Juan Anaya, Ayush Patel, Matthew Zaldaña

Dr. Hossein Sayadi

CECS 341

August 11, 2021

Analysis on "PREEMPT: PReempting Malware by Examining Embedded Processor Traces"

According to Dictionary.com, the word preempt means to "take action in order to prevent it from happening." It could also mean, however, to "acquire (something) in advance". The analysis fulfills both definitions and works in conjunction with its desired outcome to be as robust as designed. Viruses are nothing new and anti-virus software tools have been created to address the issues that malware presents to the consumer of which malware can "exploit software vulnerabilities to subvert the AVS" (Basu, Elnaggar, et al 1). PREEMPT is a technique designed to re-purpose the debug hardware component called the embedded trace buffer, or ETB, which controls and monitors the internal activities of a chip, and combines the observations with "machine learning-based classifiers to preempt Malware before it can cause damage" (Basu et al 1). As it is hardware-based, it creates a high-accuracy, low-latency, difficult to hack, performance penalty-free, robust examiner.

PREEMPT catches malware in real-time. First, it selects the trace signals from the ETB, then analyzes the circuit to remove any redundant signals, and creates a database of traces for the benign programs and malware, which are inputs for the ML classifier. Using various platforms, processors, boards, two sets of malware families downloaded from VirusTotal and four classifiers referred to as: KNN, RF, DT and NN, trace signals from the cache to the core returned almost a 98% of load return or instruction fill in transactions. This way "the classifiers distinguish malware traces from the benign by analyzing the cache-lines and the instructions" (Basu, et al 4). After this, reset values and benign traces are pre-processed. An ML-based classifier is used to distinguish between malware and benign classes, predicting the probability the new data it is fed belongs to a particular bin. From the figures, the classifiers: KNN, RF, DT and NN kept the FP under 1.5% and the TP above 90%. PREEMPT tested cross-family validations. The results, while lower, were to the authors, still satisfying: more than 78% for

a high classification accuracy. Speed was also a factor and all malicious traces were detected “within 600 cycles from their activation” (Basu, et al 5). When visualized as a scatter plot, malware traces composed a big red dot, which demonstrates the speed and high effectiveness that PREEMPT has, but also shows how it leaves out the benign traces out, shown in blue dots scattered around the entire window.

As a collaborative piece, Basu, et. al. created a strong paper that uses figures and pictures effectively to illustrate the PREEMPT experiment or describe the results. For example, the first figure gives an overview of a typical validation design overflow compared to PREEMPT. Figure 2 shows what parts in the board architecture are being used and tested in the experiment. The bar graphs and the scatter plots aided in understanding the results in the case of table 2 which outlines the speed with which the malware traces are found. Another thing that helped was how the researchers outlined each process and explained what the components used would do and what each step accomplishes. Easy to follow.

However, the paper does have some flaws. While the two families of malware tested used more than 300 variants, they are specific to the experiment and only used for a particular board and processor. Due to the lack of malware tested, disputes may come up that the results may turn out to be inaccurate to many designs that exist out there. Although the authors did use a Gafgyt family malware written in C, which is cross-compiled and applied across processor architectures (Basu, Elnagger, et al 3), the downloaded viruses may not apply to other architectures existent in present day. On another note, the conclusion speculates how periodic dumping of traces affects classification accuracy (Basu, Elnagger, et al 6). Its latency could worsen and would affect the techniques’ overhead and its speed as these processes would have to be executed prior to continuing to examine the traces.

In all, PREEMPT uses machine learning to detect malware at the hardware level with a high TP value, while keeping a low FP, re-using debug data to find these malicious traces. As a group, we believe that this method of hunting down malware comes with great benefits which far outweigh the weaknesses it may incur. While the paper lacks experimentation with other types of malware, its procedure and findings are wholesome and solid. The technique is still waiting to be built into modern processors; however, we are sure that some variation of PREEMPT will be implemented soon.