Adversarial Attack on Microarchitectural Events based Malware Detectors.

Sai Manoj Pudukotai Dinakarrao, Sairaj Amberkar, Sahil Bhat, Abhijitt Dhavlle, Hossein Sayadi, Avesta Sasan, Houman Homayoun and Setareh Rafatirad George Mason University, Fairfax, VA, USA

Problem:

Our computer is vulnerable to many sophisticated attacks and attackers exploits the vulnerabilities of our system and deploy malware applications. Malware are the malicious software or a program designed by attackers to infect the computing system and use for harmful purpose like Denial-of-service attacks, Ransomware attacks. To detect these type of malicious program software techniques including signature and semantics-based techniques are used. As the complexity of system increases, software techniques becomes overhead to system. So, we need to find alternative to detect malwares. Hardware Malware Detection (HMD) approaches are proposed to detect malware by utilizing the low-level microarchitectural hardware events. Various paper have shown that by deploying Machine Learning model fed with low-level microarchitectural events can aid in differentiating benign and malware applications. Low level microarchitectural events are captured by Hardware Performance Counters (HPCs). ML based malware detectors (HMD) can be implemented in microprocessor hardware with significantly low overhead as compared to the software based methods and detection is also fast few clock cycle.

This work proposes an adversarial attack on HMDs in which the adversarial samples are generated through a benign code that is wrapped around a benign or malware application to produce a desired output class from the embedded ML-based malware detector.

Motivation for the problem :

The main focus of this work is to create a false alarms for the running application like malware classified as benign and benign classified as malware in order to weaken the trust on the embedded defense. Therefore, it shows the scope of an attack which is undetectable by HMD.

Motivation for the solution :

To explore the possibility of misusing the HMD detector. So that we can counter the problem and address the solution before an attacker gets benefit from it.

Details of the solution proposed:

Theory:

We assume that, the victim's defense system to be a blackbox such that the HMD detector is unknown to a attackers. First attacker need to mimic the behavior of HMD. for this we need to perform reverse engineering to mimic the functionality of the victims HMD. The reverse engineering process is described in Fig 1.

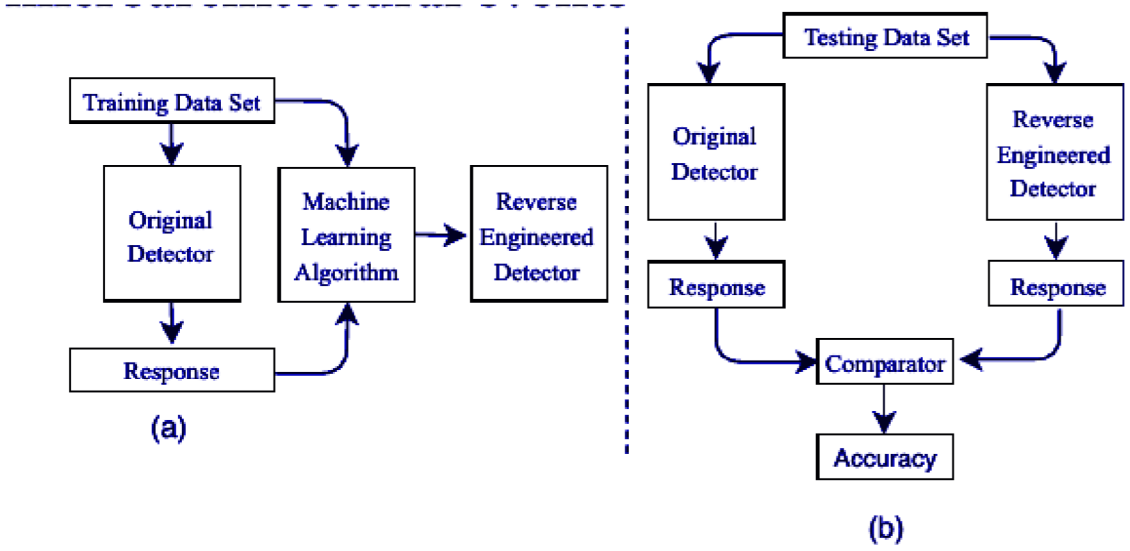

Fig 1. Reverse Engineering Process.

In order to reverse engineer the victim's HMD, we first create a training dataset that contains benign and malware applications. The victim's HMD is fed with all the training data and the response are recorded. This responses are utilized as label to train different ML classifiers in order to mimic the functionality of the victim's HMD as shown in Fig 2(a). Then training dataset is fed to both the original and reverse engineer detector and compare their response. Which reverse engineer detector's accuracy is closest to the victim's detector mimics the victim's HMD.

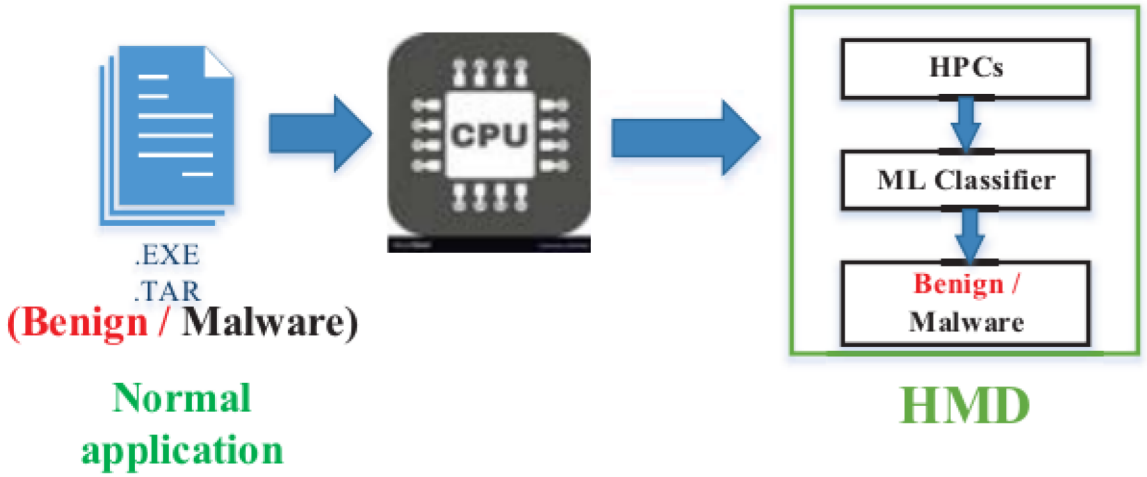

Fig 2(a). Process of utilizing HPC with ML for malware detection

Once the reverse engineered HMD is build, we need to determine the level of perturbations that need to be injected into HPC patterns in order to get the application misclassified. To determine the number of such HPC events to be generated, we deploy an

adversarial sample predictor. To perturb the HPC patterns, they use a low-complex gradient loss based approach, similar to FGSM.

$$X^{adv} = X + \varepsilon sign(\nabla_x L(\theta,x,y))$$

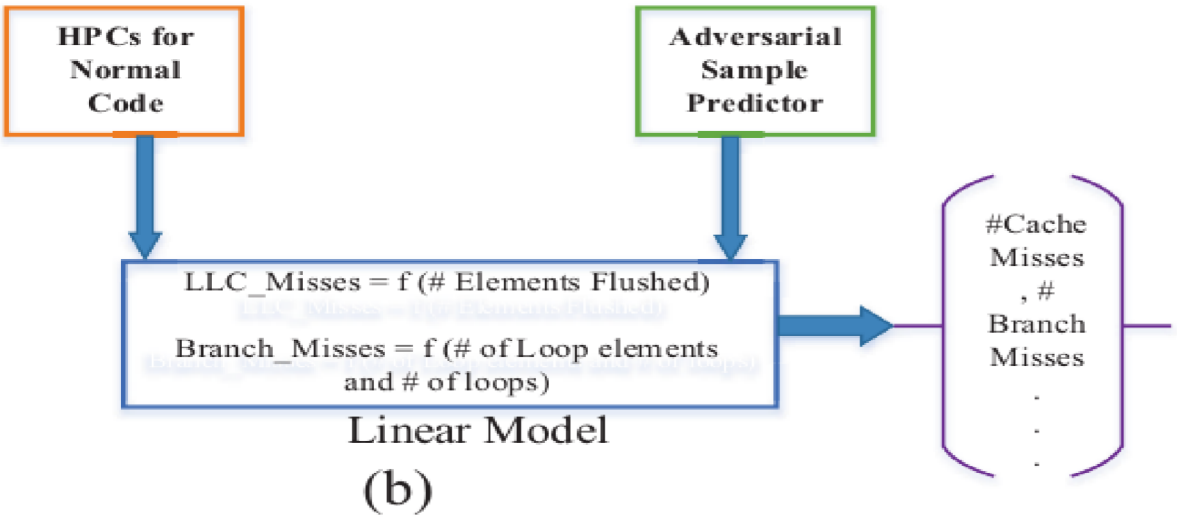

Fig 2(b) : Determining parameter of adversarial code generator with the aid of adversarial HPC predictor

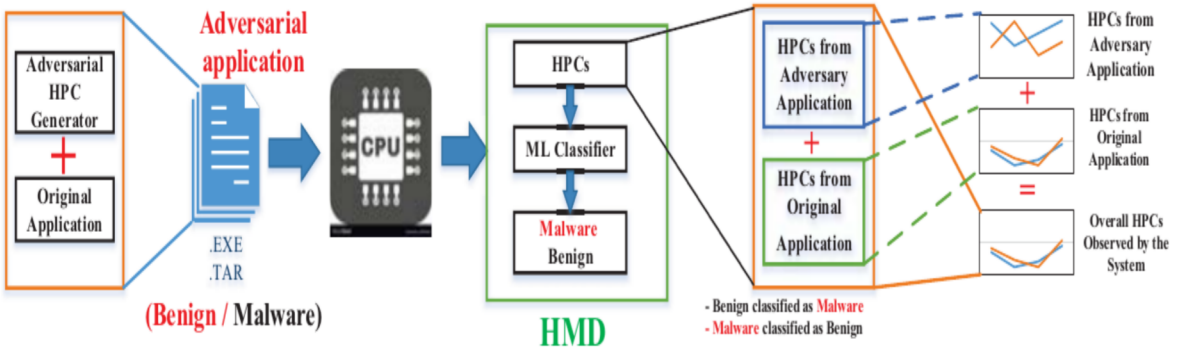

Fig 2(c) : process of adversarial HPC generator embedded into original application yet spawned as separate thread leading to adversarial output.

In order to generate the required number of HPCs, we craft an application that spawns as a separate thread and generates the additional number of HPC events that makes the overall HPC count similar to the predicted HPC count by the adversarial HPC predictor. We show the pseudo code to create adversarial LLC load misses and branch misses. In order to generate LLC load misses, an array of size n is initially loaded from the memory and flushed to generate LLC load misses. N is the size of array and K s the number of elements flushed. Once the adversarial sample predictor predicts the number of LLC load misses to be generated to craft an

adversarial sample, the N and K are determined accordingly. In similar manner, branch misses and branch instructions are generated. Adversarial HPC generator does not interfere with the original application's source code, yet is able to mislead the embedded defense mechanism. HPC generator application is spawned as separate thread together with normal application make overall HPC generated by the modified application close to the malware application HPC leading to the misclassification.

Implementation :

The application are executed on an Intel Xeon X5550 machine running Ubuntu 14.04 with linux 4.4 kernel. To extract the HPC information we used perf tool. It exploits perf-event-open function call in the background which measure multiple events simultaneously. Benign application include MiBench benchmark suite, Linux system program, browsers, text editors and word processor. Linux malware is collected from virustotal.com and virusshare.com. Malware application includes five classes of malware comprising 607 Backdoor, 532 Rootkit, 2739 Virus, 1264 Worm and 7221 Trojan samples. The adversarial sample predictor is implemented in python using the cleverhans library. The adversarial sample generator is implemented using C and execute on a linux terminal as a shell script that facilitates to execute the user/attacker's application in parallel.

It compares the number of LLC load misses and Branch misses of a benign application before and after an adversarial attack. Fig shows the number of LLC load misses in normal case. For this the HPC pattern predicted by the adversarial sample is shown in Fig . One can observe that there exist some spikes in the pattern compared to the normal HPC pattern, as marked by circle. Fig shows the HPC generated when the application is integrated with the adversarial HPC generator. In similar manner, they depict the branch misses shown in Fig .

The neural network based HMD achieves an accuracy of 82.76% with normal samples. When the application is integrated with the proposed adversarial sample generator application, the accuracy reduces to 18.04%. Similarly, a drastic reduction in precision, F1-score and recall are observed with the proposed attack on different applications.

	Accuracy	Precision	F1-score	Recall
Before	82.7%	80.0%	80.0%	83.0%
After	18.3%	45.0%	10.0%	18.0%

Critique:

The paper shows how to misclassify a benign application by adding some LLC load misses and Branch misses predicted by adversarial sample predictor but don't comment anything about how to misclassify a malware application. How to reduce the HPC value of malware application so that the HMD detector will unable to detect it.

The Accuracy of the mimic HMD is drastically reduced form 82% to 18% as we do an adversarial attack on it, which is a very strong point of this paper.
HMD detector detect an application benign or malware after it runs completely on a processor. If an application is a malware and it run completely before detecting by HMD then the system is already corrupted and there is no point of detecting it now. So what we can do, we can add checkpoints to the application that is running on a processor where each processor act as a window. After every checkpoint, the HMD verify the running application and calculate the possibility of being it malware. If it is malware, system will throw an error to the user that the malware is detected and interrupt the application.