VECTRA®

Team Vectra AI

Michigan State University

Hybrid Cyberattack Simulator

Project Plan

Spring 2024

Project Sponsors
Brad Woodberg

Team Members
Henry Barton
Alisha Brenholt
Nathan Motzny
Campbell Robertson
Andrew Talbott

Table of Contents

Executive Summary

Just as technology has been on an exponential incline since the turn of the millennium, cyber-attacks have grown in maturity and severity at a similar rate. Seemingly every week, malicious actors compromise top companies, leaving the companies with two options: leak their customers' private and protected data or pay hundreds of thousands of dollars in ransom to keep the information private. To combat these attacks, a company in San Jose, California is revolutionizing cybersecurity. While generative AI has only made headlines the past year or so, Vectra AI has been utilizing the technology since 2011. Vectra offers automated cyber-attack detection, investigation, and response solutions to companies worldwide.

More specifically, Vectra's Attack Signal Intelligence platform uses AI to analyze the behavior of attackers and prioritize these threats in real time. By utilizing machine learning to detect patterns, the platform sees past alert noise and highlights truly malicious threats. Vectra does not stop at protecting their customers' own premise network. Modern organizations have third party services (i.e. SaaS, PaaS, IaaS) that offer little to no visibility into attacks that span different technologies and providers. With Vectra's Attack Signal Intelligence, customers gain visibility into these traditional blindspots.

In the Fall 2023 semester, Vectra AI partnered with MSU Capstone to produce a Command and Control (C2) Simulator. This tool enables Vectra to design command and control channels to test and generate training samples for their revolutionary AI models. This semester's Team Vectra AI will be taking the tool to new heights by adding more sophisticated capabilities. The team is responsible for advancing the C2 framework, integrating hybrid attack support, as well as including capability enhancements. The C2 techniques involve leveraging relays/proxies, advanced internet protocols, APIs, and Webshells. The team will incorporate tools that generate multi-vector hybrid attacks which can trigger on traditional networks as well as Cloud and Identity infrastructures as well. Capability enhancements include UI improvements to improve platform robustness.

Through these improvements, Vectra AI will generate more sophisticated data to feed to their AI models to allow for a wider variety of detections, ultimately resulting in a better product and a safer world altogether.

Functional Specifications

Vectra AI creates and provides modular AI-based cyberattack detection and prevention services to organizations and businesses around the globe. For their services to be effective, Vectra's AI and ML systems must be trained on an extensive database of up-to-date and relevant cyberattack traffic. Without proper ML training over time, Vectra's detection software would be made doubly ineffective: simultaneously letting real cyberattack traffic pass without issue and flagging legitimate traffic incorrectly. Cybersecurity Analysts would have to investigate non-issues while allowing real threats to slip through. As a result, Vectra makes it a priority to continually maintain and update their detection systems to be on the cutting-edge of the cybersecurity field.

This project will assist Vectra in this goal by implementing several more networking protocols and methods of intrusion, quality-of-life improvements on the Web UI to speed up the simulation configuration process, and new simulation profile options within the existing project code to allow for more dynamic simulation data.

By using our improved C2 Simulation project, Vectra AI will be able to replicate the deployment of considerably more diverse attacks, including hybrid attacks using cloud and infrastructure services like those demonstrated by the MAAD Attack Framework and DeRF against Microsoft Office 365, Azure AD, AWS, and Google Cloud Projects.

Design Specifications

Overview

The existing design for this project includes four main components – the client, server, web UI, and MySQL database. Currently, the web UI is used as a user-friendly way to generate job lists which are then sent to the server, which then waits for a client to connect. Once a client is connected, the client opens a tunnel with and begins beaconing the server to indicate it should run tasks. As the server is running tasks, it is also running a packet capture which logs all traffic to the database. When jobs are completed, information about them may be viewed in the web UI, including the job list and a graph of the logged network traffic. Being a simulation of network traffic, all these components are run locally, and network latency will be simulated using existing libraries.

Advanced C2

Currently, the simulator can only simulate a basic type of C2 server which utilizes beaconing to indicate to the server when tasks should be run. These beacons are sent over the network, so it is entirely possible that any AI's trained with this data may become biased or overfit for this type of traffic. To help mitigate this, we will implement an option to run the C2 server using a beaconless method of communication. This will provide essential data by simulating real-life scenarios in which attackers employ a more detection sensitive approach.
Additionally, we will be implementing the option to simulate Webshell attacks. Webshell and C2 attacks are similar because they both aim to gain control over a compromised system. However, a Webshell attack operates differently regarding its communication channel. For a Webshell attack, a compromised server is often the initiator of the communication to a client; the server establishes connection and runs a script on the client. Opposingly, the primary communication for a C2 attack is a compromised system pinging to a command server for jobs. The additional option to simulate Webshell attacks, as a result, will generate a different traffic profile. Expanding the traffic profiles with the inclusion of beaconless and Webshell attacks, overall, should help to alleviate concerns regarding data reliability.

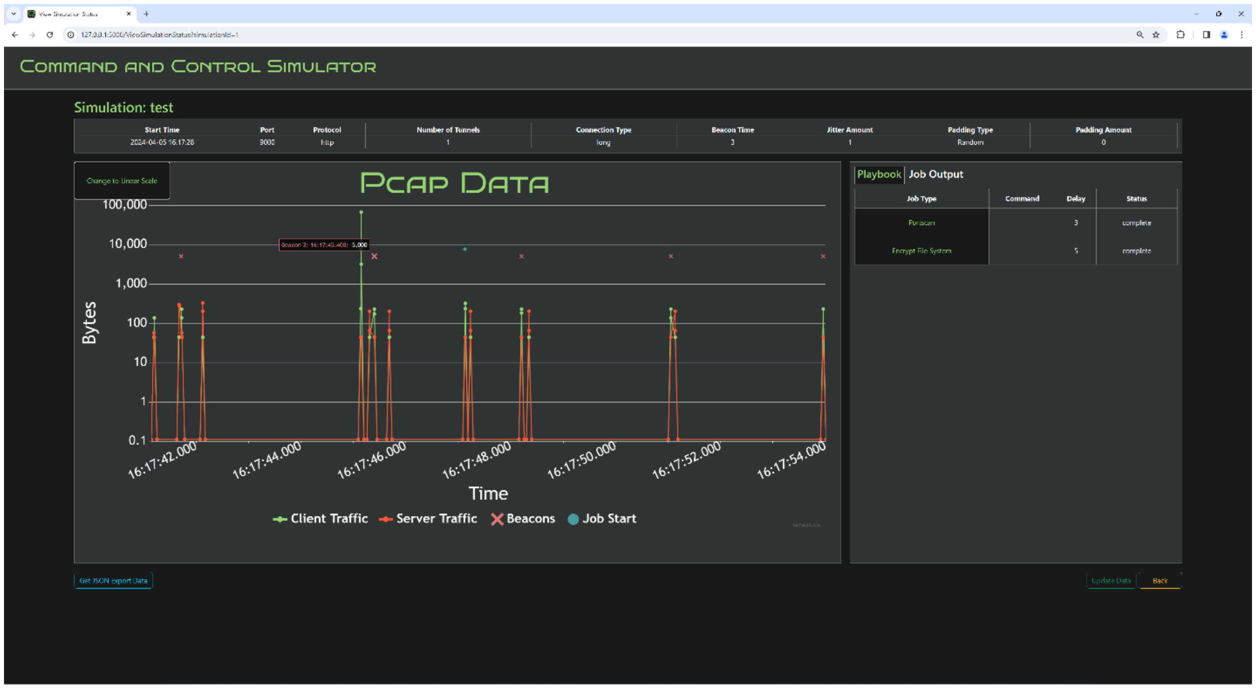
Hybrid Attacks

The level of sophistication behind cybersecurity attacks has only been increasing with time. This raises the concern that the relatively simple jobs currently being simulated could fail to generate realistic data for Vectra AI. To address this concern, we will be integrating the MAAD-AF (Microsoft 365 & Azure AD Attack Framework) and the DeRF (Detection Replay Framework) software packages. These frameworks both utilize simulations to evaluate and train their detection strategies for cloud-based servers undergoing hybrid attacks. The ability to quickly and

correctly detect attacks within a cloud-based server is of high importance; likewise, it would prove significant to have data from these frameworks implemented into our simulator.
The implementation of these frameworks is essential for the creation of realistic data that will train AI on cybersecurity attacks. Both software packages will provide our simulator with the capability to record data on hybrid attacks against cloud-based servers. Specifically, MAAD-AF is a cloud-based attack framework for testing the security of Microsoft 365 & Azure AD (now Entra ID). Accordingly, DeRF is a cloud-run application that tests the security of AWS (Amazon Web Services) and GCP (Google Cloud Platform) with its own attack techniques. Overall, both hybrid attack simulation frameworks will provide immensely helpful data for training Vectra's AI and helping to detect future attacks.

Capability Enhancements

Capability enhancements mainly refer to user interface improvements. We will be enhancing various aspects of the simulation overview webpage to improve the browsing experience.



Simulation: test

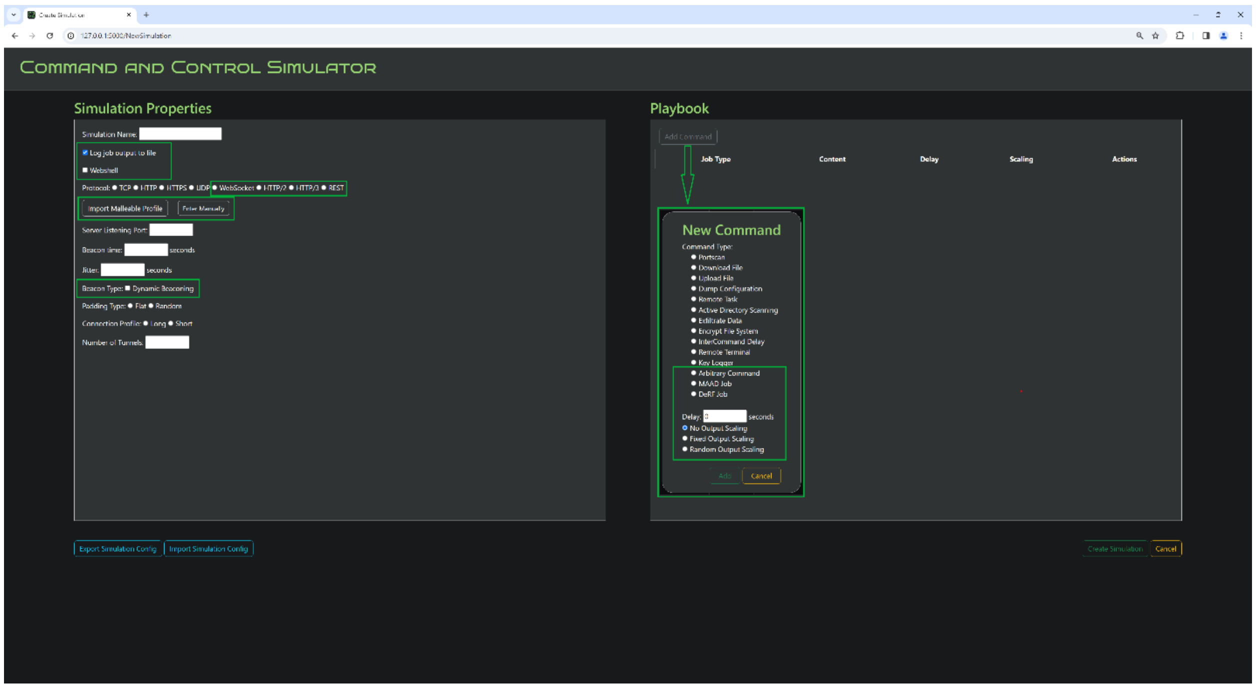
Start Time	Port	Protocol	Number of Tunnels	Connection Type	Beacon Time	Jitter Amount	Padding Type	Padding Amount
2024-04-05 16:17:28	9000	http	1	long	3	1	Random	0

Playbook | Job Output

Job Type	Command	Delay	Status
Portscan		3	complete
Encrypt File System		5	complete

Figure 1: Simulation Overview with Playbook Display

We plan to enhance the chart by adding options to include plots for when beacons are transmitted and when jobs are run. Additionally, all plotted points will be selectable, and information will be displayed when the points are selected. Because of axis scaling issues, an option to switch to a logarithmic scale will be included. Options to filter the graphed data by the specified jobs will be included as well.



Simulation Properties

Simulation Name:
[✓] Log job output to file
Webshell
Protocol: ● TCP ● HTTP ● HTTPS ● UDP ● WebSocket ● HTTP/2 ● HTTP/3 ● REST
Server Listening Port:
Beacon time: seconds
Jitter: seconds
Beacon Type: Dynamic Beaconing
Padding Type: ● Flat ● Random
Connection Profile: ● Long ● Short
Number of Tunnels:

Playbook

Job Type	Content	Delay	Scaling	Actions

New Command

Command Type:
● Portscan
● Download File
● Upload File
● Dump Configuration
● Remote Task
● Active Directory Scanning
● Exfiltrate Data
● Encrypt File System
● InterCommand Delay
● Remote Terminal
● Key Logger
● Arbitrary Command
● MAAD Job
● DeRF Job

Delay: 0 seconds
[✓ No Output Scaling]
● Fixed Output Scaling
● Random Output Scaling

Figure 2: Create Simulation Page

Additional menu options will be included on the page used to create new simulations. These menu options will allow you to select the additional functionalities we are adding to the project. This includes additional protocol options, Advanced C2 options, and additional commands. New options are highlighted in a green box. When arbitrary commands, MAAD, or DeRF jobs are selected, a textbox will open allowing the user to type in the exact commands they wish to run using the hybrid attack modules or arbitrary commands. Additionally, users have the new option to scale the output of the jobs by a fixed amount or random amount.

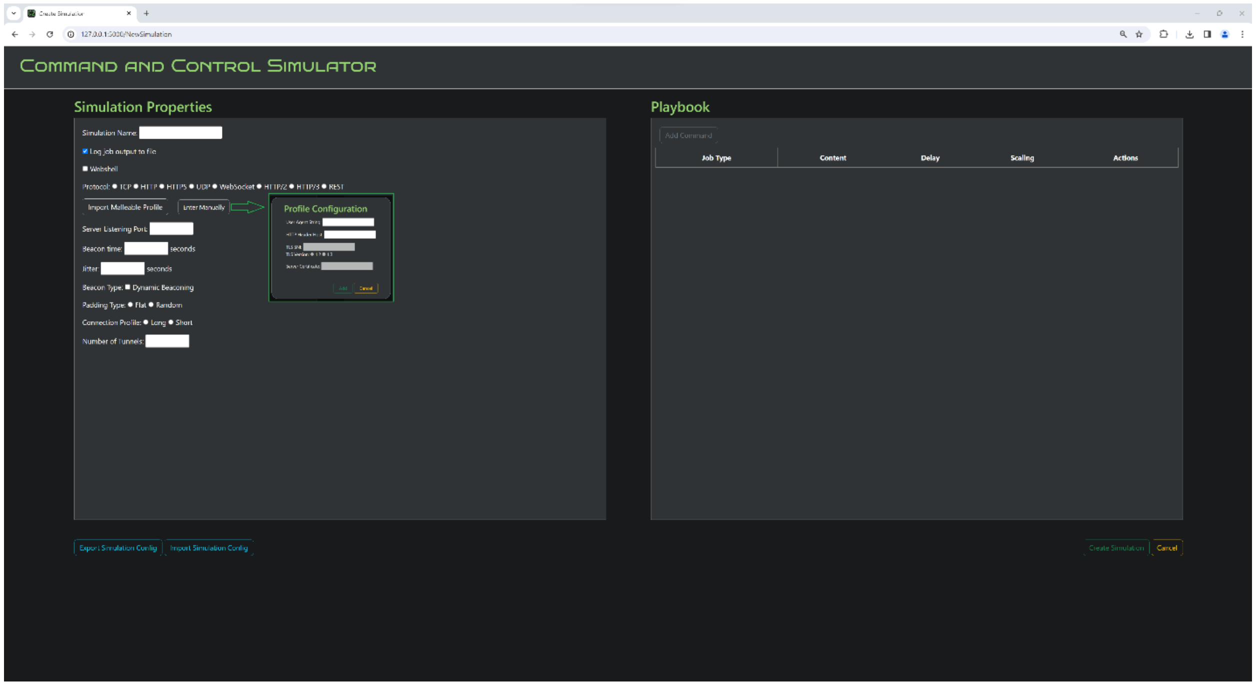
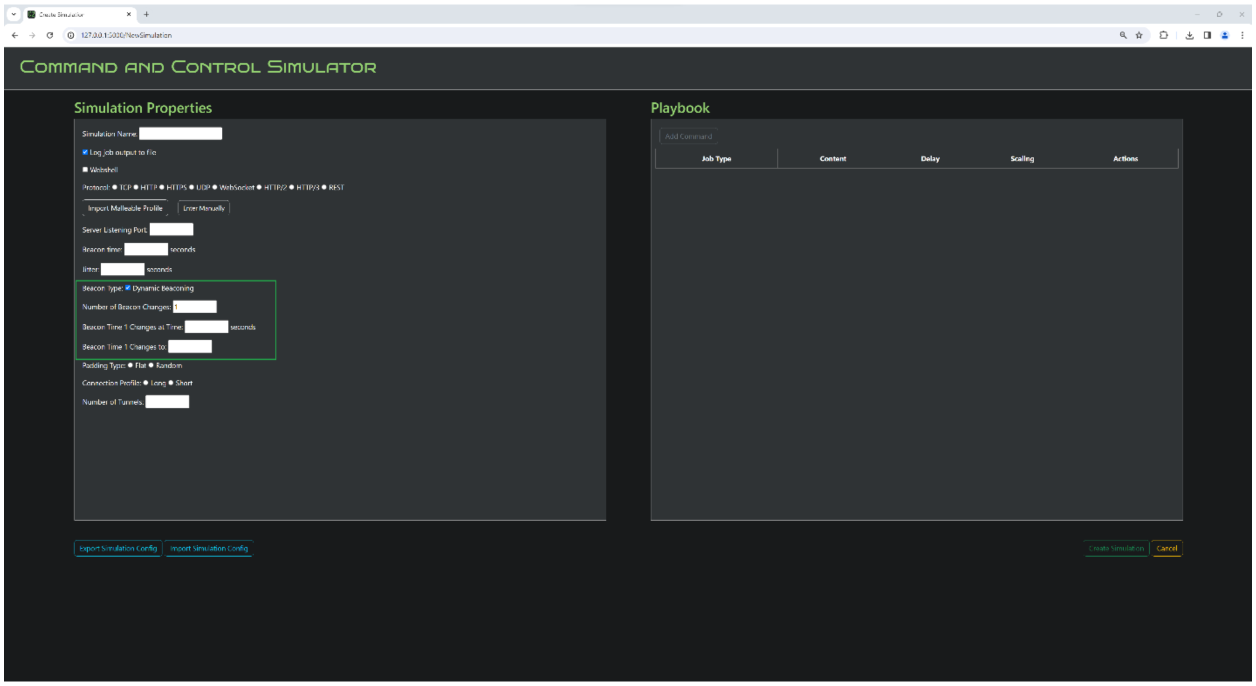


Figure 3: Malleable Profile Option

If a user wishes to simulate malleable profiles, they have the option of pasting a JSON file or entering fields by hand, adding the capability to specify user agent string and TLS version, among others.



Simulation Properties

Simulation Name:

[✓] Log job output to file

Webshell

Protocol: ● TCP ● HTTP ● HTTPS ● UDP ● WebSocket ● HTTP/2 ● HTTP/3 ● REST

Server Listening Port:

Beacon time: seconds

Jitter: seconds

Beacon Type: [✓] Dynamic Beaconing

Number of Beacon Changes: 1

Beacon Time 1 Changes at Time: seconds

Beacon Time 1 Changes to:

Padding Type: ● Flat ● Random

Connection Profile: ● Long ● Short

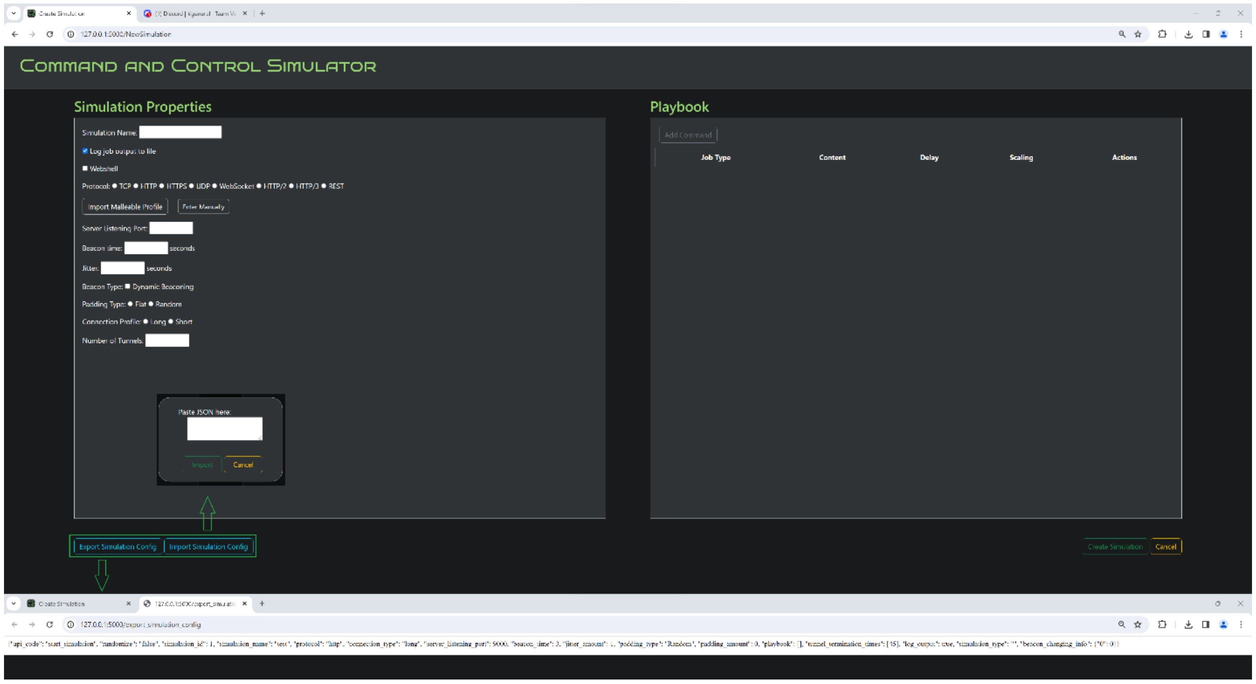
Number of Tunnels:

Playbook

Job Type	Content	Delay	Scaling	Actions

Figure 4: Dynamic Beaconing

Dynamic beaconing allows users to change the beacon at specific times during the simulation. When selected the users is first prompted with how many beacon changes they wish to make. Based on that number, the correct number of inputs shows. The first input field asks for what time during the simulation the beacon should change and the second field asks what the beacon interval should change to.



{"api_code": "start_simulation", "randomize": "false", "simulation_id": 1, "simulation_name": "test", "protocol": "http", "connection_type": "long", "server_listening_port": 5000, "beacon_time": 3, "jitter_amount": 1, "padding_type": "Random", "padding_amount": 0, "playbook": [], "tunnel_termination_times": [15], "log_output": true, "simulation_type": "", "beacon_changing_info": {"0": 0}}

Figure 5: Export/Import Simulation

Users have the option to export the simulation data if they wish to revisit the simulation without having to conduct the simulation again. Users can then import that configuration file for ease of use.

Technical Specifications

Software Architecture

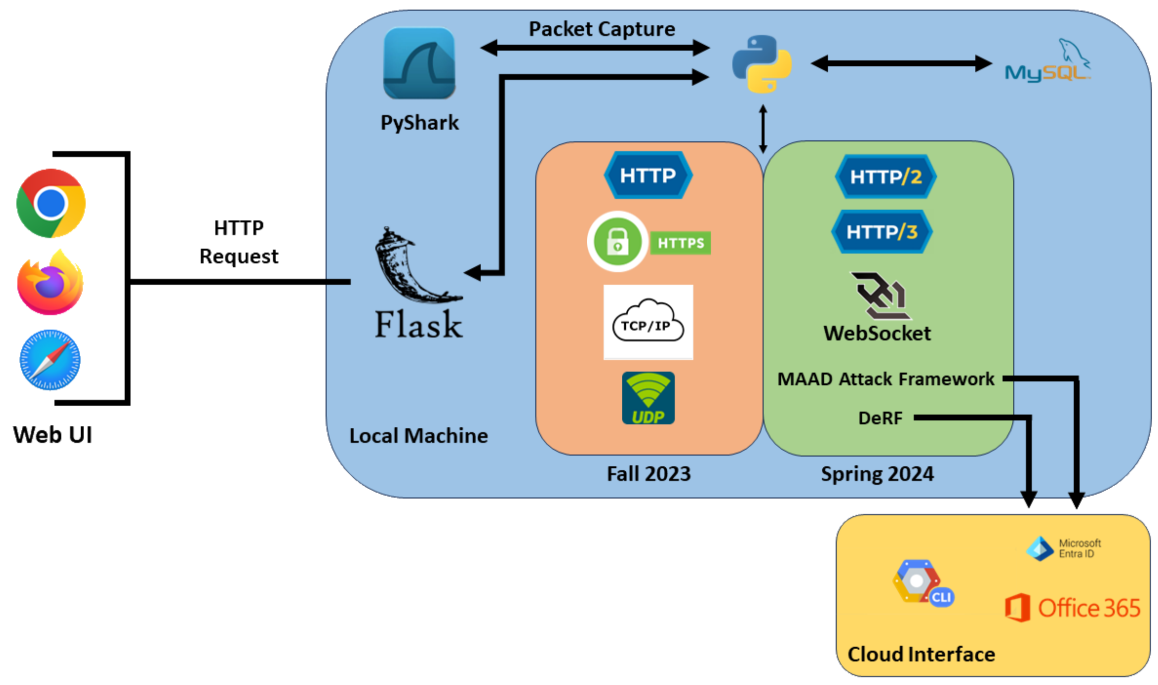


Figure 5: Software Architecture Diagram

The Command and Control (C2) Simulator is built using Python3 and MySQL running together on a local machine, along with implementations of 7 different networking protocols and 2 hybrid-attack simulation frameworks (MAAD-AF and DeRF). In Figure 05 there is a clear differentiation between the Fall 2023 and Spring 2024 semesters and the network protocols implemented within each to show what exactly our team is adding to the Simulator. This semester, our team is adding HTTP/2, HTTP/3, and WebSocket as well as the previously mentioned hybrid-attack frameworks. The previous semester's team implemented HTTP/1.1, HTTPS, TCP, and UDP.

The implementations for all of these protocols and the packet capture process are assisted by multiple Python libraries and frameworks. As opposed to most other Python libraries we use in the project, the PyShark library is shown in this diagram to illuminate how integral it is to the goals of the C2 Simulator project. PyShark analyzes the network packets sent between the client and server and delivers those to a local MySQL database via a database interface created within our project, which is useful for feeding accurate network traffic data to Vectra's AI models and to the Web UI for visualization via graphs.

The front-end user interface is created using the Python Flask framework to generate a clean and resource-efficient web page displayed on popular web browsers via an HTTP request. Users can configure a simulation as they please on the front-end interface and view the network traffic displayed on a line graph which shows bytes sent by both the client and the server.

To enable more automated simulations and testing, users can also skip the Web User Interface and choose to instead use a python script included in the project. This script can hold as many pre-defined simulations configured with the exact conditions a user desires. Rather than having to re-input configuration details each time a simulation is run, users can replace which pre-defined set of jobs, or “playbook” they would like to use.

System Components

Development Environments/Libraries

Python
Python is a popular high-level programming language designed for general purpose usage. Python includes many libraries that add functionality for client-server interactions, web-based user interfaces, network analysis, and database integration.

VSCode
Visual Studio Code is a free developmental IDE. It is easy to install and works on all of the teams computers. It also allows for faster development with built-in debugging tools and simple code analysis.

PyShark
PyShark is a Python library which creates a wrapper for Wireshark’s network traffic analysis functionality. Users can directly interact with and parse packet capture files using Python, making it ideal for this project.

Flask
Flask is a micro web framework within Python that enables programmers to create web applications quickly and simply. Flask is known for its flexibility and lightweight design, making it easy to implement and fits with the purposes of this project.

Server Software

MySQL
MySQL is an open-source database management system used for storing and managing relational data. It is highly focused on reliability and performance, which makes it useful for the purposes of this project.

Risk Analysis

Compatibility and Integration Issues

Difficulty: Low

Risk: We need to implement third party tools such as MAAD and DeRF. There is a risk of these tools not being able to be integrated properly or work with each other's versions and library requirements.

Mitigation: Ensure that all technologies can be updated and will continue to be supported for the lifecycle of our project. As long as the tools are continued to be updated then there won't be any versioning or compatibility issues between them.

Generating Realistic Data

Difficulty: Low

Risk: Vectra's AI training requires realistic data for accurate attack mitigation. The risk lies in the difficulty of obtaining and modeling data that truly reflects real-world scenarios, which is essential for effective AI training.

Mitigation: We will find real world examples and model our data and timing based on those real-world attacks. This approach will ensure the AI model is exposed to a wide range of situations.

Performance Issues

Difficulty: Medium

Risk: The project needs to be able to generate substantial amounts of data for an AI model to be trained on. Third party apps might not allow for efficient API calls. In addition, Large Language Models require a significant amount of computational power. In addition, we need our program to be fast and stable enough to continuously generate data for at least 24 hours.

Mitigation: We will evaluate performance and optimize code. In addition, we will look at fallback features such as load balancing and distributed computing. With this any performance bottleneck should be mitigated

Portability to Other OS

Difficulty: Medium

Risk: The project may face challenges in being compatible and portable across different operating systems. This would make it hard for Vectra to use with all their employees. It should be available across MacOS, Windows, and Linux.

Mitigation: The team will utilize cross-platform libraries to ensure compatibility. In addition, we will look at other options such as AWS for hosting a server and using API calls allow for more compatibility.

Schedule

Week 1 (1/7 – 1/12)

- Initial meeting with team members
- Initial client meeting
- Get starter code running (last semester's project)

Week 2 (1/13 – 1/20)

- Initial triage meeting with TM Luke Sperling
- Research & understand starter code
- Understand project requirements
- Cut tickets using Jira Agile Board, divide tickets into sprints for the rest of the semester

Week 3 (1/21 – 1/27)

- Status Report Presentation
- Work on Project Plan Document and Presentation
- Begin Sprint 1
  - Add required internet protocols (Websockets, HTTP2)
  - MAAD-AF Integration
  - Run arbitrary command on C2 client
  - Add logarithmic scaling to PCAP graph

Week 4 (1/28 – 2/3)

- Continue Sprint 1
- Begin Sprint 2
  - DeRF Integration

    - Beaconless C2
    - Webshells
    - Dynamic Response Support
    - Add UI for job output

Week 5 (2/4 – 2/10)

- ➢ Prepare Alpha Presentation
- ➢ Status Report Presentation
- ➢ Continue Sprint 2

Week 6 (2/11 – 2/17)

- ➢ Prepare Alpha Presentation
- ➢ Continue and complete Sprint 2
- ➢ Begin Sprint 3
    - Usability improvements: Import/Export JSON Config
    - Stop Job
    - Add actions to PCAP plots
    - REST API Integration
    - Continue Beaconless & Webshell Support

Week 7 (2/18 – 2/24)

- ➢ Alpha Presentation
- ➢ Continue Sprint 3

Week 8 (2/25 – 3/2)

- ➢ Spring Break

Week 9 (3/3 – 3/9)

- Status Report Presentation
- Prepare Beta Presentation
- Continue Sprint 3

Week 10 (3/10 – 3/16)

- Continue Sprint 3

Week 11 (3/17 – 3/23)

- Status Report Presentation
- Prepare Beta Presentation
- Complete Sprint 3
- Begin Sprint 4
    - Malleable Profile
    - Add HTTP/3 protocol
    - Dynamic Beaconing

Week 12 (3/24 – 3/30)

- Status Report Presentation
- Beta Presentation
- Complete Sprint 4
- Bug Fixes in preparation for Beta

Week 13 (3/31 – 4/6)

- Beta Presentation
- Bug fixes

Week 14 (4/7 – 4/13)

- Status Report Presentation

- Create project video
- Complete Documentation
- Bug fixes

Week 15 (4/14 – 4/20)

- Design Day
- All deliverables due