2 Project Plan

2.1 Task Decomposition

- Find/ write code that can collect the necessary data. This data includes but is not limited to CPU L3 cache size. This information will be used to pinpoint the internet traffic.
- Gain access to our user space on the server. This user space will allow us to test things in a closed environment.
- Start creating lambda functions locally on the server. These functions will used to gather information.
- Test these lambda functions locally to ensure they are working properly. We will test locally to insure that our functions are working properly.
- Gain access to our Firecracker environment. This will be a closed environment and will provide us with more information about security flaws in serverless functions
- Start gathering data about which languages allow the most vulnerabilities to be exploited.
- Write a research paper that accurately describes our findings and can be easily understood by anyone with a serverless background.

2.2 Project Management/Tracking Procedures

For our project we will adopt a waterfall+agile management style. We will use the agile management style to quickly complete any small steps that need to be completed before we can complete larger parts of the project. We will use the waterfall approach to complete bigger parts of the project and set goals and dates to have certain aspects of the project completed. For example we will use the agile approach to complete steps like rewriting attack code and the waterfall approach to complete bigger steps such as getting our server set up.

Our group will use Gitlab, Discord and Microsoft Teams to track progress throughout this semester and next. We will be using the features available in Gitlab to place new and completed tasks as well as assigning team members to tasks.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

We will utilize the sprint system in agile to break out milestones down into two-week long goals. Each sprint can be further broken down into easily-divisible parts or tasks that are simple to understand, have as few dependencies as possible, and are simple to verify for correctness. Milestones we are looking at include: Working server with all required access and technology set up for development, All attack codes written and verified for testing, Attack codes run on server and dataset created based off results, Machine learning model understands dataset and can make conclusions on new datasets, and Our machine learning model recognises running functions based on timing with a fifty percent accuracy.

2.4 Project Timeline/Schedule

	Mid-Oct.	End of Oct.	Mid-Nov.	End of Nov.	Mid-Dec.	End of Dec.	Mid-Jan.	End of Jan.	Mid-Feb.	End of Feb.	Mid-Mar.	End of Mar.	Mid-Apr.	End of Apr.
Server fully set up	🟩													
Attack codes written and verified	🟨	🟨	🟨	🟨	🟨									
Attack code dataset created			🟧	🟧	🟧									
Machine learning model trained					🟦	🟦	🟦	🟦	🟦					
Machine learning model accuracy raised									🟪	🟪	🟪	🟪	🟪	🟪

Subtasks by milestone:

- Server fully set up
  - Firecracker verified to be properly installed on server via serverless function test
  - Member permissions confirmed correct
- Attack codes written and verified
  - Attack codes in variety of languages: minimum one per team member from pool of Node.js, Python, Java, Go, Ruby, .NET, others
  - Attack codes functional on our computers and on server
  - Data being collected in reasonable shared format
- Attack code dataset created
  - Functions chosen to run on Firecracker while attack codes are running
  - Determination of which attack codes are most accurate
  - Data collection automated
- Machine learning model trained
  - Type of model, parameters, and hyperparameters determined
  - Models tested on new data
- Machine learning model accuracy raised
  - Iterative process of improving model performance
  - Aiming for >50% chance of identifying function being run

2.5 Risks And Risk Management/Mitigation

Working on a project that is doing a significant amount of research, means that our estimates for the length of required times are just that estimates. As such, any of the items that we need to accomplish for this project could have a requirement time that could dramatically underestimate that actual required time, after moving from purely theoretical, to actual implementation. As this risk is very likely, we have some pre-decided mitigating factors in place including have multiple alternative paths for all possible deliverables. An example of this is each group member starting with a different programming language supported by serverless functions to find the most useable language. Alongside this, as we are using a sprint based planning strategy, so as inevitable issues arise, we can budget extra time, by adding items to our backlog.

Alongside this, in the case our project produces findings that confirm serverless functions can leak data, many steps need to be taken to avoid risk. This would mean we would need to deliver server-providing companies such as AWS with relevant information and our findings. If we do not collect data in a thorough way and if we do not record our process with proper due diligence, these companies might not heed our advice. If we don't record every step of the process, and companies don't take our findings seriously because of it, users will be at risk to the vulnerability.

2.6 Personnel Effort Requirements

For personal effort requirements, we expect each team member to put in good personal effort that will get the needed work done. A good guideline for this would be using the Iowa State University provided estimate of 10 hours outside of class per week, for a 3 credit course. Accounting for the 1.5 hours we spend each week meeting, that give each member an estimated time of 8.5 hours of possible work per week. For our 5 team members, multiplied with the useable weeks each semester, that gives us a maximum of 1100 total person hours. Accounting for general loss, as each member being full time students, will be unable to maintain full momentum for all sprints, we can reasonably budget 1000 total person hours for the whole project.

Weekly Time Budgeting Table

Items Needing Done	Estimated Time	Total Time for Item	Combined Total Time
General Documentation	2 total hours/week every week	60 total hours	60 total hours
Learning basics of new programming languages	5.5 hours/week per person for the 3 weeks	83 total hours	143 total hours
Design an implementation of attack code in each language	6 hours/week per person for 2 weeks	60 total hours	203 total hours
Testing each set of attack codes	7 hours/week per person for 2 weeks	70 total hours	273 total hours
Using Attack Code to collect and quantify dataset	7 hours/week for 5 weeks	175 total hours	448 total hours
Learning tools to build a machine learning model on the dataset	5 hours/week for 2 weeks	50 total hours	498 total hours
Training the model on the collected dataset	6 hours/week for 4 weeks	120 total hours	618 total hours
Verifying the model properly quantifies new data	6 hours/week for 2 weeks	60 total hours	678 total hours
Refining the model for further accuracy	7 hours/week for 5 weeks	175 total hours	853 total hours
Compilation to prepare to for final deliverables	7 hours/week per person for 2 weeks	70 total hours	923 total hours
General Overhead	.5 hours/week per person every week	75 total hours	998 total hours

2.7 Other Resource Requirements

Some other resources that we need as a group to complete this project is support from Dr. Gulmezoglu's PhD student who is also working with serverless functions, and is helping maintain the server that we will be using to conduct our testing.