PREEMPT: PReempting Malware by Examining Embedded Processor Traces

Juan Anaya

Ayush Patel

Matthew Zaldana

Virus vs. Anti-Virus Software (AVS)

- Malware is a catch-all term for any type of malicious software designed to harm or exploit any programmable device, service or network.
- Anti-Virus Software (AVS) tools are used to detect Malware in a system
- Software-based Anti-Virus Softwares are extremely vulnerable to attacks from Malware and viruses
- A malicious entity can exploit these vulnerabilities to subvert an Anti-Virus Software

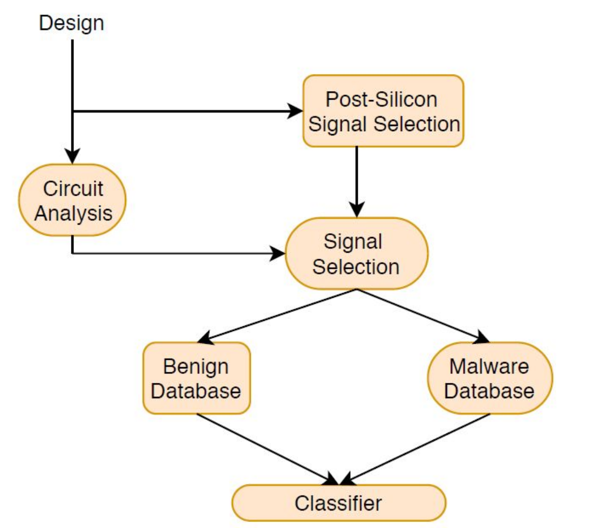
The Ugly Truth Behind Anti-Virus Softwares

- Anti-Virus Softwares have lots of software vulnerabilities.
- According to a recent study, a stealthier Malware can always be created to circumvent a top of the line Anti-Virus Software
- Most Anti-Virus Softwares use static signatures to detect Malware, however, Malware can have multiple executable formats to circumvent these signatures
- Because AVS is a software, it is slow, which results in high latency for Malware detection

HPCs - High Performance Counters

- Researchers have started using on-chip hardware components like HPCs to detect Malware in a system
- Researchers think that although an Anti-Virus Software can be circumvented by variations in Malware code, it is difficult to subvert a hardware-based detector, since the Malware function will remain the same
- Although what seemed like a good idea, HPC-based Malware detection has an unacceptably high false positive rate (FPR) of 15%

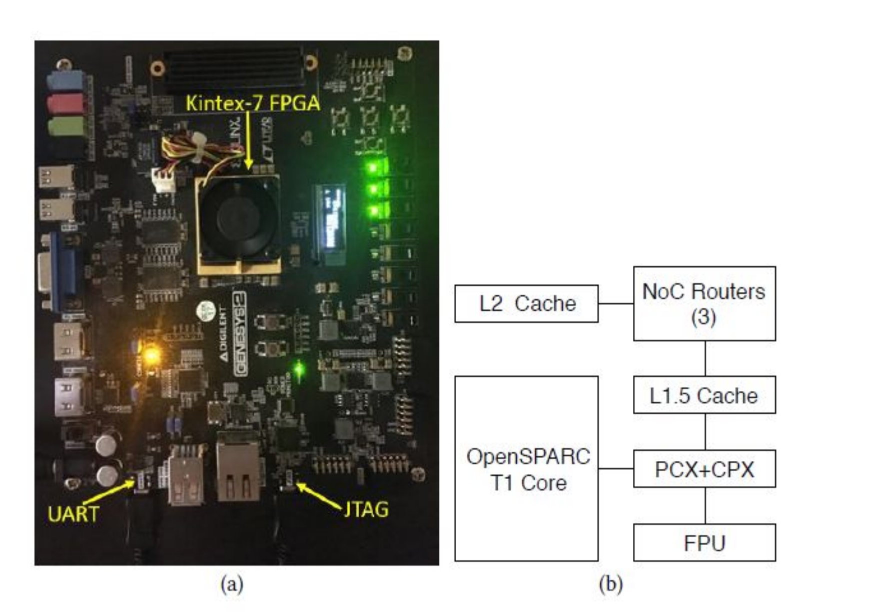
What is PREEMPT?



- PREEMPT stands for PReempting Malware by Examining Embedded Processor Traces
- Essentially, a malware bounty hunter, hardware antivirus

1. It selects the trace signals from the embedded trace buffer, which controls and monitors the internal activities of a chip AKA ETB
2. 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
3. Finally, 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

How it works - 1



(a) (b)

- The researchers downloaded 300 samples from two different types of malware families
- After running benign and malware programs on an OpenPiton platform, ETB trace signals from the cache core crossbar are analyzed
- Classifiers distinguish malware traces from the benign by analyzing the cache-lines and the instructions
- The traces are now pre-processed in two stages
  - Cleanup of reset values - registers initialized to reset states
  - Cleanup of benign traces - 2 benign traces, 1 in benign programs and 1 in benign functions within the malware; the former is removed

How it works - 2

- There are 4 classifiers that are used to test for the traces
    - KNN - K-nearest neighbor
    - RF - Random forest
    - DT - Decision Tree
    - NN - Neural Networks
- Each classifier is trained from data of both classes - benign and malware
- Then it predicts the probability that a new trace is from a particular class

Strengths of the paper

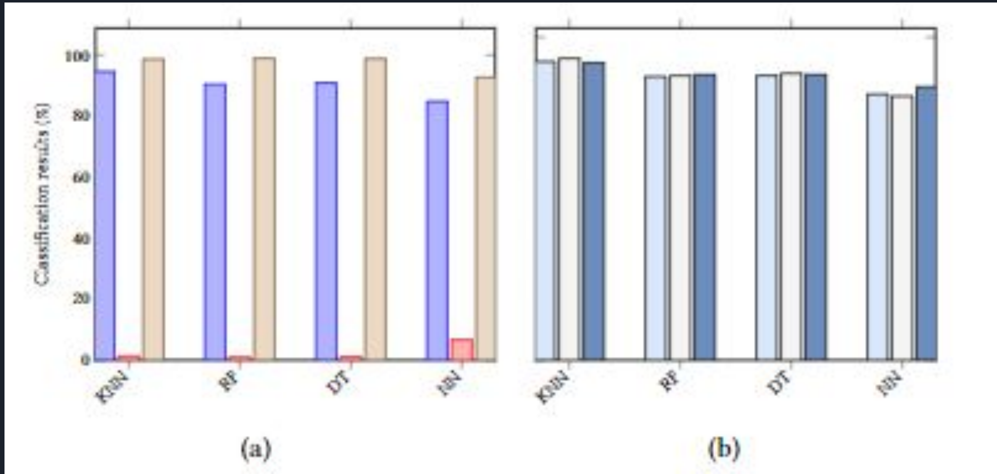
- Use of pictures and figures
  - Helps with outlining the results
  - Enhances understanding of the topic’s methodology
- Detailed paper format outline
  - Dives into inner workings of what each component will do
  - Steps through each step in the method

Weaknesses of the paper

- Only 2 families of malware
  - Lack of experimentation
  - More explanation as to why other types were not used
  - Experiment specific to board and processor
- Speculative conclusion
  - Mentioning this beforehand is helpful
  - Implies that its experiment is flawed

Classifiers Results

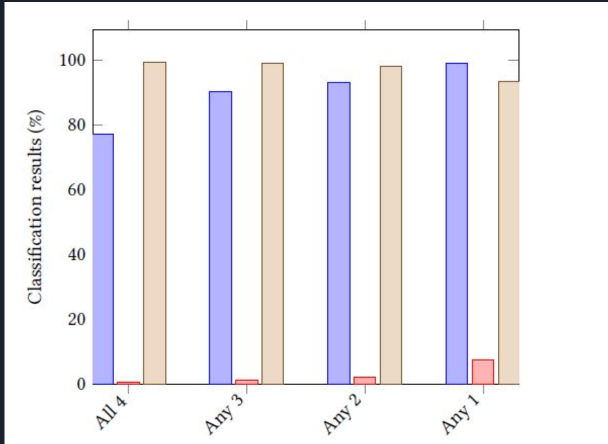
- Figure a all 4 classifiers surpass the 90% mark of finding malware traces
- Figure b reports cross-validation accuracy when tested against additional unknown malware
- KNN is the best.

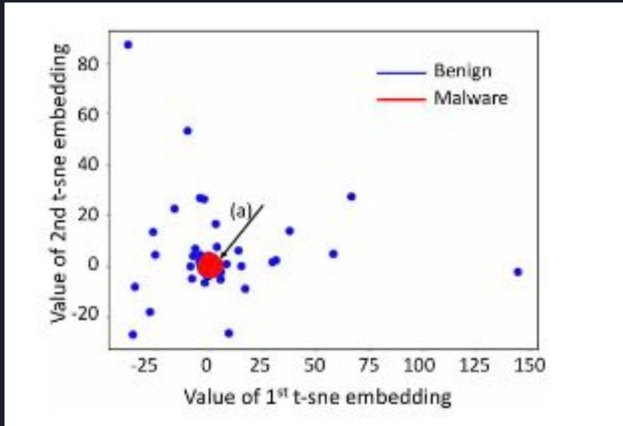


Other Results

- Factor: speed - Malicious traces were detected within 600 cycles; table
- When combined with other classifiers, TP increases, but FP worsens
- Scatter plot shows big red dot which is malware
  - Blue dots are benign programs

Malware Sample	Malicious trace (cycles)			
	KNN	RF	DT	NN
1	118	118	118	118
2	50	50	50	50
3	42	42	42	42
4	211	528	528	528
5	266	266	266	266
6	138	138	138	248
7	0	0	0	0
8	98	98	98	98
9	4	4	4	4
10	56	56	56	56
11	272	272	272	439
12	97	97	97	97
13	93	94	94	93
14	29	29	29	29
15	463	463	463	463





Conclusion

- PREEMPT uses ML 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.
- We think that the benefits outweigh the paper's weaknesses and the topic's methodology
- The technique is still waiting to be built into modern processors; however, we are sure that some variation of PREEMPT will be implemented soon.