Efficient Hardware Malware Detectors that are Resilient to Adversarial Evasion

Md Shohidul Islam, Student Member, IEEE, Khaled N. Khasawneh, Member, IEEE,
Nael Abu-Ghazaleh, Senior Member, IEEE, Dmitry Ponomarev, Senior Member, IEEE, and Lei Yu, Senior Member, IEEE
IEEE Transaction on computers, 29 march 2021.

Problem adressed in this paper :

Malicious attackers compromis system by installing malware and gain access to a system or steal sensitive information. Preventing these attacks is difficult, detecting malware is becoming more complicated. Hardware Malware Detectors (HMDs) have recently been proposed to make system more malware-resistent. HMD classifies malware based on low-level hardware features such as instruciton mixes, memory references patterns, and architectural state information such as cache miss rates and branch prediction rates. HMD offer a substantial advantage to defenders because the detector is always on and has a little impact on performance or power. In this paper, the explore whether attacker can adapt malware to create evasive malware, that bypass HMD detection. If the HMD is compromised, is it possible to made it resilient to evasion.

- Can HMDs be reverse-engineered?
- If we know the model of detector, can malware developers modify malware to avoid detection?
- If we retrained HMD with the evasive malware can it detect malware?
- After showing that the current generation of HMD is evadable, we explore whether the new HMDs can be constructed that are robust to evasion.
- Having shown that the RHMD principle makes detection evsion-resilient, then how to control the base detectors switching behaviour to optimize detection and resilience jointly?

Motivation for the problem :

To check, how effective is HMD as a solution for malware detection and whether an attacker can able to evade HMD to miss-classify a malware. How much and how istruction needs to be added to a malware so that it can evade HMD. If HMD is compromised, then there is any approach we can make our HMD resilient to such type of evasion.

Motivation for the solution :

HMD is known to best in buisness to detect malware because the detector is always on and has little impact on performance or power. If HMD is compromised then the attacker will have upper hand. In this never ending fight between attacker and researcher, we always look up for all possible ways so that we can find the vulnerabities and fixed them before an attacker do.

Theory :

Reverse-Engineering HMDs :

The adversary prepares a training dataset that is composed of both malware and regular programs. Use this dataset to query the victim HMD and record its detection descision. This descision used as the label for the data in constructing the reverse-engineered detector. The adversary used various ML classifier to build the new reverse-engineered detector and selects whose output is matched with victim’s HMD output.

Since the attackers do not know the details of the target victim detector. Therefore, they don’t know the important configuration parameters of the detector :

1. size of the instruction window that is used to collect the features.
2. Specific features used for the classification.
3. Classification algorithm used by the target detector.

The attacker can reverse-engineer the detection period and the features used in training the target detector.

Developing Evasive Malware :

Develop systematic transformation of malware so that it can evade HMDs detection. Since, the source code is not available to us as we are work with their binaries. So they developed a methodology to dynamically insert instrucitons into the malware execution in a controllable way. They construct the Dynamic Control Flow Graph (DCFG) of the malware during execution by instruenting it through the PIN tool. Next, they add instruction into the control flow graph in a way that does not affect the program’s execution state. The process is shown in belwo figure.

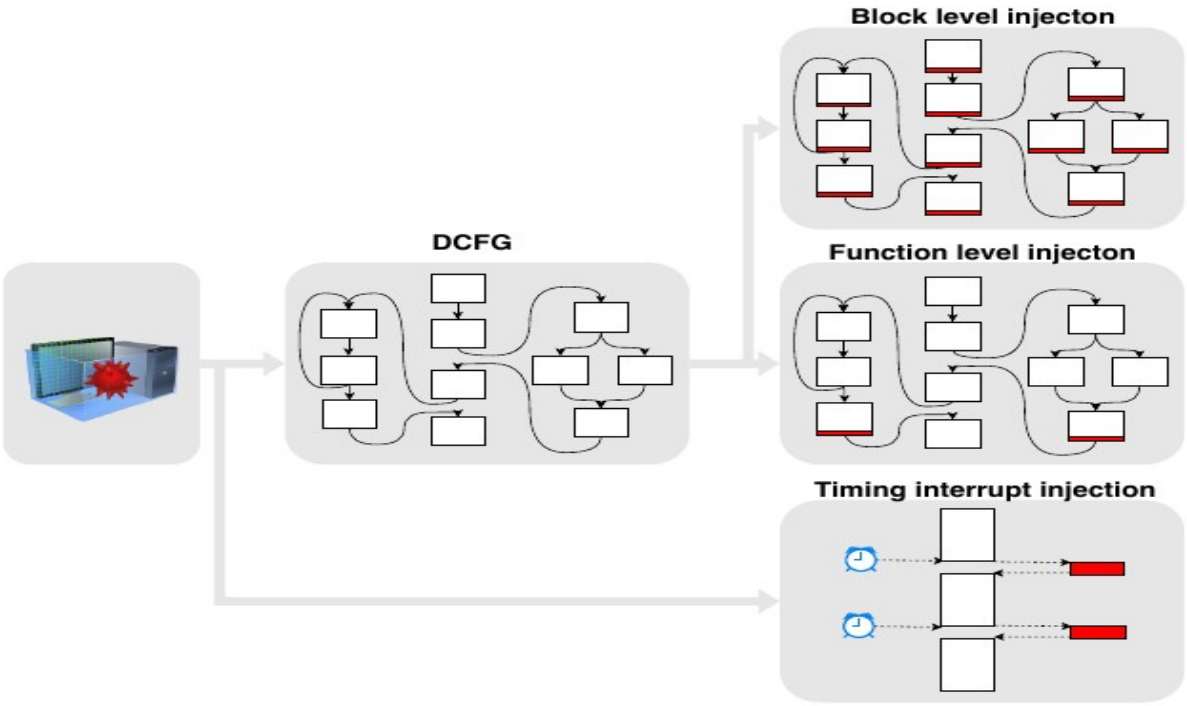

Figure 1: Methodology for generating evasive malware

The injected instructions must change the feature vector in a controlled way based on the reversed-engineered classifier to attempt to move the malware across the classification decision boundary to be classified as normal. They explore three approaches for injecting instructions :

1. Block level : insert instructions before every control flow altering instruction.
2. Function level : insert instructions before every return instruction.

3. Periodic injection : set a timer interrrupt that injects instructions at a controllable frequency.

Retraining Victim Detectors :

If we retrained the detector with the addition of evasive malware samples in the training dataset then the weights can be updated regularly to allow the detector to adapt to emerging malware. However, there is still the possibility that the retrained detector could itself be reverse-engineered and evaded again. As attacker continue to evolve, it is not clear if the detector will eventually become ineffective due to the numberof classes it is attempting to separate or converge to classification setting that is impossible to evade. After 7 generations, the detector can no longer be trained successfully as malware and normal programs became inseperable using our NN becasue the feature is not sufficiently discriminative, or NN could no longer represent the complex decision boundary between the different classes of evasive malware and normal programs.

Evasion-Resilient HMDs :

RHMDs introduce randomization to make detectors resistant to reverse-engineering. They randomize two detectors settings :

1. feature vectors used for detection.
2. Collection periods used in the detection.

Using an RHMD with two detectors, reverse-engineering the detector becomes substantially more difficult because the model now includes two diverse detectors which are selected randomly. To further increased the diversity of detector, they construct detectors with two different collection periods resulting a pool of six detectors, which are randomly choosen by the detection logic.

Implementation :

They collected malware samples from MalwareDB maware set. This set consisted of 3000 malware programs. For regular program samples, they used windows programs including acrobat reader, Notepad++ and Winrar. Total of 554 non-malware programs. The whole dataset divided into victim training (60%), attacker training (20%), attacker testing (20%). the dataset collected by running both malware and regular programs on a virtual machine with a windows 7 operating system. Dynamic traces of executed programs were collected using PIN instrumentation tool. The collected trace duration for each executed program was 5000 system calls or 15 million committed instrucitons. They collected different feature vectors,

- Executed instruction mixes (number of times each opcode executed).
- Memory address patterns
- Architectural events (unaligned memory accesses and taken branches).

Critique :
Even for 6 different detectors and switching between them makes the reverse-engineering difficult and resiliant to envasion. After reverse-engineering, it does not comment anything about the number of instruction needed to be inserted so that overall HPCs will evade the HMD process. We can also use different ML classifiers to add diversity and trained on different feature vectors. These classifiers switches based on random number generator in each detection window.

Regarding related works :

- *A taxonomy and survey of attacks against machine learning* – adversary can make changes to an image's pixels to cause the miss-classification of the image but will not change the image's visibility to the human eye.
- *Adversarial attack on microarchitectural events based malware detectors* – it proposed evasion attacks on HMD by utilizing a wrapper and executes a benign workload to hide the malicious malware activities.
- *Rhmd : Evasion-resilient hardware malware detectors* – it basically evaluated RHMDs that uses a huge number of base detectors.