Assimilate - A Takedown Resistant Botnet Simulation

Elang Sisson, Zachary Adelson, Lukes Godwin, Christopher Mendoza
Mentors: Justin Van Der Sluys, Sean Hodgson, Fredy Fernandez, Nathan Waltz

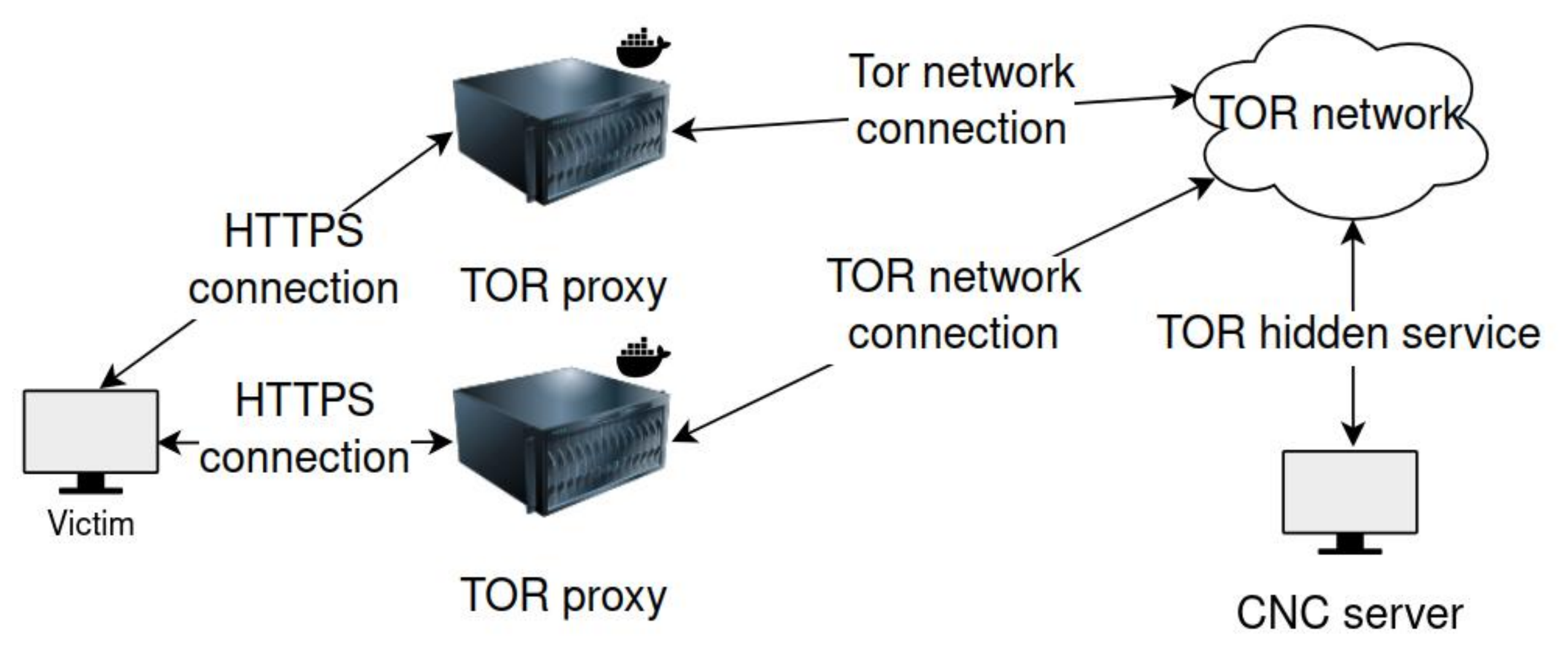
Introduction

Traditional methods to disrupt botnets by government bodies - especially law enforcement agencies - include physically seizing the command and control (CNC) servers or forcing the registrar to revoke the domain name used by the botnet - known as DNS sinkholing. These methods target the infrastructure used to manage the botnet - aiming to sever the communication between the operations server and the infected devices.

To address this problem, we implemented a distributed botnet simulation with the following properties:

- We created the CNC server as a TOR hidden service - this makes it next to impossible to take down the botnet simulation's TOR domain name using DNS sinkholing.
- We wrote the botnet in the Go programming language, which has superior support for network programming and cross compilation to operating systems (Linux, Mac, Windows, Plan9, BSD) and different architectures - including obscure CPU architectures like RISC-V, ARM, and PowerPC.
- We implemented a distributed, poll-based architecture - where clients poll a message queue residing in the CNC server. We chose this over a websocket approach due to the ability to tune the poll frequency for improved covert communications.

Assimilate network diagram



Networking

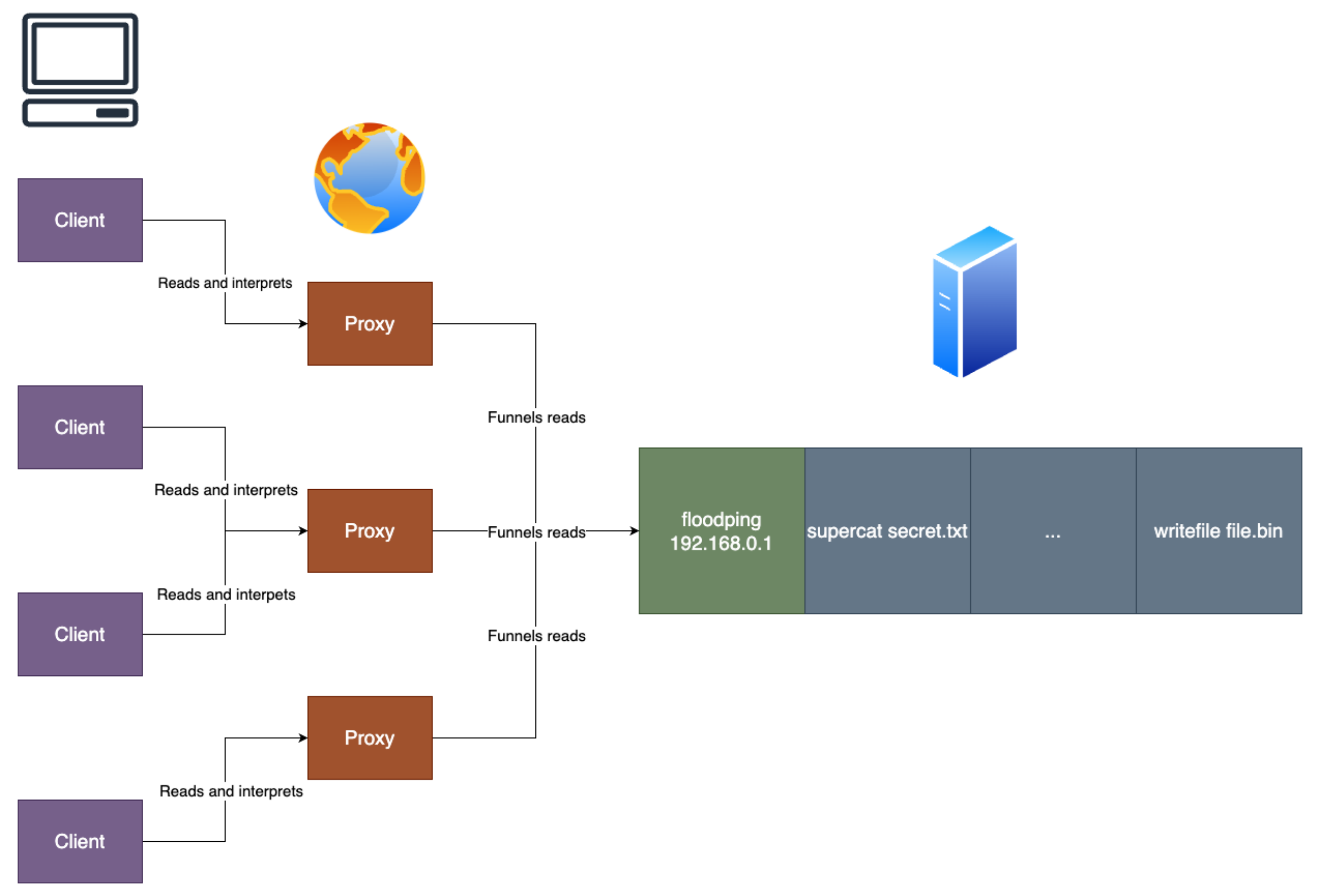
The network focused segment of our project creates a simple proxy driven model to ferry data to-and-from victims. Connections are routed through the TOR network. Furthermore, the proxy servers are engineered to be operated on trusted cloud VPS servers - which allows them to blend in with regular servers.

- All victims poll the CNC server though the proxies to hide the suspicious connection to TOR from the victim's anti-malware.
- The victims work on a pull down model for data flow to get around NAT limitations.
- Established a secure DNS name assignment for the CNC server via the TOR network to make DNS sinkholing impossible.

Command and Control

The command and control server is the heart of the botnet simulation, and provides a centralized location for management of the cluster. We implemented a custom command interpreter - allowing our custom commands to be run on MacOS, Linux, and Windows.

Message Queue



Later Work

- Embed a Lua interpreter or create a custom DSL for arbitrary code execution support without depending on client's installed packages.
- Add x509 certificate management to assimilate - this is a very big security hole in our implementation.
- Add a user interface for the command and control server for easy management of the botnet simulation.

Endpoint

We provide a simple, statically-linked binary executable for the victims that can be easily extended in the future to support further capabilities. We implement custom commands which may be interpreted differently on any of the three supported operating systems (MacOS, Windows, Linux).

- Each compromised device registers itself as an endpoint using a unique hardware hash - implementing a rudimentary mechanism for device identification.
- The endpoints rely on a polling mechanism which periodically sends POST requests to the proxy servers. This allows us to evade detection by mimicking regular HTTP traffic.
- Viewed as a black box, an endpoint can either register itself, receive a command, or execute a command and return the results.

Acknowledgement

This work is supported by funding for the VICEROY Northwest Institute for Cybersecurity Education and Research (CySER) provided by The Office of the Undersecretary of Defense for Research and Engineering, in collaboration with the Air Force Research Laboratory and Griffiss Institute.