Modular Botnet Simulation and Analysis

Presented by: Team – अनिर्वचनीय
Internship: ECSIP
Date: 12th July 2025

Understanding Botnets

Botnets are networks of compromised systems (bots) remotely controlled by a central Command and Control (C2) server. They are commonly used in cyberattacks such as Distributed Denial of Service (DDoS), credential theft, and malware distribution.

To demonstrate the real-world impact, the infamous **Mirai botnet** continues to evolve and threaten global infrastructure. In 2024, Kaspersky reported a new Mirai variant exploiting DVR devices (CVE-2024-3721), with over **50,000 vulnerable systems exposed online**, including many in India*. This reflects the growing risk from millions of unpatched IoT devices like routers, cameras, and DVRs that can be silently hijacked and weaponized for large-scale attacks.

* securelist.com

The Challenge of Botnet Defense

Bridging the Knowledge Gap

Security professionals require an in-depth understanding of botnet architecture to develop effective defenses. A lack of safe, practical training environments limits the ability to gain hands-on experience with real-world botnet behavior.

Learning Through Simulation

Our project aims to build practical expertise by simulating a modular botnet in a safe, controlled environment using Virtual Machines. This hands-on approach helps us better understand botnet behavior paving the way for stronger mitigation strategies in real-world scenarios.

Key Objectives

Modular Bot Development

Build Python-based bots compatible with both Linux and Windows.

C2 Web Dashboard

Develop a centralized web dashboard for comprehensive bot control.

Attack Simulation

Implement realistic botnet modules, including keyloggers, stealer, spyware, scanners, and DDoS attacks.

Wireshark Monitoring

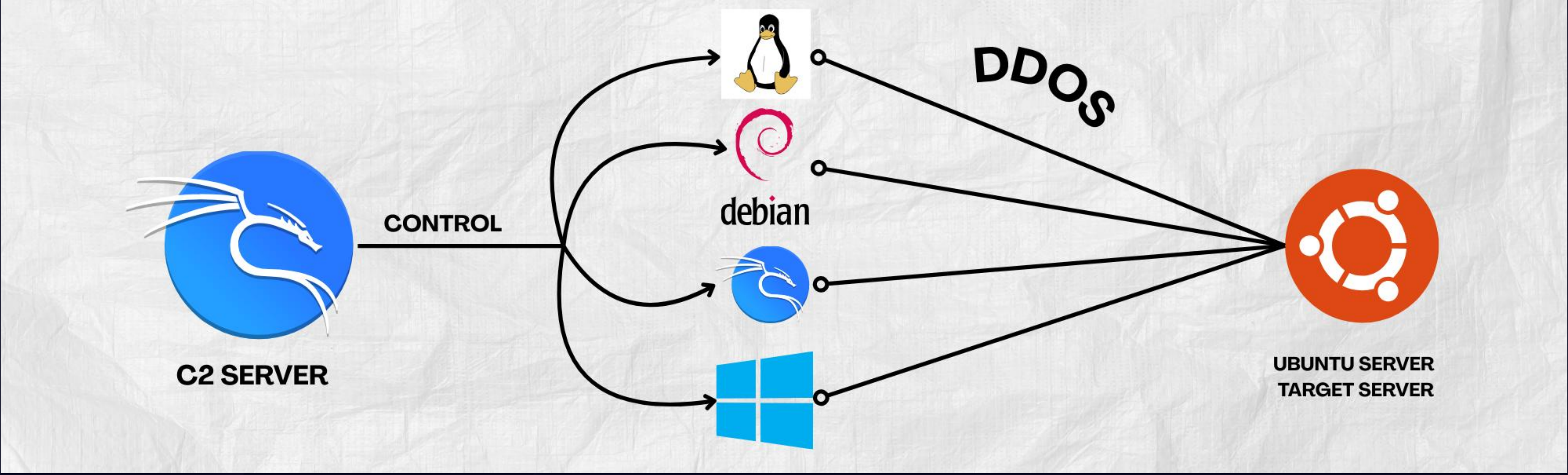
Monitor all network traffic using Wireshark for in-depth analysis.

Ethical Isolation

Ensure complete ethical containment within a dedicated virtual lab setup.

Tools & Technologies

Component	Tools Used
Bots	Python (requests, subprocess etc)
C2 Server	Python(Flask)
Virtual Lab	VirtualBox, OpenVPN server for connection between VMs
Target Server	Python(Flask) with login page.
Monitoring	Wireshark



System Architecture

To safely simulate real-world botnet operations, we created a virtual lab where each team member runs a VM as a bot or server. A Flask-based C2 server manages command distribution and logging, while an OpenVPN network links all nodes securely. Bots include four Linux agents and one Windows variant, targeting a dedicated Ubuntu web server. This design allows realistic attack testing without exposing systems to the public internet.

Bot Functionality

C2 Registration & Polling

Each bot registers with the C2 server on startup and periodically polls for new tasks via HTTP.

Command Execution

Bots execute various commands, including system information retrieval (`whoami`) and attack commands from modules like `keylogger, stealer,` etc.

Payload Modules

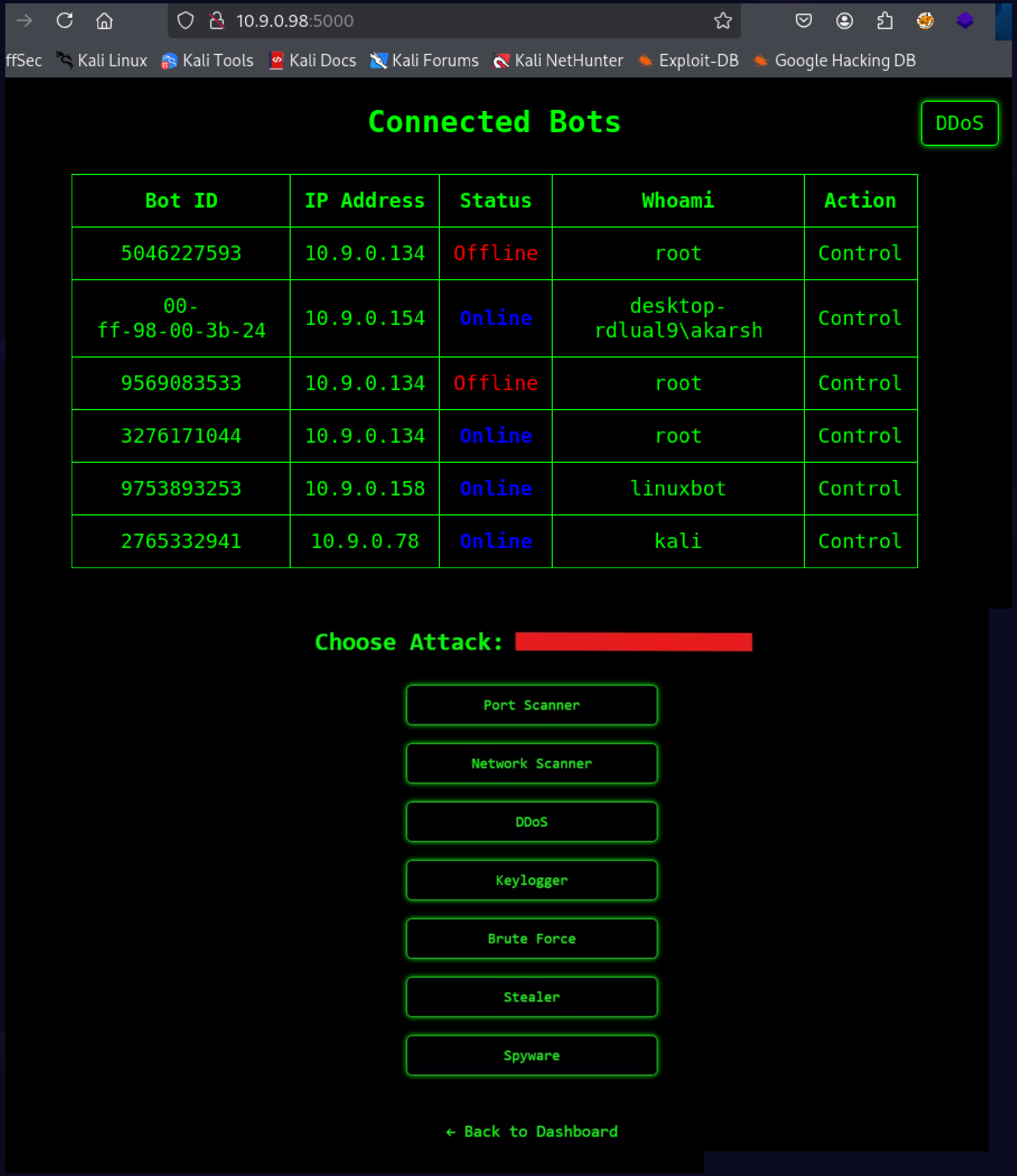
Supports advanced modules like keyloggers for data capture, port scanners for reconnaissance, and DDoS for service disruption.

Result Reporting

Execution results from commands and modules are transmitted back to the C2 server.

C2 Dashboard

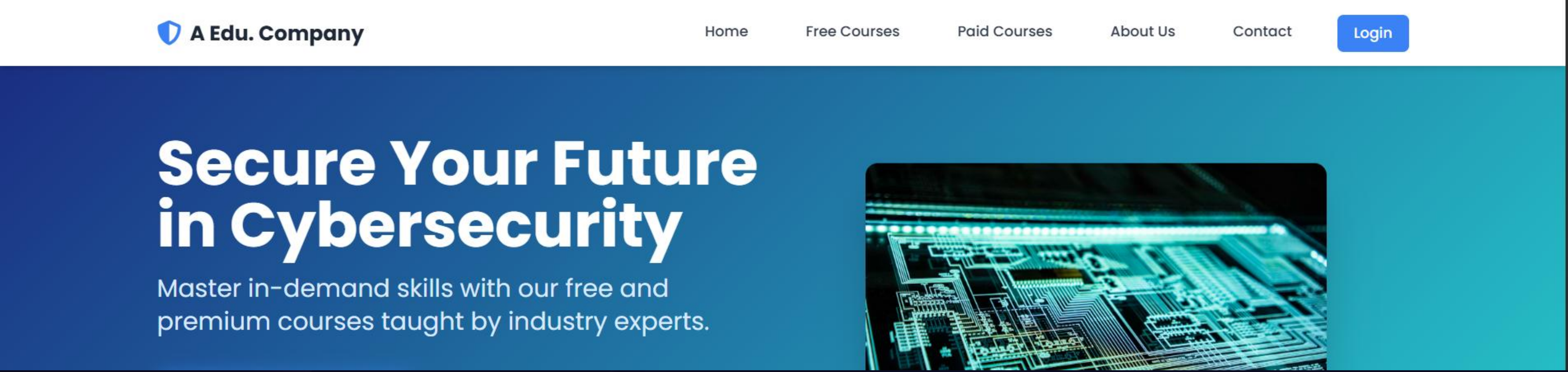
- Built with Flask.
- Allows viewing of registered bots and their status.
- Facilitates sending commands to individual or multiple bots.
- Provides a simple interface for inputting attack commands.
- Enables triggering of advanced modules, such as DDoS attacks.
- Displays real-time responses and logs from bot executions.



Connected Bots

Bot ID	IP Address	Status	Whoami	Action
5046227593	10.9.0.134	Offline	root	Control
00-ff-98-00-3b-24	10.9.0.154	Online	desktop-rdlual9\akarsh	Control
9569083533	10.9.0.134	Offline	root	Control
3276171044	10.9.0.134	Online	root	Control
9753893253	10.9.0.158	Online	linuxbot	Control
2765332941	10.9.0.78	Online	kali	Control

Choose Attack: [redacted]



A Edu. Company

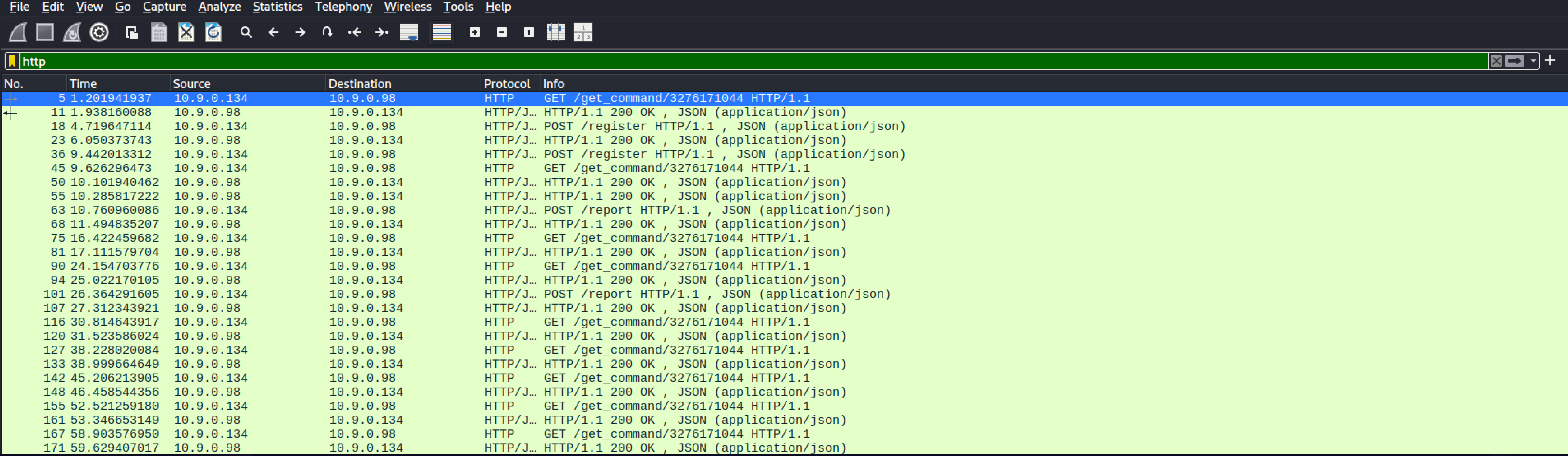
Home · Free Courses · Paid Courses · About Us · Contact · Login

Secure Your Future in Cybersecurity

Master in-demand skills with our free and premium courses taught by industry experts.

Target Server & Attack Simulation

Our target server, an Ubuntu machine running a simple Flask HTTP application, serves as the victim for our simulated attacks. The primary focus is on a **DDoS Flood** using HTTP GET requests, orchestrated by selected bots via the C2. We also simulate port scanning to demonstrate reconnaissance capabilities. Attack logs are observed directly in the Flask server terminal and monitored in real-time through Wireshark for network-level insights into the HTTP floods.

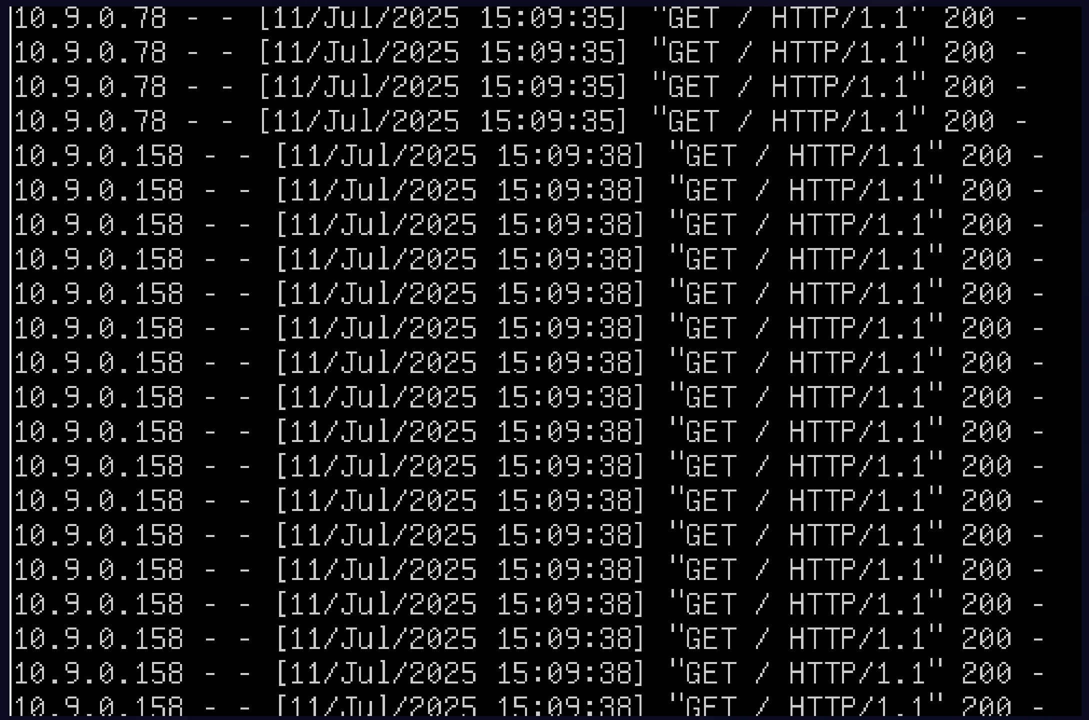


```
http
```

```
No.   Time          Source       Destination  Protocol  Info
    5 1.201941937   10.9.0.134   10.9.0.98    HTTP      GET /get_command/3276171044 HTTP/1.1
   11 1.938160088   10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
   18 4.719647114   10.9.0.134   10.9.0.98    HTTP/J…   POST /register HTTP/1.1 , JSON (application/json)
   23 6.050373743   10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
   36 9.442013312   10.9.0.134   10.9.0.98    HTTP/J…   POST /register HTTP/1.1 , JSON (application/json)
   45 9.626296473   10.9.0.134   10.9.0.98    HTTP      GET /get_command/3276171044 HTTP/1.1
   50 10.101940462  10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
   55 10.285817222  10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
   63 10.760960086  10.9.0.134   10.9.0.98    HTTP/J…   POST /report HTTP/1.1 , JSON (application/json)
   68 11.494835207  10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
   75 16.422459682  10.9.0.134   10.9.0.98    HTTP      GET /get_command/3276171044 HTTP/1.1
   81 17.111579704  10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
   90 24.154703776  10.9.0.134   10.9.0.98    HTTP      GET /get_command/3276171044 HTTP/1.1
   94 25.022170105  10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
  101 26.364291605  10.9.0.134   10.9.0.98    HTTP/J…   POST /report HTTP/1.1 , JSON (application/json)
  107 27.312343921  10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
  116 30.814643917  10.9.0.134   10.9.0.98    HTTP      GET /get_command/3276171044 HTTP/1.1
  120 31.523586024  10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
  127 38.228020084  10.9.0.134   10.9.0.98    HTTP      GET /get_command/3276171044 HTTP/1.1
  133 38.999664649  10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
  142 45.206213905  10.9.0.134   10.9.0.98    HTTP      GET /get_command/3276171044 HTTP/1.1
  148 46.458544356  10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
  155 52.521259180  10.9.0.134   10.9.0.98    HTTP      GET /get_command/3276171044 HTTP/1.1
  161 53.346653149  10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
  167 58.903576950  10.9.0.134   10.9.0.98    HTTP      GET /get_command/3276171044 HTTP/1.1
  171 59.629407017  10.9.0.98    10.9.0.134   HTTP/J…   HTTP/1.1 200 OK , JSON (application/json)
```

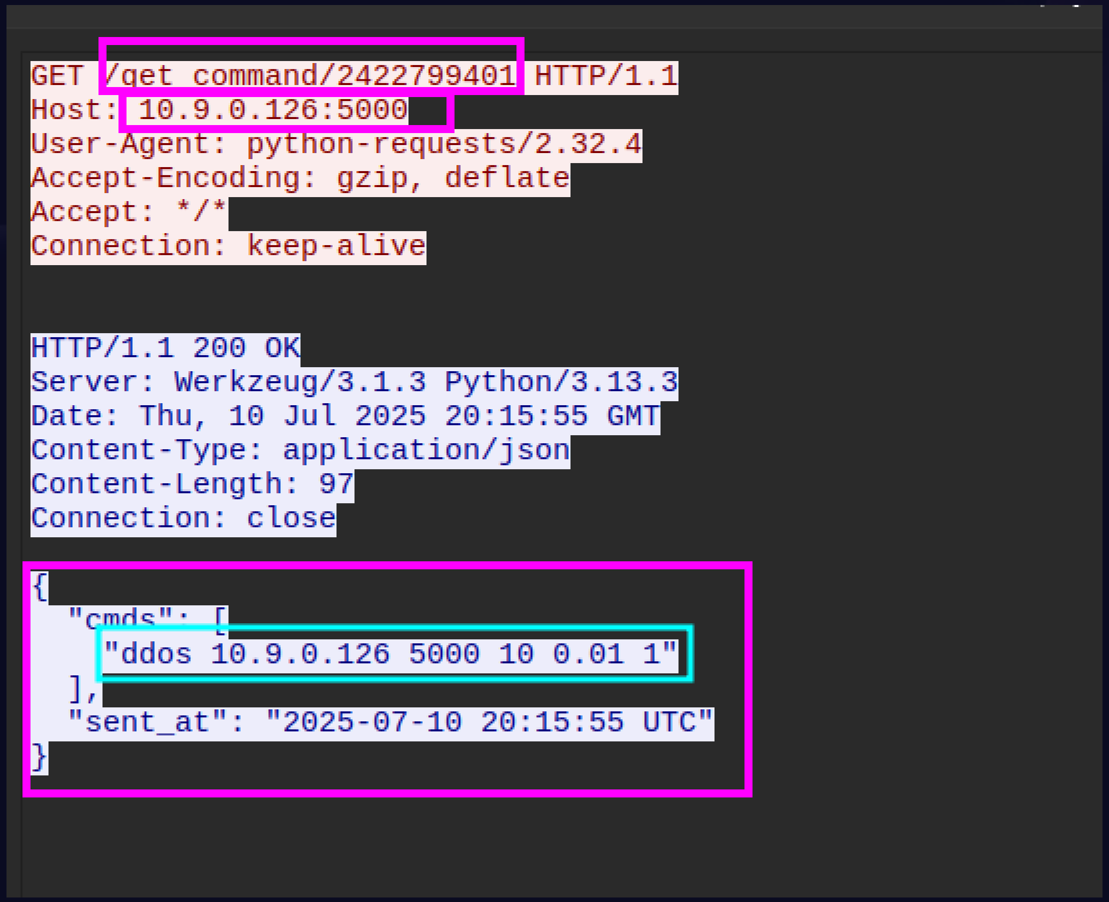
Wireshark-Based Network Analysis

Wireshark was instrumental in capturing and analyzing all network traffic within our virtual lab. This included detailed observations of bot registration, HTTP command polling, and the distinct patterns of DDoS flood attacks. Our analysis focused on IP/port behavior, packet timing, and frequency to understand the botnet's operational footprint. However, it's important to note the limitations: our scope was restricted to unencrypted HTTP-based C2, excluding encrypted traffic or other covert channels like DNS or TCP reverse shells.



```
10.9.0.78 - - [11/Jul/2025 15:09:35] "GET / HTTP/1.1" 200 -
10.9.0.78 - - [11/Jul/2025 15:09:35] "GET / HTTP/1.1" 200 -
10.9.0.78 - - [11/Jul/2025 15:09:35] "GET / HTTP/1.1" 200 -
10.9.0.78 - - [11/Jul/2025 15:09:35] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
10.9.0.158 - - [11/Jul/2025 15:09:38] "GET / HTTP/1.1" 200 -
```

DDoS Attack on Test Website by bots



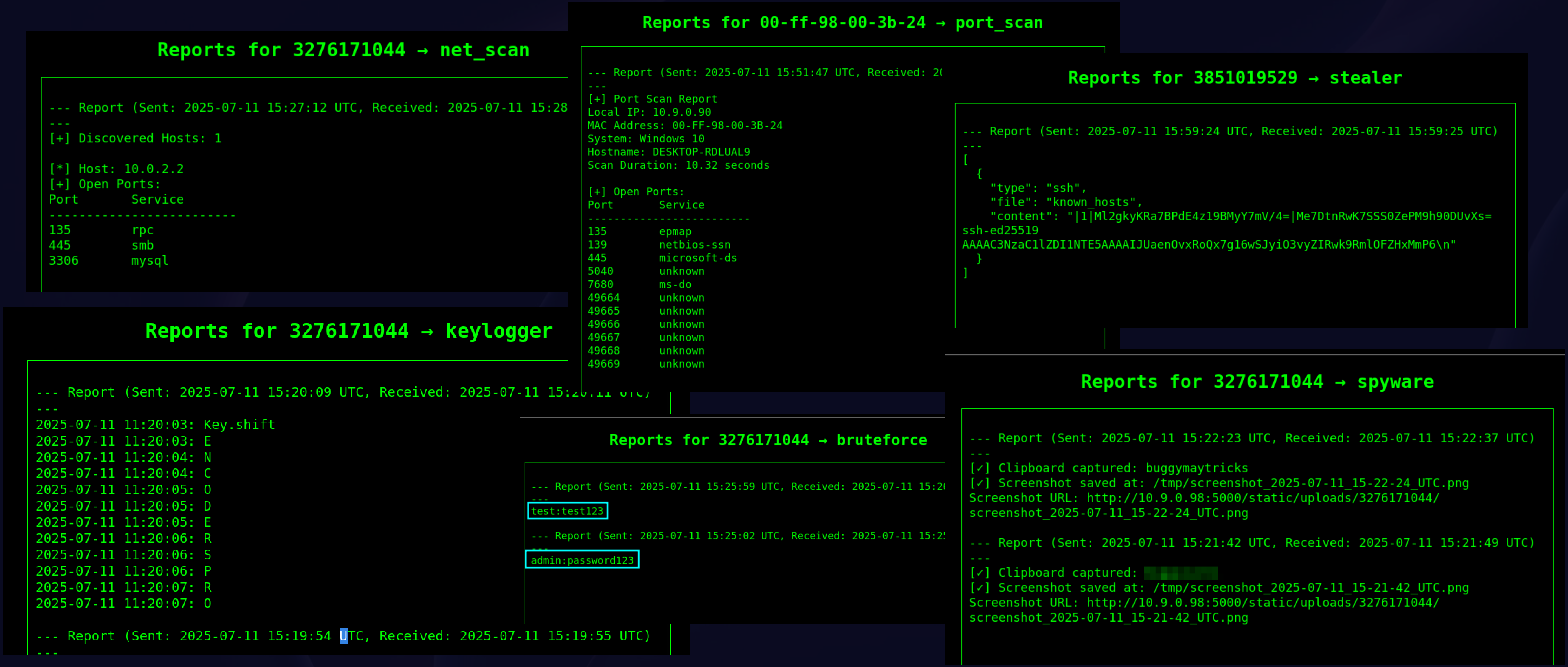
```
GET /get_command/2422799401 HTTP/1.1
Host: 10.9.0.126:5000
User-Agent: python-requests/2.32.4
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive


HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.13.3
Date: Thu, 10 Jul 2025 20:15:55 GMT
Content-Type: application/json
Content-Length: 97
Connection: close

{
  "cmds": [
    "ddos 10.9.0.126 5000 10 0.01 1"
  ],
  "sent_at": "2025-07-10 20:15:55 UTC"
}
```

HTTP communication between a C2 server and a bot, observed using Wireshark

Various Attack Reports on a Bot System



Reports for 3276171044 → net_scan

```
--- Report (Sent: 2025-07-11 15:27:12 UTC, Received: 2025-07-11 15:28
---
[+] Discovered Hosts: 1

[*] Host: 10.0.2.2
[+] Open Ports:
Port       Service
-------------------------
135        rpc
445        smb
3306       mysql
```

Reports for 00-ff-98-00-3b-24 → port_scan

```
--- Report (Sent: 2025-07-11 15:51:47 UTC, Received: 2(
---
[+] Port Scan Report
Local IP: 10.9.0.90
MAC Address: 00-FF-98-00-3B-24
System: Windows 10
Hostname: DESKTOP-RDLUAL9
Scan Duration: 10.32 seconds

[+] Open Ports:
Port       Service
-------------------------
135        epmap
139        netbios-ssn
445        microsoft-ds
5040       unknown
7680       ms-do
49664      unknown
49665      unknown
49666      unknown
49667      unknown
49668      unknown
49669      unknown
```

Reports for 3851019529 → stealer

```
--- Report (Sent: 2025-07-11 15:59:24 UTC, Received: 2025-07-11 15:59:25 UTC)
---
[
  {
    "type": "ssh",
    "file": "known_hosts",
    "content": "|1|Ml2gkyKRa7BPdE4z19BMyY7mV/4=|Me7DtnRwK7SSS0ZePM9h90DUvXs=
ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIJUaenOvxRoQx7g16wSJyiO3vyZIRwk9RmlOFZHxMmP6\n"
  }
]
```

Reports for 3276171044 → keylogger

```
--- Report (Sent: 2025-07-11 15:20:09 UTC, Received: 2025-07-11 15:20:11 UTC)
---
2025-07-11 11:20:03: Key.shift
2025-07-11 11:20:03: E
2025-07-11 11:20:04: N
2025-07-11 11:20:04: C
2025-07-11 11:20:05: O
2025-07-11 11:20:05: D
2025-07-11 11:20:05: E
2025-07-11 11:20:06: R
2025-07-11 11:20:06: S
2025-07-11 11:20:06: P
2025-07-11 11:20:07: R
2025-07-11 11:20:07: O

--- Report (Sent: 2025-07-11 15:19:54 UTC, Received: 2025-07-11 15:19:55 UTC)
---
```

Reports for 3276171044 → bruteforce

```
--- Report (Sent: 2025-07-11 15:25:59 UTC, Received: 2025-07-11 15:2(
---
test:test123

--- Report (Sent: 2025-07-11 15:25:02 UTC, Received: 2025-07-11 15:2!
---
admin:password123
```

Reports for 3276171044 → spyware

```
--- Report (Sent: 2025-07-11 15:22:23 UTC, Received: 2025-07-11 15:22:37 UTC)
---
[✓] Clipboard captured: buggymaytricks
[✓] Screenshot saved at: /tmp/screenshot_2025-07-11_15-22-24_UTC.png
Screenshot URL: http://10.9.0.98:5000/static/uploads/3276171044/
screenshot_2025-07-11_15-22-24_UTC.png

--- Report (Sent: 2025-07-11 15:21:42 UTC, Received: 2025-07-11 15:21:49 UTC)
---
[✓] Clipboard captured: [redacted]
[✓] Screenshot saved at: /tmp/screenshot_2025-07-11_15-21-42_UTC.png
Screenshot URL: http://10.9.0.98:5000/static/uploads/3276171044/
screenshot_2025-07-11_15-21-42_UTC.png
```

Challenges Faced

- Team Coordination & Workflow
- Shift in Architecture (Late Redesign)
- Bot Stability & Functional Gaps
- VM Performance Issues
- Testing Limitations
- Time Constraints

Future Enhancements

- Encrypted Communication
- Bot Hardening and Stealth
- Real-Time Dashboard Enhancements
- Lightweight Bot Deployment
- Advanced Traffic Analysis
- Propagation Simulation

Thank You

Team - अनिर्वचनीय