Offensive Security Orchestration

Client: Withheld at clients request
Advisor: Doug Jacobson

Summary

Our project aims to develop a security orchestration platform for our client which will allow them to conduct red team engagements in a stealthy and efficient manner. By developing a custom implant which integrates into the security orchestration platform we will be able to bypass the tools which flag on more common implants. The security orchestration platform will also automate the deployment of implants so that our client can rapidly develop and test different payloads.

Functional Requirements

Bot requirements

Communicates with C2 via a secure, encrypted RESTful API
Is tested and executable on recent versions of Microsoft Windows Operating System
Is able to disconnect from a C2 and destroy itself
Supports domain fronting with Amazon Cloudfront and frontable domains
Demonstrates scheduled task persistence while remaining stealthy
Endpoint detection and response solution bypass capabilities

Command and Control requirements

Communicates over encrypted channel with multiple bots
Has a dedicated ReactJS single page application for managing bots
Uses Django Python Framework to manage APIs and database queries
Provides a user interface for sending commands to different bots
Provides documentation and help to the user
Allows user creation, deletion, and authentication
Logs all activity by users
Has realtime websockets for receiving data from the bots
Allows building and downloading of malware in-app
App managed as containers via docker-compose

System Implementation

Semantic UI CSS framework
Django Python Web framework
Django REST Framework (for APIs)
jQuery Javascript Library
SQLite
Docker
C#
Cuckoo

Testing and Evaluation

Tests focused on completing deliverables
Focusing on manual testing over Automated testing
Time sink is too large for Automated testing with minimal reward
Can use that time for developing additional features
EX: Ensure that our malware is not detected by common EDR solutions

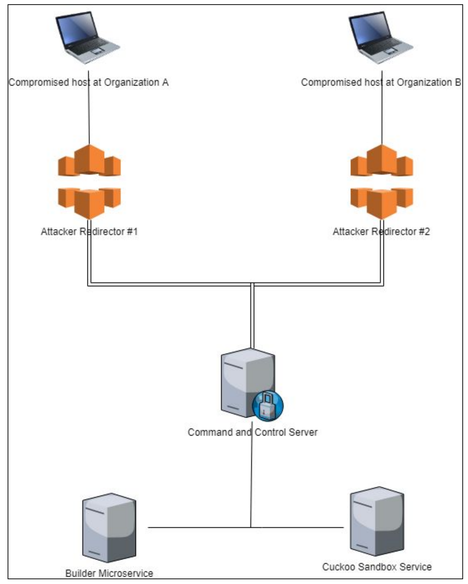
Applicable Standards

TLS protocol standard
https://tools.ietf.org/html/rfc5246
HTTP standard
https://tools.ietf.org/html/rfc2616

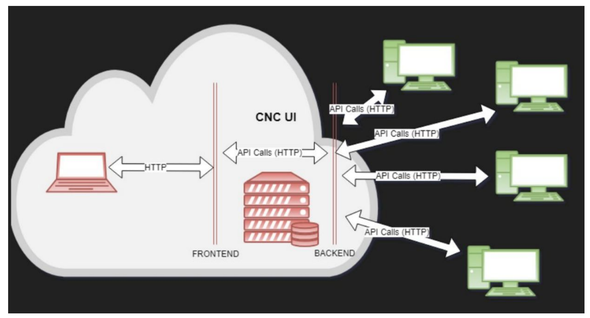
Non-Functional Requirements

Non-regular intervals for heartbeats
Bot self-destruction
Stealthy bots - no virus warnings
Ease of navigation on C2 server
Multi-user simultaneous access

System Architecture and Design



Concept diagram



Conclusion

Over this semester, we have completed work on this security orchestration platform that allows our client to conduct red team engagements. We completed almost every single requirement, and the ones we did not do were determined to not be important. While the project is now functional and can be used for future engagements, future extensions to the project will be focused on maintaining the ability to perform domain fronting as Content Delivery Networks attempt to disable this functionality.