2. Project Plan

2.1 Project Management/Tracking Procedures

Our teams plan to adopt a hybrid development process to manage our project. Since some tasks depend on one another, it is necessary to follow the waterfall process to meet the requirements of these tasks. Our project has a unique feature in that each task must be done for each one of the attacks we are modifying. This feature allows us to work in parallel by assigning an attack to each member. Doing so allows us to work simultaneously on the same tasks for each attack. By doing this, we follow an agile process that would allow our team to tackle various tasks at the same time, which will save us time.

The team will use GitHub for version control and Notion to keep track of assigned tasks and due dates. We have a Kanban board on Notion and will follow the Gantt chart to ensure we're on track.

2.2 Task Decomposition

First Semester:

1. **Task 1:** Become familiar with attack codes and test system
    a. **Task 1A:** Understand the necessary Ubuntu scripts and test systems configuration
    b. **Task 1B:** Collect power measurements/model power signatures
    c. **Task 1C:** Modify attack codes to extract data leak rate & collect detection accuracy
    d. **Task 1D:** Understand each attack's atomic instructions & possible areas to insert code
2. **Task 2:** Implement UI with basic functionality
    a. **Task 2A:** Set up python environment and UI foundation
    b. **Task 2B:** Create necessary functions
    c. **Task 2C:** Implement command line interface
    d. **Task 2D:** Design a graphical user interface
    e. **Task 2E:** Test UI

Second Semester:

1. **Task 3:** Analyze power signatures, instructions' power consumption, and evasive attacks
    a. **Task 3A:** Analyze differences in malicious and benign power signatures
    b. **Task 3B:** Attempt to mimic benign power signatures by inserting instructions
    c. **Task 3C:** Profile and record x86 instruction's effects on power consumption

2. Task 4: Implement basic instruction insertion and attack logic
    a. **Task 4A:** Develop instruction insertion structure with instruction's power signature dataset and code insertion functions
    b. **Task 4B:** Develop attack structure with ammeter synchronization, code execution, and attack analysis functions.
    c. **Task 4C:** Test functionality
3. Task 5: Leverage NLP and CNN AI techniques to create adversarial examples (2 months)
    a. **Task 5A:** Develop ML models
    b. **Task 5B:** Implement model into instruction insertion logic
4. Task 6: Finalize project
    a. **Task 6A:** Add additional functions to the GUI
    b. **Task 6B:** Final testing
    c. **Task 6C:** Fix any minor issues
    d. **Task 6D:** Wrap-up Documentation

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestone 1: Understand the attacks

The team should be able to collect the power measurements of the model. Understand where instructions can be added/removed from the attack code to reduce detection. Be able to calculate the leakage rate and detection accuracy.

Milestone 2: Completed UI

The UI should allow the user to upload an attack code, a detection model, and a data set. The user should be able to select the type of attack as well as run and end the attack. Once the attack is executed, the UI should display the detection rate, the average leak rate (bytes/sec), and the total bytes leaked.

Milestone 3: Complete 3 different attack codes

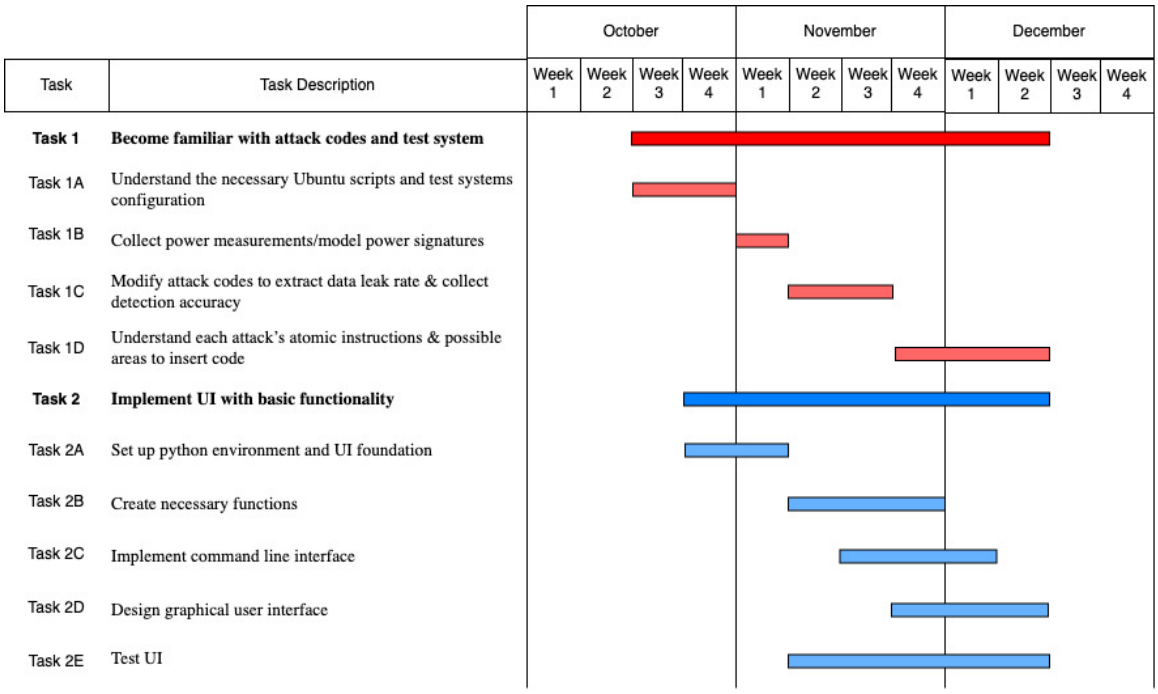
The team should have a completed specter, row hammer, and port smash attack. The ML model confidence rate should be below 20% after including instructions in each attack.

Milestone 4: Deliver the project to client

The GUI is updated with any changes that were made throughout development. Any final modifications of the attack codes should be done.

2.4 Project Timeline/Schedule

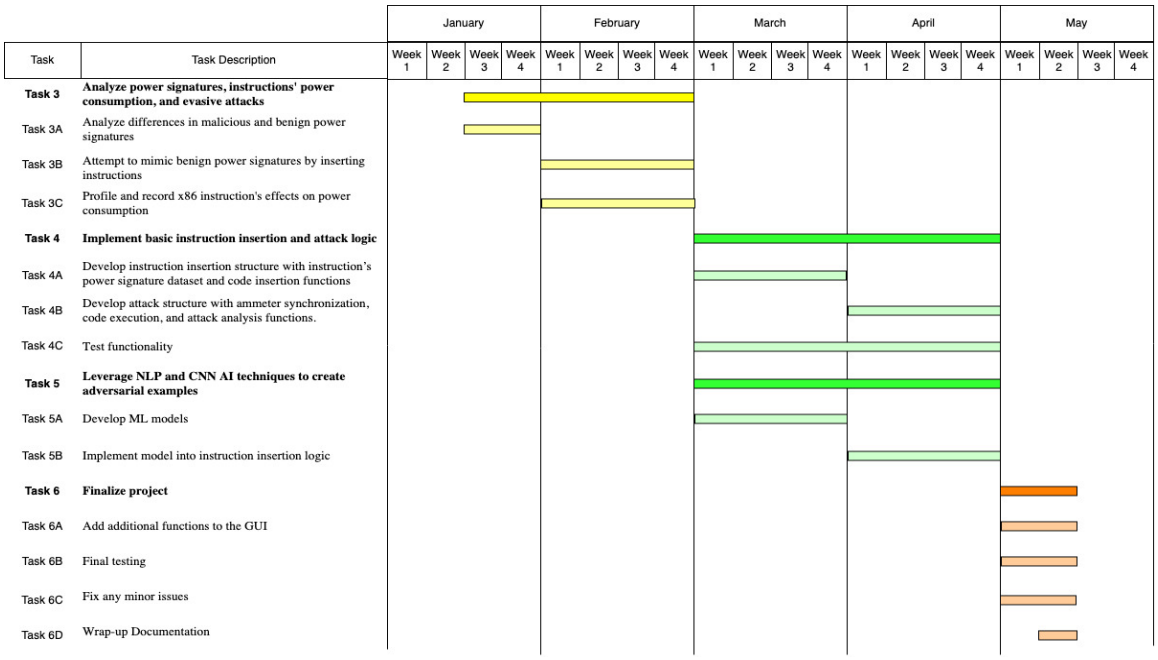
Fall 2022 Schedule



Explanation:

Tasks 1 and 2 were assigned to take the remainder of the semester to complete. This is because having a strong foundation of the attack codes is important to understand before developing them. Having the UI ready will make testing faster in the later tasks. These two tasks can be worked on at the same time because knowledge of the attack is not necessary to complete the UI. Task 1 has few overlaps since all of them build on each other. Task 2 allows for more overlap since the subtasks can be worked on simultaneously.

Spring 2023 Schedule



Explanation:

Task 3 is an important place to start the second semester to understand the power signatures of the model fully. First, it is necessary to analyze the benign and malicious power signatures. 3B and 3C can be done simultaneously after testing and analyzing the additional instructions that affect the power signatures. Task 4 and 5 is where the attack codes start development. The team will be split to allow both to be done at the same time. Task 6 is where the team finishes up the project. Here the team will allocate resources to complete the project at the same time.

2.5 Risks and Risk Management / Mitigation

Low Risk: Team Members are unable to finish work by assigned deadline.

- Probability of occurring: high(70%)
- Mitigation: Clear communication with team members so that duties can be transferred and delegated, if necessary, to finish work by deadline.

Moderate Risk: An attack code has a detection rate higher than 20%

- Probability of occurring: moderate (50%)
- Mitigation: allocate more time into analyzing the benign application's power consumption.

Moderate Risk: Leakage rate may be difficult to calculate

- Probability of occurring: moderate (40%)

- Mitigation: Spend more time researching the attack code.

High Risk: Two people run an attack at the same time (will cause incorrect data for both attacks)

- Probability of occurring: low (15%)
- Mitigation: Communicate when one starts and ends an attack.

High Risk: Row hammer attack may crash the system resulting in needing a new ram.

- Probability of occurring: low (5%)
- Mitigation: Have a good understanding of the attack and carefully implementing it.

High Risk: Inability to analyze and understand power samples from ammeter

- Probability of occurring: low (5%)
- Mitigation: Ask for assistance if unable to understand the power consumption graphs

2.6 Personnel Effort Requirements

Tasks	Total number of person-hours	Justification
Task 1: Become familiar with attack codes and test system	150 hours	Task 1A: Understand the necessary Ubuntu scripts and test systems configuration (All members, 6 hours each) Task 1B: Collect power measurements / model power signatures (All members, 3 hours each) Task 1C: Modify attack codes to extract data leak rate & collect detection accuracy (All members, 8 hours each) Task 1D: Understand each attack's atomic instructions & possible areas to insert code (All members, 8 hours each)
Task 2: Implement UI with basic functionality	120 hours	Task 2A: Set up python environment and UI foundation (three members, 2 hours each) Task 2B: Create necessary functions (three members, 18 hours each) Task 2C: Implement command line interface (three members, 9 hours each) Task 2D: Design a graphical user interface (three members, 8 hours each) Task 2E: Test UI (three members, 3 hours each)
Task 3: Analyze power signatures, instructions' power consumption, and evasive attacks	240 hours	Task 3A: Analyze differences in malicious and benign power signatures (All members, 8 hours each) Task 3B: Attempt to mimic benign power signatures by inserting instructions (All members, 16 hours each) Task 3C: Profile and record x86 instruction's effects on power consumption (All members, 16 hours each)

Task 4: Implement basic instruction insertion and attack logic	168 hours	Task 4A: Develop instruction insertion structure with instruction's power signature dataset and code insertion functions (three members, 22 hours each) Task 4B: Develop attack structure with ammeter synchronization, code execution, and attack analysis functions (three members, 25 hours each) Task 4C: Test functionality (three members, 9 hours each)
Task 5: Leverage NLP and CNN AI techniques to create adversarial examples	168 hours	Task 5A: Develop ML models (three members, 30 hours) Task 5B: Implement model into instruction insertion logic (three members, 26 hours each)
Task 6: Finalize project	60 hours	Task 6A: Add additional functions to the GUI Task 6B: Final testing (all members, 4 hours each) Task 6C: Fix any minor issues (all members, 3 hours each) Task 6D: Wrap-up Documentation (all members, 3 hours each)
Total	906 hours	Finish

2.7 Other Resource Requirements

The project will require specialized hardware to pull the necessary CPU power consumption data and achieve the performance needed to run the AI-based microarchitecture attack detector. Our team will be provided and required to use the following experimental setup:

- Intel Comet Lake Microarchitecture
    - CPU Model: Intel(R) Core (TM) i7-10610U CPU @ 1.80GHz
    - OS: Ubuntu 20.04 LTS
    - Linux Kernel: 5.11.0-46-generic
- Server Information

- Nvidia GeForce RTX 3090 GPU
- CPU Model: Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz