW_CACHE_BPU MISS 24.PERF_COUNT_HW_CACHE_BPU READ

B. *Feature selections*

Due to the many features extracted, we try to optimize the features and hope to further improve the accuracy of the classifier. Moreover, in the real system, the general performance counter only provides four interfaces, so in order to be able to be applied to real-time monitoring, we screened these 24 features to 4 best features.

In 2001, American scientist Leo combined his proposed Bagging theory with Ho's random subspace algorithm to propose a new machine learning algorithm, namely the random forest algorithm [26]. The random forest algorithm is an integrated machine learning method that uses the bootstrap method to extract multiple samples and establish a decision tree. Based on the prediction results of all the decision trees constructed, the final classification result is generated by voting. Random forests have the following advantages: they do not cause over-fitting, can assess the classification importance of each feature, have good robustness to noise, and the algorithm is relatively easy to understand.

The method of feature selection of random forests is to directly measure the influence of each feature on the prediction accuracy of the model. The basic idea is to rearrange the order of a column of feature values and observe how much the accuracy of the model is reduced. By comparing the worth of r_2_scores with this feature and without this feature:

$$scores = (r_2\_score - r_2\_score1) / r_2\_score \tag{1}$$

r_2_scores represents the degree of fitting of the system when the feature set has the measured feature and r_2_scores represents the degree of fitting of the feature set without the measured feature. So the more important the feature is, the closer the final scores is to 1. The less important the feature is, the closer the final scores is to 0.

To achieve good accuracy under different loads, we separately filter the feature sets under different loads. The following table shows the features with higher feature weights under different loads.

TABLE III. THE SIX MOST WEIGHTED FEATURES UNDER DIFFERENT LOADS.

Category	**System load**		
	No Load	***Average Load***	***Full Load***
Event	DTLB READ	L1D ACCESS	DTLB READ
	L1D ACCESS	DTLB ACCESS	CACHE_NODE READ
	DTLB ACCESS	CACHE_NODE READ	CACHE_NODE
	BPU ACCESS	CACHE_LL READ	DTLB MISS
	CACHE_MISSES	CACHE_REFERENCES	L1D_ACCESS
	CACHE_REFERENCES	CACHE_NODE	DTLB ACCESS

The table3 shows the six features with the highest weights under different loads. To make this model perform well under any conditions, the four features selected in this article are DTLB READ, L1D ACCESS, DTLB ACCESS, and CACHE_NODE (for measuring local memory accesses).

C. Support Vector Machine (SVM)

The classification algorithm chosen in this paper is the support vector machine (SVM). The SVM proposed by Vapnik et al. is a statistically based classification algorithm [27]. The core idea is to map the sample feature space to high-dimensional and use the simple linear classifier to divide the mapped sample space to achieve the minimum experience sharing and confidence range so that the classification has the strongest generalization ability. The SVM kernel function used in this paper is the Radial Basis Function (RBF).

The advantage of the SVM algorithm is that it can be used for linear/non-linear classification, and can also be used for regression, with low generalization error rate. It has good classification and decision-making ability for solving machine learning problems in the case of small samples, which is very suitable as the algorithm model of this paper.

V. RESULT

This paper uses four features to establish a unified machine learning model for FF, FR, and PP, and finally achieves excellent classification accuracy. In this paper, the samples obtained under all three attacks are marked as 1, and the non-attack samples are marked as 0, because for practical applications, as long as the system can detect the attack, it does not need the system to know the specific attack.

From the table below, it can be seen that under different loads, the machine learning models composed of these three attacks have achieved extremely high accuracy.

TABLE IV. RESULTS USING SVM MODEL FOR ATTACK DETECTION

Table Head	**System load**		
	No Load	***Average Load***	***Full Load***
ACCURACY (FR,FF,PP)	99.92%	99.85%	96.58%
ACCURACY (FR) [9]	99.51%	99.50%%	99.44%
ACCURACY (FF) [9]	99.97%	98.74%	95.20%

It can be seen from the above table that under different loads, the machine learning models composed of these three attacks have achieved extremely high accuracy. Compared with the results of choosing different hardware event modeling for FF and FR in paper 9, this paper uses the hardware event modeling under different load conditions to identify the three types of attacks more convincingly and more accurately.

After the classification model is established, its classification performance needs to be evaluated. The parameters we chose were: Precision, F_1 scores (F_1 scores, the average score for each level), and recall rate. Precision is the proportion of correct positive sample size to the correct classification of the sample. The F_1 scores is an indicator used for statistics to measure the accuracy of a binary model. It is often used for information retrieval and considers category balancing. The recall rate is the ratio of the positive sample size to the true positive sample number in the test sample set. The following are the formulas of the four model parameters, where TP represents the number of positive samples that are classified correctly, FP represents the number of positive samples that are classified incorrectly, FN represents the number of negative samples that are classified incorrectly, and TN represents the number of negative samples that are classified correctly:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F_1\_\text{score} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{4}$$

The performance indicators of this model are shown below:

TABLE V. RESULTS USING SVM MODEL FOR ATTACK DETECTION

classification performance	**System load**		
	No Load	***Average Load***	***Full Load***
Precision	1.0	1.0	0.97
Recall	1.0	1.0	0.97
F1 scores	1.0	1.0	0.97
KAPPA	0.9966	0.9956	0.9421

The kappa coefficient in the system is also higher than 0.9 (when the kappa coefficient is greater than 0.8, it indicates that the model and the potential correlation are almost identical). It shows that these four features can clearly distinguish whether there is an attack during the encryption process, regardless of whether it is under load.

or not.

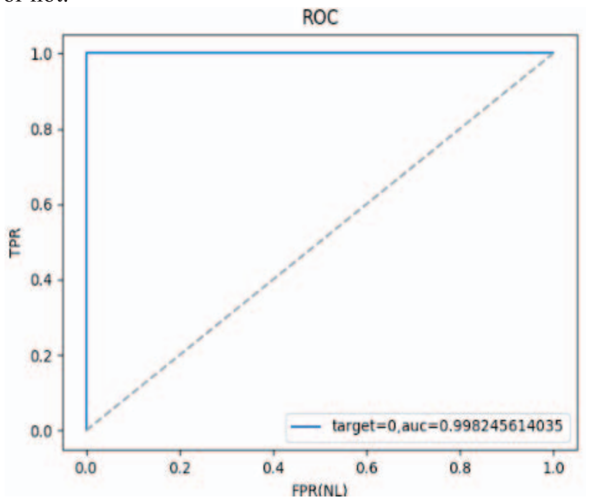

Fig 2. Receiver Operating Characteristic (ROC) Curves for SVM Models while detecting attacks under NL

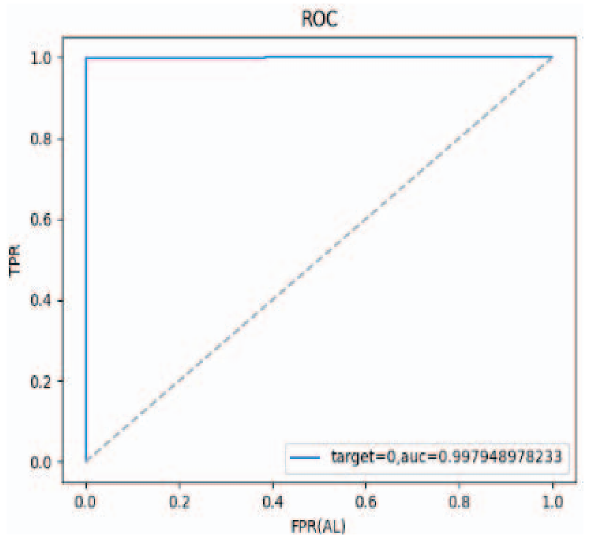

Fig 3. Receiver Operating Characteristic (ROC) Curves for SVM Models while detecting attacks under AL

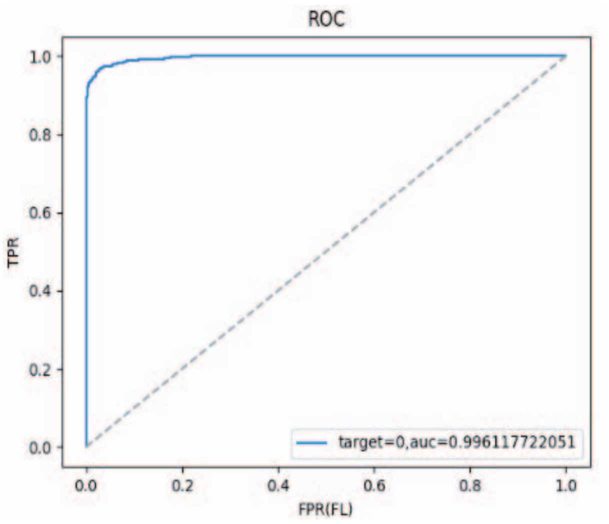

Fig 4. Receiver Operating Characteristic (ROC) Curves for SVM Models while detecting attacks under FL

Figure 2-4 shows the ROC curve of the SVM model, illustrating their ability to diagnose three types of attacks under variable load conditions. The ROC curve represents the ratio of attack cases classified as an attack (true positive) and no attack cases that are incorrectly classified as an attack (false positive) in the SVM classification model. The AUC (Area Under Curve) in the figure is defined as the area under the ROC curve. The AUC value is often used as the evaluation standard of the model because in many cases the ROC curve does not indicate which classifier performs better, and as a value, a classifier with a larger AUC performs better. In the case of NL, the AUC is 0.9982; in the case of AL, the AUC is 0.9979; in the case of FL, the AUC is 0.9961. Under different loads, the AUC of the model is close to 1, indicating that the classification effect is excellent.

In summary, the detection model can achieve good results regardless of whether it is under actual load or no load.

VI. Conclusion And Discussion

Aiming at the three most popular cache-based side-channel attacks, this paper uses different low-level events collected by hardware counters to obtain 4 optimal feature subsets after feature screening, and use the SVM model to detect, and finally obtain excellent classification accuracy. It shows that the system is suitable for analyzing three kind of cache side-channel attacks based on the AES encryption system.

Another work of this paper is that we simulate actual system load conditions to provide an experimental evaluation. We used SPEC benchmark tests to create a no-load environment, an average load environment, and a full load environment to train and evaluate the ML model and achieved good results. It shows that the model composed of these four features has a high detection efficiency for side-channel attacks under variable system load conditions. Our results show detection accuracy of 99.92%, 99.85%, and 96.58% for attacks in the case of NL, AL, and FL conditions, respectively.

To know the reason for the high accuracy of the model, this paper further analyzes the relationship between the features in the case of no attack and three attacks. We choose L1D ACCESS as the object of analysis. The following figure shows the distribution of L1D ACCESS(AL) collected under four conditions, namely three attacks and no attacks.

From the figure5-7, the x-axis shows the value of the L1D ACCESS counter within 100ms. The y-axis on the left shows the number of statistics in the same interval, and the y-axis on the right shows the proportion of the sample size in a certain interval in all samples. There is a clear difference between the L1D ACCESS event when there is no attack and when there is an attack.

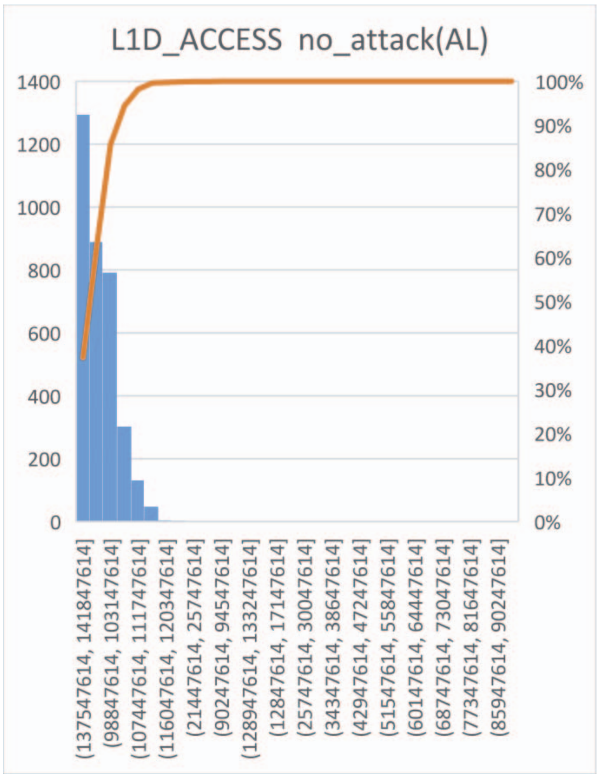

Fig 5. Distribution map of L1D ACCESS under no attack

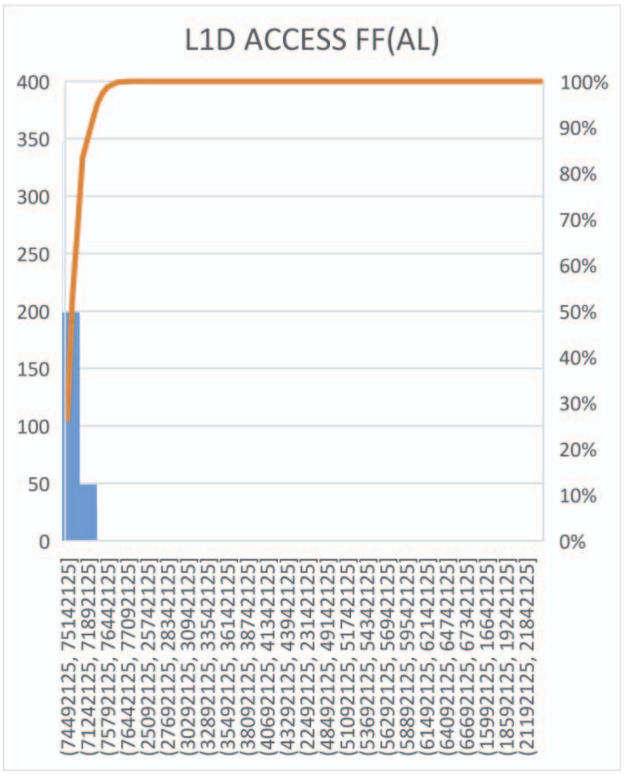

Fig 7. Distribution map of L1D ACCESS under FF attack

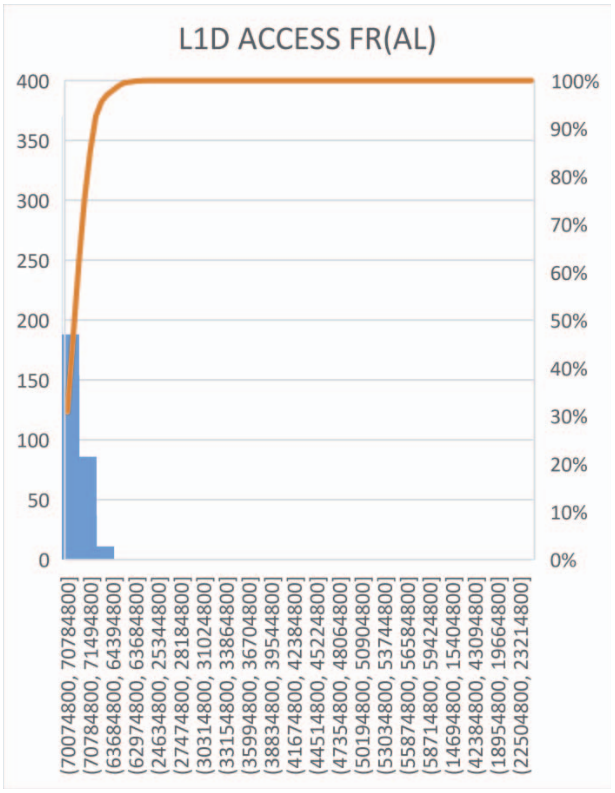

Fig 6. Distribution map of L1D ACCESS under FR attack

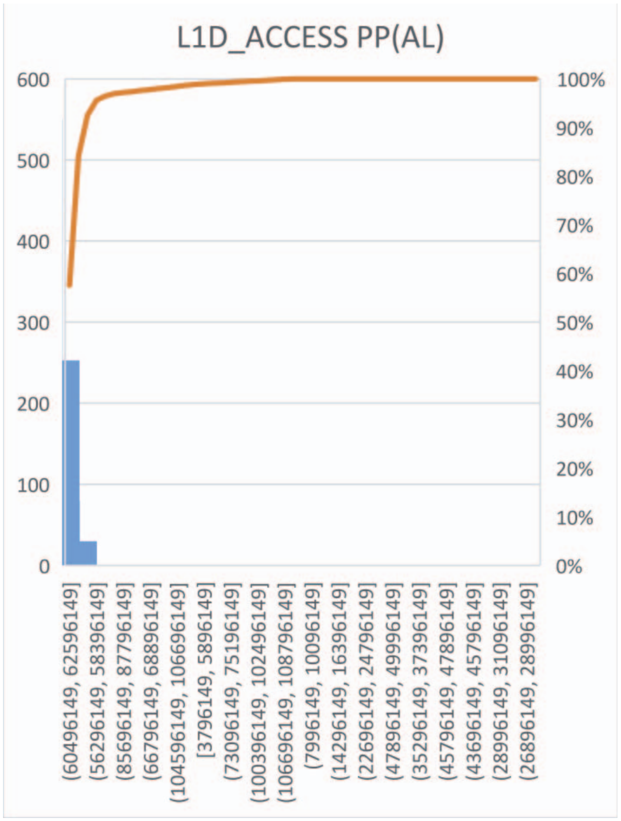

Fig 8. Distribution map of L1D ACCESS under PP attack

In the case of no attack, maintain a high level, generally more than 100000000; but in the case of attack, the L1D access value of each 100ms event will significantly decrease, and all three attacks will drop to about 70000000. Because in the case of an attack, some data will be refreshed or overwritten, which will affect the access speed, so the value of L1D ACCESS within a unit time (100ms) will drop. This shows that the hardware performance event L1D ACCESS is suitable as a feature for distinguishing cache side-channel attacks with or without.

Although this paper has got a good detection accuracy, it does not establish an actual detection module in the system. In the future, hardware information can be obtained directly from hardware performance counters for timely detection of intrusions, which can help the operating system take preventive measures, such as stopping or killing identifiable malicious processes or completely isolating the victimized process. In the future, we will further analyse the accuracy of the model in the actual use of the system as well as the performance overhead of adding the detection module to the system, and try to apply the detection system to the latest attacks about Meltdown and Spectre. Try to increase the sampling rate of the hardware counter to further explore the differences and connections between different attacks.

REFERENCES

[1] Spreitzer, Raphael, et al. "Systematic classification of side-channel attacks: A case study for mobile devices." *IEEE Communications Surveys & Tutorials* 20.1 (2017): 465-488.

[2] Lipp, Moritz, et al. "Meltdown: Reading kernel memory from user space." *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 2018.

[3] Kocher, Paul, et al. "Spectre attacks: Exploiting speculative execution." *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019.

[4] Inci, Mehmet Sinan, et al. "Seriously, get off my cloud! Cross-VM RSA Key Recovery in a Public Cloud." *IACR Cryptol. ePrint Arch.* 2015 (2015): 898.

[5] Inci, Mehmet Sinan, et al. "Cache attacks enable bulk key recovery on the cloud." *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, Berlin, Heidelberg, 2016.

[6] Brickell, Ernie, et al. "Software mitigations to hedge AES against cache-based software side channel vulnerabilities." *IACR Cryptol. ePrint Arch.* 2006 (2006): 52.

[7] Crane, Stephen, et al. "Thwarting Cache Side-Channel Attacks Through Dynamic Software Diversity." *NDSS*. 2015

[8] Chiappetta, Marco, Erkay Savas, and Cemal Yilmaz. "Real time detection of cache-based side-channel attacks using hardware performance counters." *Applied Soft Computing* 49 (2016): 1162-1174.

[9] Mushtaq, Maria, et al. "Nights-watch: A cache-based side-channel intrusion detector using hardware performance counters." *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*. 2018.

[10] Mushtaq, Maria, et al. "Run-time detection of prime+ probe side-channel attack on AES encryption algorithm." *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2018.

[11] Mushtaq, Maria, et al. "Machine learning for security: The case of side-channel attack detection at run-time." *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2018.

[12] Page, Dan. "Theoretical use of cache memory as a cryptanalytic side-channel." *IACR Cryptol. ePrint Arch.* 2002.169 (2002): 1-23.

[13] Irazoqui, Gorka, and Xiaofei Guo. "Cache side channel attack: Exploitability and countermeasures." *Black Hat Asia* 2017.3 (2017): 1-72.

[14] Gullasch, David, Endre Bangerter, and Stephan Krenn. "Cache games--bringing access-based cache attacks on AES to practice." *2011 IEEE Symposium on Security and Privacy*. IEEE, 2011.

[15] Gruss, Daniel, Raphael Spreitzer, and Stefan Mangard. "Cache template attacks: Automating attacks on inclusive last-level caches." *24th {USENIX} Security Symposium ({USENIX} Security 15)*. 2015.

[16] Yarom, Yuval, and Katrina Falkner. "FLUSH+ RELOAD: a high resolution, low noise, L3 cache side-channel attack." *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 2014.

[17] Irazoqui, Gorka, Thomas Eisenbarth, and Berk Sunar. "S $ A: A shared cache attack that works across cores and defies VM sandboxing--and its application to AES." *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015.

[18] Gruss, Daniel, et al. "Flush+ Flush: a fast and stealthy cache attack." *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, Cham, 2016.

[19] Bahador, Mohammad Bagher, Mahdi Abadi, and Asghar Tajoddin. "HPCMalHunter: Behavioral malware detection using hardware performance counters and singular value decomposition." *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE, 2014.

[20] Tang, Adrian, Simha Sethumadhavan, and Salvatore J. Stolfo. "Unsupervised anomaly-based malware detection using hardware features." *International Workshop on Recent Advances in Intrusion Detection*. Springer, Cham, 2014.

[21] Wang, Xueyang, et al. "Confirm: Detecting firmware modifications in embedded systems using hardware performance counters." *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015.

[22] Zhang, Tianwei, Yinqian Zhang, and Ruby B. Lee. "Cloudradar: A real-time side-channel attack detection system in clouds." *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, Cham, 2016.

[23] Alam, Manaar, et al. "Performance Counters to Rescue: A Machine Learning based safeguard against Micro-architectural Side-Channel-Attacks." *IACR Cryptol. ePrint Arch.* 2017 (2017): 564.

[24] Allaf, Zirak, Mo Adda, and Alexander Gegov. "A comparison study on flush+ reload and prime+ probe attacks on aes using machine learning approaches." *UK Workshop on Computational Intelligence*. Springer, Cham, 2017.

[25] Dongarra, Jack, et al. "Accurate cache and TLB characterization using hardware counters." *International Conference on Computational Science*. Springer, Berlin, Heidelberg, 2004.

[26] Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5-32.

[27] Drucker, Harris, et al. "Support vector regression machines." *Advances in neural information processing systems*. 1997.