4 Design

4.1 Design Context

4.1.1 Broader Context

The tool our team is developing is designed for those in academia researching machine learning solutions to malware attacks. It will help test, evaluate, and strengthen malware detection software to create more robust and secure systems and ultimately addresses the ever-growing need for safe and reliable information technology.

In addition, this can be used to supplement existing softwares that chip manufacturers like AMD and Intel can utilize.

Safety and Welfare

If we are not careful, our software can easily be used maliciously. Someone could use it to help them get past a system's ant-malware protection system and compromise the machine, affecting the confidentiality, integrity, and availability of the victim's data. We must keep this in mind when making design choices so the software can only be used for research purposes. Some of these design choices include not outputting the evasive attack code and only leaking designated data

4.1.2 Prior Work/Solutions

Before we started designing our project, we read a research paper, Using Power-Anomalies to Counter Evasive Micro-Architectural Attacks in Embedded Systems, published by the Department of Electrical and Computer Engineering, The University of Texas at Austin.

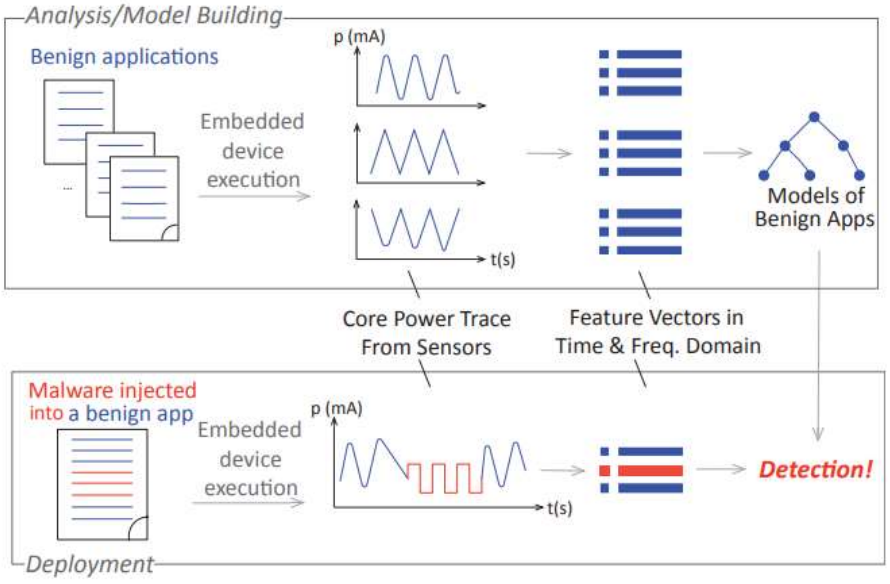


Figure 1: Overview of power-anomaly detector system[1].

This research proposed a power-anomaly detector system (Figure 1), which is very similar to the detection tool that we will create. This system provides an effective way to detect a range of microarchitectural attacks: Spectre attacks, covert-channels and row hammer attack. The researches demonstrate that power anomalies can reliably cut through the noise of diverse benign programs to detect microarchitecture attacks in complex embedded system. To model evasive attackers, they introduce evasive, power-mimicking micro-architectural attacks that replicate benign applications' power behavior while executing malicious tasks.

This power-anomaly detection can be used to defeat a particularly harmful and sophisticated class of attacks. The pro of the detector system is it is easy to deploy. Hence, it does not require expensive upgrades and can be strengthened to combat the class of attacks. But, this system is limited to certain type of attacks.

4.1.3 Technical Complexity

Since our project can be seen as a research project, it includes various scientific, mathematical, or engineering principles. A big emphasis of our project is machine learning, which is a very complex topic for all of us since we don't have any prior background, which will require us to do research to be more familiar with the topic. Another key feature of the project is the various types of micro-architectural attacks we will examine and modify. The goal of the project is to make changes to these given attacks in order for them to disguise themselves from the detection models. Our team needs to understand how these attacks exploit different weaknesses before modifying them, so we don't change the attack logic by accident when making changes to the source code. When it comes to modifying the attacks, it is essential for us to have good knowledge of C and assembly code. Since all the attacks will be written in these languages, any modification will also be required using the existing source code. Assembly language can be tricky since it varies within the architecture of the system you are working on. Since the environment we are working on requires knowledge of x86 assembly code, our team needs to learn and understand this assembly language.

4.2 Design Exploration

4.2.1 Design Decisions

List key design decisions (at least three) that you have made or will need to make in relation to your proposed solution. These can include, but are not limited to, materials, subsystems, physical components, sensors/chips/devices, physical layout, features, etc. Describe why these decisions are important to project success.

1. Implementing a GUI in python

- Choosing a specific language is important to get everyone on the same page.

- Python is the language the team has the most experience. This means less time is being spent on learning a new programming language.

2. Attack codes will all be in C

[1] Wei, Shijia & Aysu, Aydin & Orshansky, Michael & Gerstlauer, Andreas & Tiwari, Mohit. (2019). Using Power-Anomalies to Counter Evasive Micro-Architectural Attacks in Embedded Systems. 111-120. 10.1109/HST.2019.8740838.

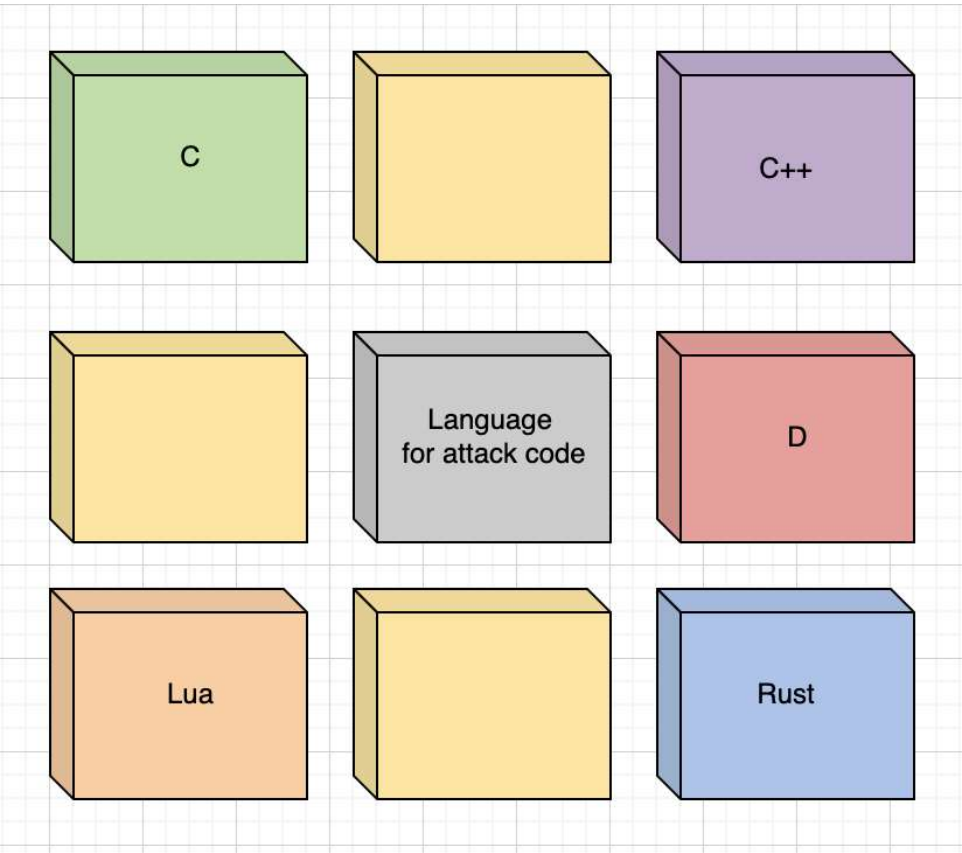
- The team was supplied template C code on certain attacks. Analyzing and building off the templates would speed up the process of development. Translating the existing code into another language would be unnecessary and take too much time.

3. All assembly instructions will be in x86

- We will be trying to attack an Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz to test our attacks. This CPU is known to use assembly x86.

4.2.2 Ideation

For at least one design decision, describe how you ideated or identified potential options (e.g., lotus blossom technique). Describe at least five options that you considered.



We ideated in this manner – lotus blossom – but slightly less complicated.

4.2.3 Decision-Making and Trade-Off

Demonstrate the process you used to identify the pros and cons or trade-offs between each of your ideated options. You may wish you include a weighted decision matrix or other relevant tool. Describe the option you chose and why you chose it.

There where 3 main factors when we decided: the language of the attack codes experience, execution speed, and help from advisor. Experience had a higher weight because the level of experience we have on a language determines the speed of development. Execution speed also plays a role in ensuring the results of the attack return quickly. This project is new territory for most of the team so being able to get help from the advisor would ensure the team has a good understanding of the project.

Below is a weighted decision matrix of the 5 potential languages for the attack code. D, Rust, and Lua scored very low because no one in the group had any experience with the languages. On top of that the advisor didn't want to use those languages for the project. C was the best choice because the entire group has worked with it in previous classes. The advisor gave a recommendation to use C and provided a lot of help getting started on the project.

Critiria	Weight	C		C++		D		Rust		Lua	
		Score	Total	Score	Total	Score	Total	Score	Total	Score	Total
Experience	0.5	9	4.5	6	3	0	0	2	1	0	0
Execution Speed	0.2	7	1.4	8	1.6	8	1.6	9	1.8	8	1.6
Help from advisor	0.3	10	3	8	2.4	0	0	0	0	0	0
Total	1		8.9		7		1.6		2.8		1.6