unCEAN C2: Adversary Emulation

IMPLANT INFORMATION

The unCEAN C2 asynchronously awaits new implant registrations on the C2 Discord server. Discord is used to bypass traffic restrictions, anonymize traffic, and provide a defensive layer between the C2 and the implant. The implant is in a loop, regularly checking in for new commands, while updating symmetric keys upon its heartbeat.

TWO MODES OF OPERATION

The unCEAN implant automatically decides between using the Discord Bot API or an HTTP fallback as its C2 channel.

If Discord is unavailable, the implant will attempt to connect to the C2 over HTTPS.

EXECUTION

The Implant supports a wide array of functionality:

- Steals Chrome passwords, credit cards, autofill data, history, and cookies
- Gathers information about the user - such as their username, and directories
- Spy on the network and grab IPs of other systems by collecting adapter information
- Discover system information including the version, build, the system's guid, and a list of processes
- Executes processes via shell code injection and command line commands

Innovation Journey: Pivots and Future Work

BRAINSTORMING

Our goal for this project is to create an implant that would be more difficult to detect, especially on a busy network.

Upon investigating, we found the Discord Bot API to be an excellent way to move traffic between the infected machine and our C2 server.

Instead of an app making unusual connections, our implant looks like a chatty Discord user!

PLANNING

We began planning our project by creating a desired set of functionalities, while exploring how to architecture the system to ideally serve those choices.

Next, we chose which tools to use when engineering our system: Flask, MySQL, C, and a dedicated Discord Listener. Other tools include Libsodium, Docker, and the Windows API.

CONSTRUCTION/TESTING

After the planning phase, we began construction of our project's components.

We used GitHub for managing our workflow, using GitHub actions for our CI flow.

Tests include testing our Docker containers for our Flask App, Discord Listener, and MySQL database, along with unit tests for our Flask endpoints.

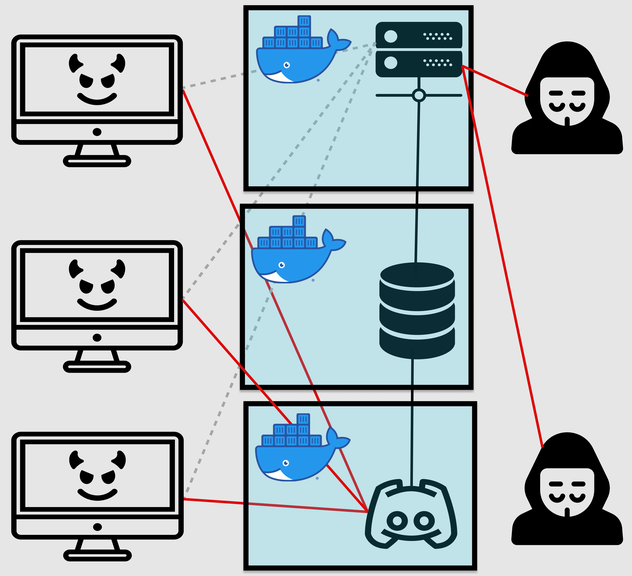
EXPANSION

After creating a prototype, we implemented features for gathering data about the user and the system.

We also continued to improve features of the implant by adding a process injection, as well as a persistence method for the implant to start on user logon.

We plan on further expanding our implant's functionality.

ARCHITECHTURE



Our C2 system architecture contains:

a. A Python Flask application dedicated to the client and HTTP fallback method

b. A Python Discord Listener dedicated to handle implant messaging

c. A MySQL database container

Our communication security is based on a hybrid cryptographic scheme including:

a. Elliptic Curve Cryptography (Asymmetric)

b. Salsa20 (Symmetric)

Our tech stack for client-facing code includes:

- Python CLI Client
- Implant written in C which utilizes the Win32 API