Anomaly detection using hardware performance counters on a large scale deployment

Malcolm Bourdon, Eric Alata, Mohamed Kaâniche, Vincent Migliore, Vincent Nicomette, Youssef Laarouchi

Anomaly detection using hardware performance counters on a large scale deployment

Short Paper

Malcolm Bourdon
EDF R&D, LAAS-CNRS
Palaiseau, France

Eric Alata and Mohamed Kaaniche and Vincent Migliore and Vincent Nicomette
LAAS-CNRS, Toulouse, France

Youssef Laarouchi
EDF R&D, Palaiseau, France

***Abstract*—The last recent years witnessed a massive and fast deployment of Internet of Things (IoT) devices. Most of them have not been designed with a careful analysis of security requirements, which makes them likely to include multiple vulnerabilities. Moreover, as these devices include various communication interfaces, they have become a privileged target for attackers. As a consequence, large scale attacks, such as Mirai, must be considered seriously and it is crucial to design and implement protection and intrusion detection mechanisms to mitigate the threats associated to the use of IoT devices in our daily activities as well in critical environments.**

This paper proposes an anomaly detection approach, in the particular context of a large scale deployment of identical IoT devices. Furthermore, we consider an attacker who can install and execute malicious software while continuing to execute legitimate software, in order to stay invisible as much as possible. The approach is based on the statistical analysis of Hardware Performance Counters (HPC) collected at a regular basis from these identical devices, and to highlight the outliers corresponding to significant deviations with respect to normal usage scenarios. This idea relies on the intuition that it is very difficult for the attacker to add some malicious software in a corrupted device without perturbing the HPCs. This paper presents this approach and the first experiments carried out to assess its relevance.

keywords : Cyber security, Anomaly detection, Hardware performance counters, Internet of Things (IoT).

1. Introduction

The expansion of the Internet Of Things (IoT) in the market is quite noticeable nowadays. These connected devices improve quality of life and usability by providing interoperability, remote control, monitoring or alert and notification services. Such features, coupled with a high level of connectivity, greatly increase the attack surface. Therefore, large scale attacks are one of the major threats.

The Mirai malware is one of the most representative examples. It can provide remote access to devices that have poor `telnet` service implementation, and once infected, such devices can act as malicious botnets and infect other devices [1]. Furthermore, Mirai source code is now public which increases the level of threat since many IoT devices are vulnerable.

In this paper, we consider the particular context of a large scale deployment of IoT devices running on the same hardware, executing the same software, with potentially a different usage profile according to the environment where the device is deployed. All these identical devices are assumed to communicate with a remote server that is managed by the service provider. This is the case for example of IoT devices deployed by energy suppliers to monitor and optimize energy consumption in individual homes, in buildings or at a larger scale. We present a novel lightweight approach that is aimed at the efficient detection of compromised devices by monitoring the behavior of the whole set of devices and detecting potential deviations with a normal usage profile. The anomaly detection algorithm, running at the server, is based on the statistical analysis of *Hardware Performance Counters* (HPC) sent by each device to the server on a regular basis. Such an architecture minimizes the computation overhead on the devices, that have generally constrained resources.

Some research works have already investigated the use of Hardware Performance Counters for security purposes, but only few of them for constrained devices. To the best of our knowledge, most of them rely on the development of a sophisticated behavioral model of the devices at pre-deployment stage. The approach proposed in this paper is an alternative that does not require any predictive model to perform the anomaly detection. It mainly consists in analyzing the different HPCs data received from all the devices and identifying outliers.

The threat model corresponds to an attacker who can install and execute a malware on the device while continuing to execute the legitimate software. Such malware may be designed to steal confidential data, carry out denial of service (DoS) attacks, or use the power of the device e.g., to mine bitcoin. In addition, as the attacker wants to let the user think that the device operates correctly, he does not modify or stop the legitimate software. We argue that according to these assumptions, it is very difficult for an attacker to perform such attacks without disrupting significantly the values of the HPCs compared to those captured by non

compromised devices. As we compare the behavior of a large set of devices, and we assume that an attacker cannot corrupt a majority of devices at once, it is very likely that the detected anomaly (or outlier) detection is not due to noise or to an unsual behavior of the legitimate software.

Such an approach does not require any predictive model of the behavior of the HPC. The anomaly detection is based on the raw values regularly sent to the server. We also assume that there is a secure channel between the device and the server.

This paper aims at describing this anomaly detection approach and presenting the results of the first experiments that we carried out to assess the relevance of our approach.

Section 2 discusses some security related research works based on hardware performance counters. Section 3 is dedicated to the presentation of hardware performance counters: how it is possible to use them, which characteristics of the system they are able to measure. In Section 4 we describe the experiments we have carried out to assess our anomaly detection approach, as well as their results. Finally, Section 5 concludes and presents future work.

2. Related Works

Hardware Performance Counters (HPC) were initially created for debugging purposes. However, some research works have already proposed to use them for security. The main idea in most of these works is that the behavior of the processor's HPCs during the execution reflects the behavior of the software executing on the processor. Hence, it is possible create a model of the software's execution on the device from the HPC values. Furthermore, using HPCs for security purposes presents several advantages: collecting them have a tiny impact on the performance and they are hard to simulate for an attacker.

Some research works investigated the use of HPCs in order to detect malware, e.g., [2], HPCMalHunter [3], HLMD [4]. For that purpose, they model the impact of attacks on the HPCs and check them during the execution. [2] uses unsupervised learning techniques to create this model, while HPCMalHunter [3] uses Single Value Decomposition based on SVM to classify the observed behavior as corresponding to benign or malicious program execution. [4] only uses malicious programs and tries to establish signatures representative of the different kinds of attacks performed.

Some other related works focus on specific attacks or specific type of software. For example, [5] proposes to use HPCs to detect time-fragmented cache attacks on AES. A threshold-based anomaly detection approach is developed for that purpose.

Some papers point out that these counters may not reflect the actual number of occurrences of the events they should describe [6]. Our approach should not be affected by this problem, generally referred to as overcounting, because it relies on the detection of outliers among a set of hardware identical devices. The overcounting is likely to affect all devices in the same way and should not prevent the identification of outliers.

The closest research work to the approach proposed in this paper is, to the best of our knowledge, ConFirm [7]. ConFirm only deals with firmware, is located on the bootloader and randomly inserts checkpoints into the code from which the HPCs are retrieved. Some pre-deployments models are built for each subroutine so that the verifier can compare the reported values with the expected one. The main difference with our approach is that we don't need to do these pre-deployment tests on each subroutine because we monitor a large number of devices and compare them. In addition, our approach must also be adapted to devices running a real operating system and not just firmware.

3. Hardware Performance Counters

Hardware performance counters (HPCs) are hardware units that provide measurements on certain micro-architectural or architectural events. They do not necessarily exist on all processors but tend to be more present in most modern processors. They can be implemented differently depending on the processor family, but their main utility remains the same for all families. More precisely, they allow to recover certain events such as L1 cache misses, hits or refills, instructions retired, bus accesses, etc. They are intended to be used for debugging purposes, but they can also be used for other purposes, such as optimization, profiling, tracing, etc.

Some specific processor registers are dedicated to the storage of these counters. Depending on the processor, there are two to eight of these registers to store the counter values of a pool of events that can range from twenty to more than one hundred events (depending on the different processors). The configuration of these registers requires kernel privilege but it is possible to configure them so that their read mode access is allowed from the user space. HPC values can be retrieved via specific instructions or by using certain dedicated interrupts called *Performance Monitoring Interrupt*.

Some tools, such as `PAPI` or `perf_event`, have been developed to profile software with HPCs for different processors. We took advantage of these existing tools to build our own code specially adapted for our own experiences.

We had to choose the counters that could best show a significant difference when malware is running at the same time as legitimate software. We selected counters that best cover the different functionalities of a hardware and software execution platform: cache memories, communication buses, prediction branches, etc. We also selected these counters using other research works, such as the ones mentioned above, which recommend some interesting counters in their work. More detailed information on the selected counters is provided in the following section.

4. Experimentation

Our approach is mainly based on comparing HPCs between a large number of identical devices to expose outliers.

Since it was impossible for us to obtain hundreds of identical devices, we repeated experiments on a set of 10 identical hardware devices under the same conditions. All devices were restarted at the end of each experiment to prevent any side effects between two consecutive experiments. The devices we used for the experiment were Raspberry Pi 3B+, with ARM A53 processors. These processors have 6 registers to monitor a pool of 28 different architectural or micro-architectural events. A 7th register that cannot be configured stores processor cycles. The counters chosen for our experiment are as follows: Level 1 data cache refill, Level 1 data cache access, Instruction architecturally executed, Exception taken, mispredicted or not predicted branch speculatively executed and Bus access.

We have developed three software programs that perform different simple tasks. *Software1* sends values to a server via the Internet connection, *software2* processes files and data, and *software3* makes intensive use of the processor by compiling a program. In this experiment, *software1* and *software2* correspond to the legitimate software and *software3* represents the added malware. The idea is then to see if there are easily noticeable differences in HPC behaviors between the two sets of software. It should be noted that we designed the experiments in such a way that it was possible to modify some of their parameters (task frequency, data size, etc.) and easily create different workloads.

To perform our experiments, the process is pinned to a single core and the counters are stored every 5 seconds in a local file on the device. The experiment takes 30 minutes and the file containing the counters is sent to the server, which then performs statistical analyses on the results. We have had 90 devices run *software1* and *software2*, and 10 ”malicious” devices also run *software3*. The time between two consecutive counter storage (5 seconds) and the time between two consecutive communications with the server for HPC recovery were empirically chosen for this experiment. They must be adapted to each experiment according to different constraints: the amount of data to be sent, the network bandwidth, the time interval at which anomalies are to be detected, etc.

The anomaly detection algorithm we chose for these first experiments is the local outlier factor (LOF) algorithm. This algorithm is based on the comparison of local density around different points. The local density is calculated with k neighbors, which was set at 20 in our experiments. This algorithm also provides a score describing how ”normal” this local density is. We used the LOF code implemented in the software *sklearn* with the contamination parameter (number of suspect outliers) set to ”auto”.

To reduce the number of inputs per device to be processed by the anomaly detection algorithm, we calculated descriptive statistics for each counter for each device throughout the experiment. These statistics are as follows: mean, standard deviation, minimum, first quartile, median, third quartile, maximum.

To avoid the so-called *curse of dimensionality* (indicating that anomaly detection or the learning algorithm may be less effective when there are many features) and to perform simple experiments first, only one descriptive statistic was selected per counter and used to perform anomaly detection on a dimensional vector. Another advantage of such a simple experiment is that it provides graphs that are easy to interpret.

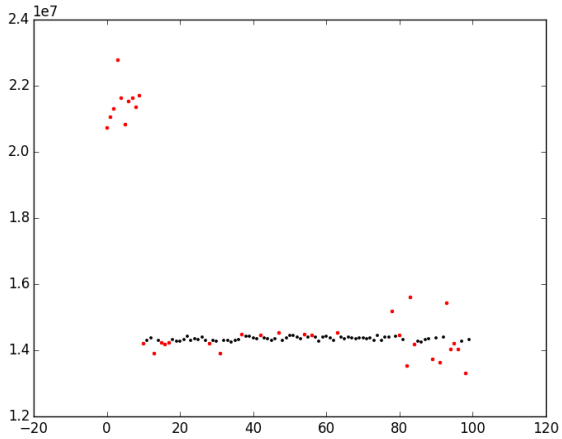

Figure 1. Mean of L1 data cache refill by device

Figure 1 shows the result of the anomaly detection algorithm for the average number of L1 data cache refills per device. The red dots correspond to the devices considered as outliers by the algorithm. The test gives good results. Indeed, in our figure, the abscissa axis corresponds to the number given to the device, and the ten first points, which are detected as outliers, actually correspond to the ”malicious” ones. For each counter we tested, the same successful result was obtained. However, we can also notice some false positives, because some legitimate devices are slightly out of bound compared to the others for this particular graph and are considered as outliers. Moreover, the edge of the main cluster is as well labeled as outlier. But this graph only represents one statistic of one counter, and does not show the actual score given to the device. So, to aggregate all our results, we summed the negative outlier factor given by the LOF algorithm for each counter and statistics in order to aggregate all the results and obtain a global score per device. The lower the score, the less ”normal” the device is. Figure 2 represents the score per device. In this experiment, green dots, which have a different score than all other devices, correspond to devices that execute the malicious software in addition to the legitimate software.

This first experiment is promising and it should be interesting to use multi-dimensional vectors to confirm the results. However, the choice of these dimensions is not straightforward. Principal Components Analysis (PCA) is a statistical method generally used to select the most relevant dimensions. But it was impossible in our experiments to use PCA because this analysis requires that we have access to representative samples of attacks. We carried out an experiment with empirically chosen vectors of two dimensions. These vectors contain the same descriptive statistics

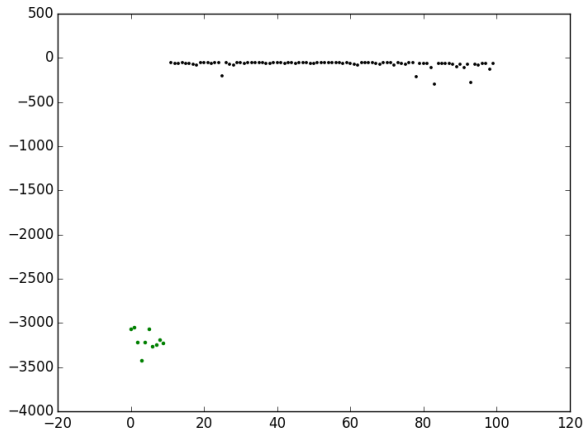

Figure 2. sum of the LOF algorithm scores for one dimension vectors by device

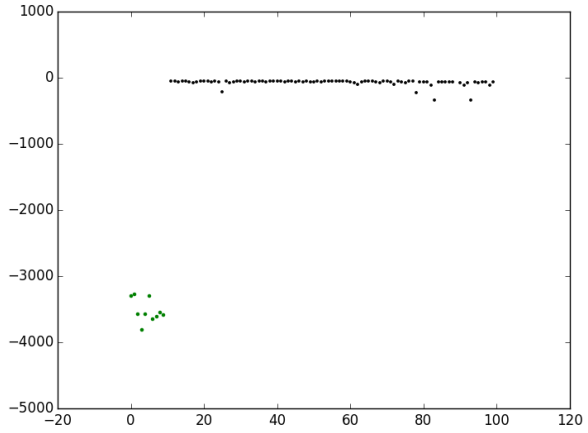

Figure 3. sum of the LOF algorithm scores for two dimensions vectors by device

of two following counters (for example the mean of the first counter and the mean of the second one). Hence we processed the algorithm on the same amount of vectors as previously but with two dimension vectors. The idea here was to correlate different device's counters in our analysis without adding much complexity and computation. Figure 3 represents the score obtained for these two dimensional vectors. It can be seen that the Figure 3 and the Figure 2 are quite similar and give good results. Other experiments need to be carried out to better parametrize and optimize the techniques used to detect the anomalies, but these first experiments give positive results, which confirm that the approach we propose, based on the comparison of HPCs and detection of outliers, is promising.

5. Conclusion and future work

This paper has proposed an anomaly detection approach based on the analysis of HPCs for massively deployed identical devices. The preliminary experiments we carried out provide promising results and highlight that it is possible to distinguish different software executions on devices using these HPCs. For future work, we plan to continue our experiments and optimize the anomaly detection process, for example by changing the way we transform the raw HPC values, and by choosing more suitable vectors to be processed by the anomaly detection algorithm. We also plan to add more dimensions to our vectors and explore other statistical techniques to process the data. Another idea is to experiment with different sets of hardware performance counters and thus create a guide to help to choose the best events to take regarding our method. Finally, we need to measure the precision and accuracy with which we detect anomalies thanks to these HPCs, and determine the differences we can detect in software behavior when using HPCs.

References

[1] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[2] A. Tang, S. Sethumadhavan, and S. J. Stolfo, "Unsupervised anomaly-based malware detection using hardware features," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2014, pp. 109–129.

[3] M. B. Bahador, M. Abadi, and A. Tajoddin, "Hpcmalhunter: Behavioral malware detection using hardware performance counters and singular value decomposition," in *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE, 2014, pp. 703–708.

[4] ——, "Hlmd: a signature-based approach to hardware-level behavioral malware detection and classification," *The Journal of Supercomputing*, pp. 1–32, 2019.

[5] I. Prada, F. D. Igual, and K. Olcoz, "Detecting time-fragmented cache attacks against aes using performance monitoring counters," *arXiv preprint arXiv:1904.11268*, 2019.

[6] S. Das, J. Werner, M. Antonakakis, M. Polychronakis, and F. Monrose, "Sok: The challenges, pitfalls, and perils of using hardware performance counters for security," in *Proceedings of 40th IEEE Symposium on Security and Privacy (S&P19)*, 2019.

[7] X. Wang, C. Konstantinou, M. Maniatakos, R. Karri, S. Lee, P. Robison, P. Stergiou, and S. Kim, "Malicious firmware detection with hardware performance counters," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 160–173, 2016.