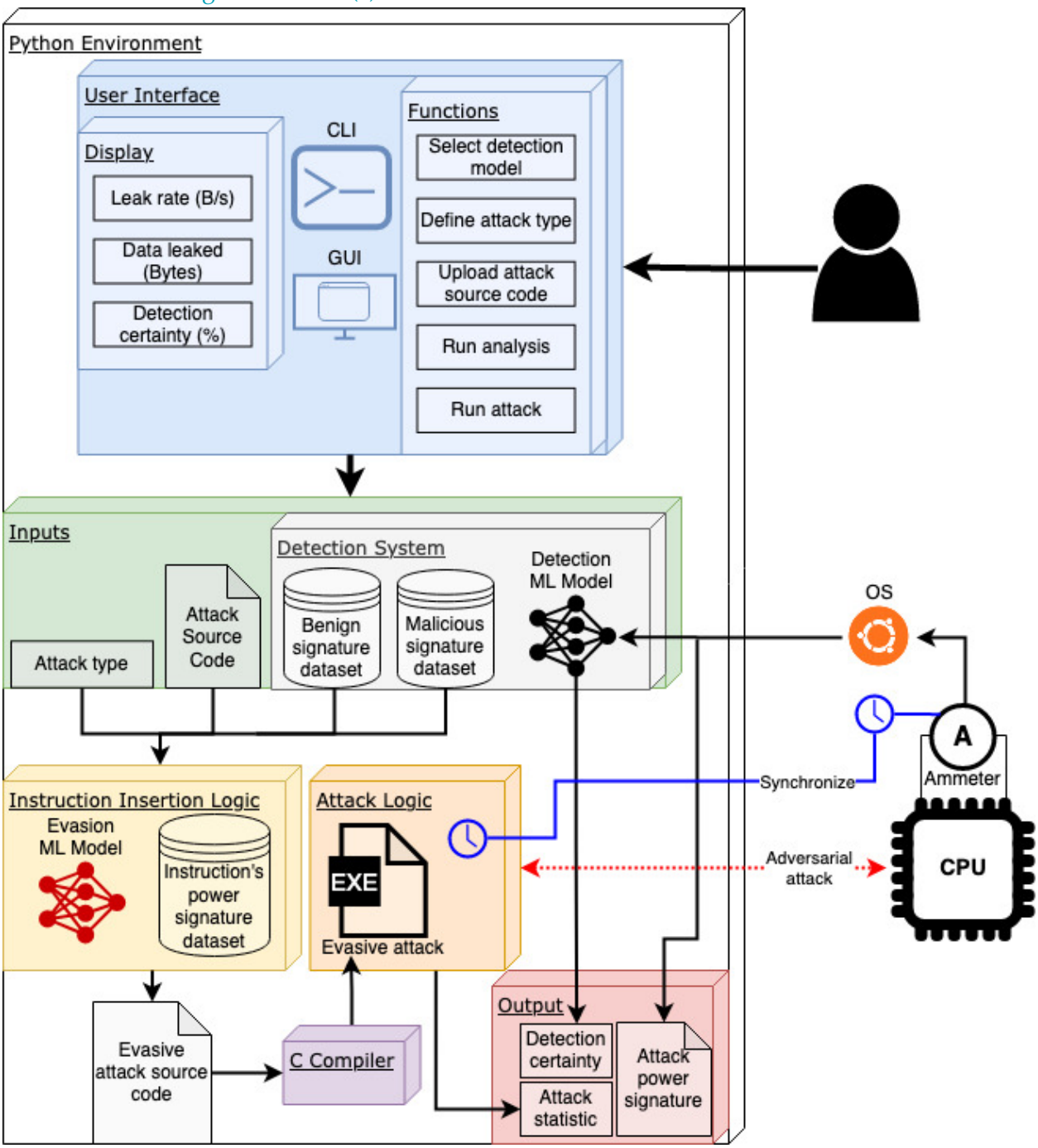
Proposed Design

4.3.1 Overview

Our software will offer a graphical user interface (GUI) or command line interface (CLI) for the user to interact with the application. Navigating the interface, the user can upload attack codes and select a detection model and attack type, which is fed into the instruction insertion algorithm. The algorithm then generates an evasive attack that exploits the detection system's underlying machine learning (ML) model. Running the attack with the program will generate statistics about the attack.

4.3.2 Detailed Design and Visual(s)



4.3.2.1 Overview

Our system primarily lives in a python environment. Currently, there exists a machine learning model that can differentiate between malignant and benign code that is available. Our job is to find ways to lower the model's detection certainty to below 20% - if possible.

Users are provided with a GUI and CLI to access the functionality of the software that we will be writing. Users can upload attack source code by navigating the interface and selecting the attack type and

detection model. This is fed into the instruction insertion logic, which generates the attack code, but is modified to attempt to avoid the selected machine learning model's detection. Running that attack with the program would create statistics about the attack, including data leaked, certainty, and leakage rate.

4.3.2.2 User Interface

We provide clients with a GUI as well as a CLI, both with the same functionality. Clients can interact with our application through it. They can input models and attack types/codes in order to receive, and output, which is displayed after the machine learning model analyzes it. This is outputted to the console/GUI with detection certainty, data leaked, and leakage rate.

4.3.2.3 Inputs

The inputs that are available for the client to put in consist of the attack type, attack code, and detection model. The attack source codes will either be C files or x86 assembly.

4.3.2.4 Instruction Insertion Logic

All of the inputs are fed into another machine learning model that is compared with the power reading dataset. This model generates attack source code that is meant to be subserve the selected mode. It then compiles the code and sends this into the attack logic block.

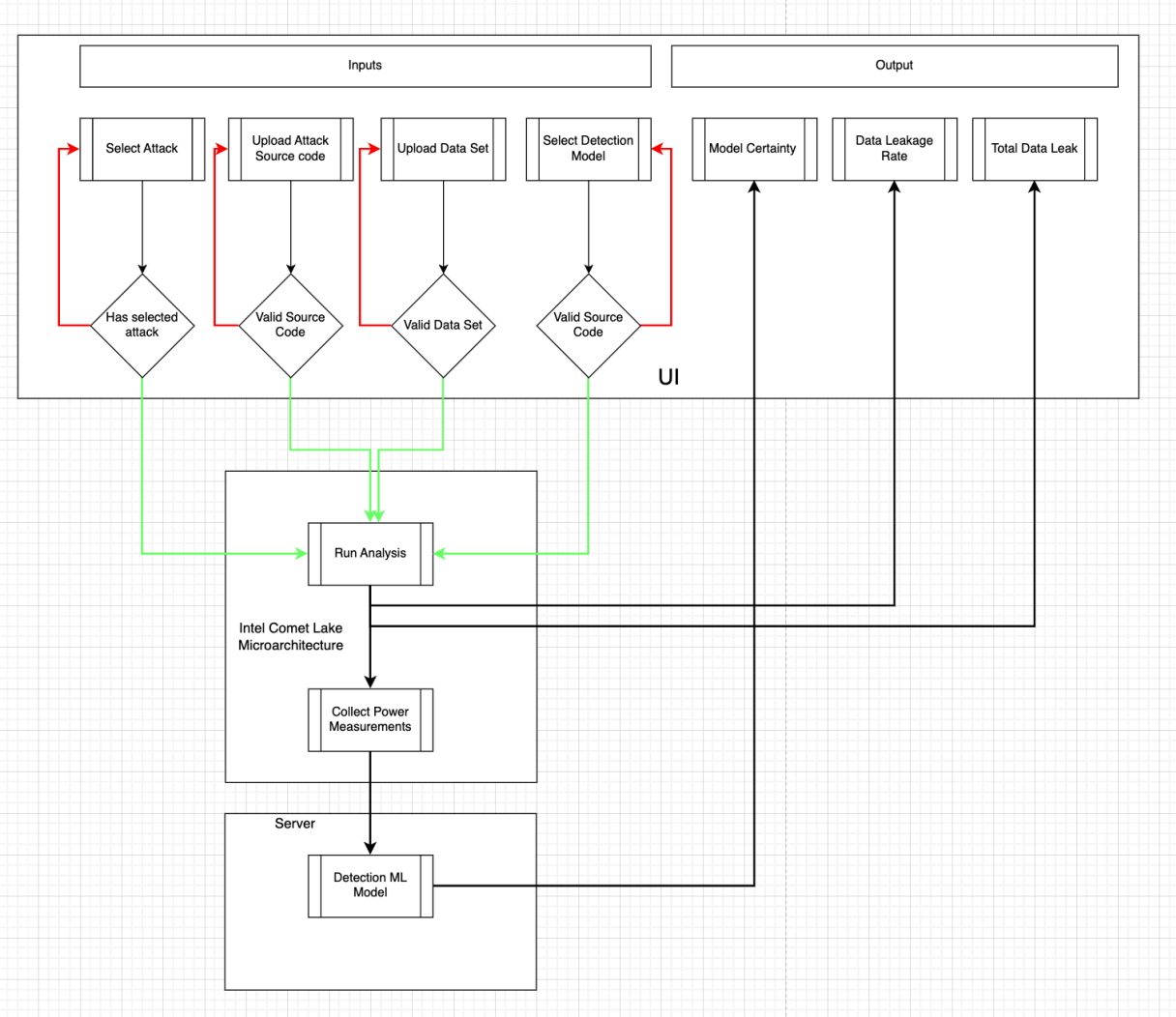
4.3.2.5 Attack Logic

After running the code, it determines whether the code was malware and outputs its certainty percentage and various statistics. This is done by reading the power usage of the computer as the attack is running and looking for any abnormalities that has already been quantified in the machine learning model.

4.3.2.6 Outputs

All of this info will be outputted on the CLI or GUI for the client to view.

4.3.3 Functionality



Description: Our design consists of using the UI as way for the user to interact with the system as well as to receive information. As show in the image above the user has the ability to select add unique inputs to the system that fits their needs. These inputs are selecting which what attack the trained model is looking for, the source code of the attack the user is testing, the data set used, and lastly selecting which trained model the user wants to test the attack on. Once the user has inputted all these paraments, then can go ahead run the analysis. The microarchitecture will the run the code and output the rate at which the attack leaked the data as well as the total amount of data that was leaked. This information will send to the UI where the user can those results. After the code was run, the system will then go ahead a collect the power measurements generated by the uploaded attack on the microarchitecture. These measurements will be sent to the server along with the data set of power measurements generated by the selected attack type where the detection model will run an analysis and output a certainty level on the match between the data set given by the user and the uploaded attack power measurements collected. If the power measurement matches the power measurements of the selected attack it would result in a higher certainty. This result would also be sent to the UI where the user can see the result of the analysis.

4.3.4 Areas of Concern and Development

Based on our design, the users are required to upload the attack source code, select a detection model and define the attack type to run the analysis. The result must meet the project requirements, especially the detection certainty and leakage rates.

One of the concerns is the tool might not be able to achieve the desired detection rate (20%). We will need to use different methods to optimize the ML model and allocate more time to analyzing the benign application's power consumption.

Besides that, the leakage rate may be difficult to calculate. We will need assistance from TA to help us understand better how to calculate the leakage rate and spend more time researching the attack code.

We are also concerned about users might upload a wrong attack source file to cause the tool to terminate or freeze. We will add some functions/error handlers to make the tool display an error message on the GUI.

4.4 Technology Considerations

Python:

- Strengths:
    - Simplicity makes it easy to use for AI/ML applications.
    - Platform independent and versatile.
    - Good library ecosystem.
- Weaknesses:
    - Slow execution due to Python being an interpreted language.
    - High memory consumption.
- Alternatives:
    - Scala, C/C++

Attack Code in C:

- Strengths:
    - Extremely fast execution and compilation
    - Low level language, making it easy to program machine level hardware.
    - Well used language in the team.
- Weaknesses:
    - No run time checking – all errors handled after writing the program.

    - No exception handling.
    - Manual memory management,
- Alternatives:
    - Rust

Intel x86 Assembly

- Strengths:
    - One of the most popular ISA (instruction set architecture) today
    - Targets the devices that the attack source codes are meant for
- Weaknesses:
    - Steep learning curve
- Alternatives:
    - RISC V, ARM

4.5 Design Analysis

No components have been implemented so far. The past couple weeks have been spent with our advisor making a design and fully understanding the problem that has been presented to us.

We have recently gained SSH access to the laptop that we will be using for the project this last week. With that, we will be able to familiarize ourselves with the microarchitecture environment (collecting power measurement data, ssh-ing, current attacks).

Our current plans are to now start analyzing the power measurements of the different attack codes that need to be implemented.

As our team just gained access to the intel comet microarchitecture last week. We haven't been able to implement any components. While the team waited, we drew up the overall design of the project. Now we have been getting familiar with the microarchitecture environment such as collecting the power measurement. The team was also able to start analyzing the different attack codes we would implement.