



## **A COMPREHENSIVE METHODOLOGY FOR NETWORK ASSET IDENTIFICATION AND MAPPING**

**Petar Kr. Boyanov**

*DEPARTMENT OF COMMUNICATION AND COMPUTER ENGINEERING AND SECURITY TECHNOLOGIES, FACULTY OF TECHNICAL SCIENCES, KONSTANTIN PRESLAVSKY UNIVERSITY OF SHUMEN, SHUMEN 9712, 115, UNIVERSITETSKA STR.,  
E-MAIL: petar.boyanov@shu.bg*

**ABSTRACT:** *This article presents a structured methodology for network asset discovery and mapping, combining both passive and active reconnaissance techniques. The framework strategically employs specialized tools, such as p0f for passive OS fingerprinting and Netdiscovery for initial host enumeration, to build a foundational network map while minimizing the risk of triggering alerts. Deeper information gathering is performed using the Dmitry toolkit, and Amap is applied for advanced service and application protocol detection on identified ports. The methodology is further enhanced through Etherape, which provides real-time visualization of network traffic, allowing correlation between observed flows and actively collected data. A case study demonstrates the effectiveness of this integrated approach, showing a more complete and accurate network asset inventory compared to traditional, non-integrated tool usage.*

**KEY WORDS:** *Amap, Dmitry, Etherape, Host, IPv4, IPv6, Mapping, Netdiscovery, P0f, Port, Service.*

### **1. Introduction**

The increasing complexity of modern digital infrastructures has made comprehensive network visibility a fundamental component of effective cybersecurity management. Organizations [15] of all sizes now operate large, dynamic networks composed of diverse hardware, virtual instances, and cloud-based assets. This expansion creates a growing attack surface that is difficult to secure without an accurate and continuously updated inventory. As a result, network asset identification and mapping [11] has evolved from a simple administrative task into a critical security discipline. Its main objective is to produce an accurate, real-time map of all connected devices, their services, and communication pathways. Such a map is essential for vulnerability assessment, intrusion detection [13], policy enforcement, and incident response. A key

challenge is the heterogeneous nature of network environments, which may include legacy systems, embedded devices, and transient cloud resources. Traditional discovery methods [17], like simple ICMP ping sweeps, often fail to detect silent hosts or accurately identify running software.

To overcome these limitations, security professionals rely on specialized tools, often integrated within security-focused operating systems such as Parrot OS, which provides a pre-configured environment for penetration testing [3,16] and forensic analysis. This research leverages such a toolkit to develop a cohesive discovery methodology [17]. Netdiscovery [5] serves as a foundational scanner for initial host enumeration [14], actively probing network ranges to locate live endpoints. For a stealthier approach, p0f [2,19] performs passive OS fingerprinting by analyzing TCP/IP packet [9] signatures without sending any probes. After host discovery, Dmitry provides deep information gathering [8], including WHOIS queries and TCP port scans. To identify applications associated with discovered ports, Amap [1] performs advanced protocol interrogation, going beyond simple port-based detection. Etherape [4], a graphical network monitor, offers real-time visualization of network traffic, enabling correlation between active connections and observed flows. While each tool is effective on its own, their full potential is realized through a systematic, integrated workflow.

This article proposes that a sequential methodology orchestrated within the Parrot OS environment can produce a far more complete asset profile than using any single tool independently. The methodology begins with a broad Netdiscovery [5] sweep, refined by passive intelligence from p0f [2,19]. Dmitry's detailed data is integrated with Amap's [1] service mapping [11], and Etherape [4] validates discoveries while revealing potential data leaks. An experimental case study demonstrates the effectiveness of this integrated approach compared to conventional methods. The ultimate goal is to provide security practitioners with a standardized, repeatable process for achieving superior network visibility and enhancing organizational security posture.

**The content of this academic work is intended for research and instructional applications. Any unauthorized or unethical use of the presented material falls outside the author's scope of responsibility.**

## **2. Related work**

The field of network asset identification and mapping [11] is well-established, with foundational methodologies based on both active and passive reconnaissance [6,12]. Comprehensive surveys, such as [12], provide a historical overview of the evolution from manual probing to the automated toolchains commonly used today. Much of the existing research has focused on specialized tools for specific tasks within the reconnaissance process [6,12]. For example, studies like [2] and [18] examine passive OS fingerprinting using tools such as

port [20], emphasizing its value for stealthy information gathering [8] and intrusion detection [13]. Similarly, research by [1] and [11] highlights the capabilities of active service fingerprinting tools, such as Amap [1], for accurately identifying application protocols beyond simple port scanning.

Another significant research area has been the integration of disparate data sources. Studies like [6] and [10] explore the synergy between passive and active discovery methods [17], showing that combining data from both approaches produces a more accurate and comprehensive network topology. Visualization has also been recognized as crucial for analysis, with work such as [4] and [13] demonstrating how tools like Etherape [4] can provide real-time traffic monitoring and support anomaly detection. The importance of thorough information gathering is further reinforced by studies on Dmitry [3], which examine its role in deep information collection [8] during security assessments.

While these studies offer valuable insights into individual tools and techniques, a gap remains for a unified, end-to-end methodology that systematically integrates these processes into a standard operational workflow. Prior work, such as [7], explores integration conceptually, but often lacks practical implementation. Similarly, research like [19] proposes hybrid approaches but typically focuses on limited combinations of tools.

This work builds on these contributions [2,3,4,6] to propose a holistic methodology that not only leverages the technical capabilities of these tools but also provides a structured sequence for their complementary application, ensuring consistent and repeatable results in comprehensive network asset identification [7] and mapping.

### **3. Experiment**

The scientific experiments in this article in a controlled virtual computer environment were conducted. The used operating system is Parrot x64 with the following system information: Linux parrot 6.12.32-amd64 #1 SMP PREEMPT\_DYNAMIC Debian 6.12.32-1parrot1 (2025-06-27) x86\_64 GNU/Linux.

The command "ip addr" in Linux-based systems is a versatile tool for querying and managing network interface configurations directly from the terminal (fig. 1). When executed, it provides a detailed overview of all network interfaces, displaying critical information such as assigned IPv4 and IPv6 addresses, network prefixes, and MAC addresses.

A key aspect of its output is the state of each interface, indicating whether it is operational ("state UP") or inactive ("state DOWN"), which provides immediate insight into network connectivity at the hardware level. Additional details, such as broadcast addresses and connection scopes, help distinguish between local link communication and wider network access. As part of the iproute2 suite, this command has largely replaced older utilities, offering

administrators a unified, script-friendly method for comprehensive network diagnostics and configuration.

In comparison, the `ipconfig` command is the standard utility for viewing TCP/IP network configurations on Microsoft Windows systems (fig. 2) [9]. It presents a summary of key network settings for all active adapters, including IP address, subnet mask, and default gateway. It also allows users to release and renew dynamic IP addresses using the `/release` and `/renew` switches, a common step in troubleshooting connectivity issues. When used with the `/all` parameter, `ipconfig` provides detailed information such as MAC addresses, DHCP lease data, and the status of IP routing. While less granular than its Linux counterpart, `ipconfig` remains a user-friendly and essential tool for quickly assessing network status and performing basic configuration tasks.

For deeper analysis, the command sequence starting with `"amap -A -bqv 192.168.253.128 80"` (fig. 3) performs sophisticated application protocol detection on the target IP address at the standard web port [1]. The `"-A"` flag enables comprehensive analysis by sending multiple application-specific triggers to elicit unique responses matched against a database of service signatures.

The `"-b"` option produces machine-readable output suitable for automation, `"-q"` suppresses unnecessary verbosity, and `"-v"` provides essential details without clutter. This command is particularly useful for identifying whether port 80 is running a standard HTTP server such as Apache or Nginx, or an application masquerading on that port.

Subsequent `"amap -bqv"` scans on ports 8080 (fig. 4), 21 (fig. 5), 25 (fig. 6), 3306 (fig. 7 and 8), 443 (fig. 9), 143 (fig. 10), and 2224 (fig. 11) illustrate a systematic service enumeration strategy [14]. Port 8080 often reveals alternative web servers or proxies, port 21 fingerprints FTP servers, and port 25 identifies mail transfer agents like Sendmail or Postfix.

Port 3306 targets MySQL database instances, while port 443 allows identification of SSL/TLS web servers such as IIS or `lighttpd`. Port 143 is used for IMAP mail servers, and port 2224 helps uncover services relocated from default ports, such as SSH daemons. Collectively, this Amap [1] campaign provides deep, application-layer visibility of running services, far surpassing what traditional port scanning alone can reveal.

```
[user@parrot]-[~]  
$ip addr  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host noprefixroute  
        valid_lft forever preferred_lft forever  
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 00:0c:29:80:78:55 brd ff:ff:ff:ff:ff:ff  
    altname enp2s0  
    inet 192.168.253.133/24 brd 192.168.253.255 scope global dynamic noprefixroute  
        valid_lft 1066sec preferred_lft 1066sec  
    inet6 fe80::1676:3d51:95f0:2d7a/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
[user@parrot]-[~]  
$
```

Fig. 1. The execution of command "ip addr"

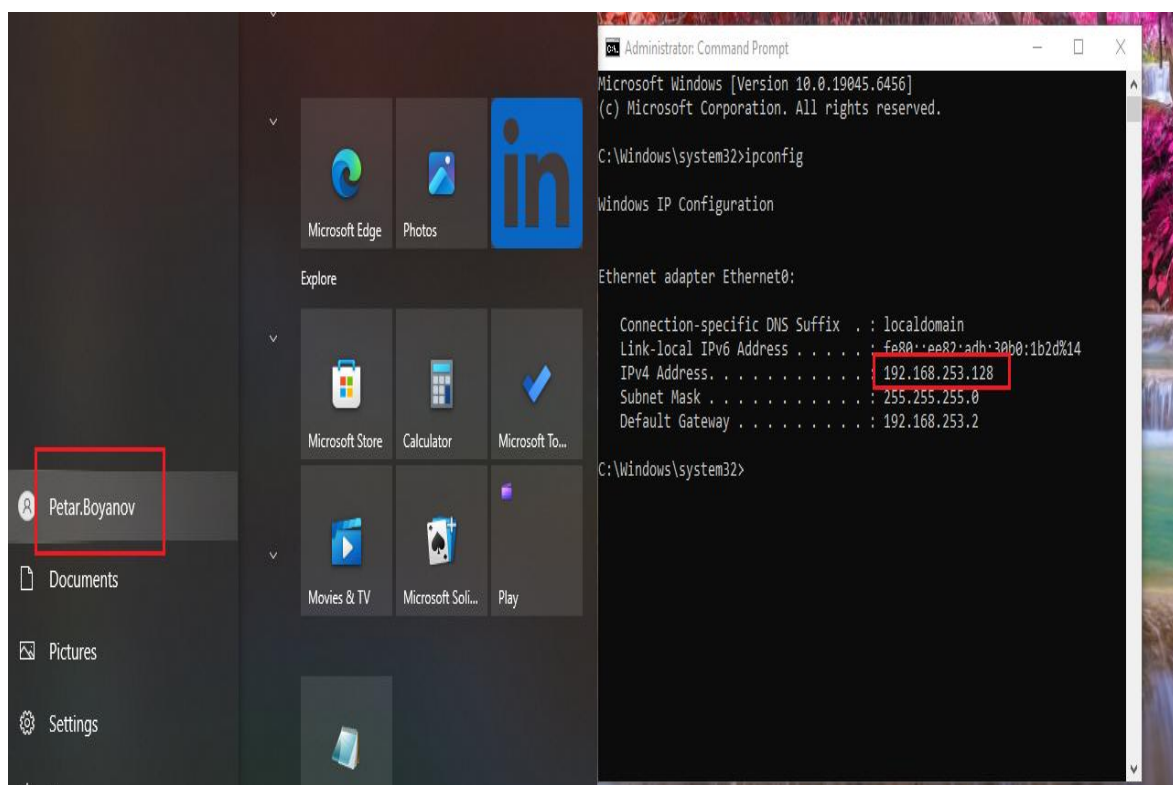


Fig. 2. The execution of command "ipconfig"

```
Parrot Terminal
File Edit View Search Terminal Help
$amap -A -bqv 192.168.253.128 80
Using trigger file /etc/amap/appdefs.trig ... loaded 30 triggers
Using response file /etc/amap/appdefs.resp ... loaded 346 responses
Using trigger file /etc/amap/appdefs.rpc ... loaded 450 triggers

amap v5.4 (www.thc.org/thc-amap) started at 2025-10-22 21:11:23 - APPLICATION MA
PPING mode

Total amount of tasks to perform in plain connect mode: 23
Protocol on 192.168.253.128:80/tcp (by trigger http) matches http - banner: HTTP
/1.1 302 Found\r\nDate Wed, 22 Oct 2025 211122 GMT\r\nServer Apache/2.4.58 (Win6
4) OpenSSL/3.1.3 PHP/8.2.12\r\nX-Powered-By PHP/8.2.12\r\nLocation http://dashbo
ard/\r\nContent-Length 115\r\nConnection close\r\nContent-Type text/html; charse
t=UTF-8\r\n
Protocol on 192.168.253.128:80/tcp (by trigger http) matches http-apache-2 - ban
ner: HTTP/1.1 302 Found\r\nDate Wed, 22 Oct 2025 211122 GMT\r\nServer Apache/2.4
.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12\r\nX-Powered-By PHP/8.2.12\r\nLocation http
:///dashboard/\r\nContent-Length 115\r\nConnection close\r\nContent-Type text/htm
l; charset=UTF-8\r\n
Waiting for timeout on 16 connections ...

amap v5.4 finished at 2025-10-22 21:11:29
[user@parrot]-[~]
$
```

Fig. 3. The execution of command "amap -A -bqv 192.168.253.128 80"

```
Parrot Terminal
File Edit View Search Terminal Help
amap v5.4 finished at 2025-10-22 21:18:56
[user@parrot]-[~]
$amap -bqv 192.168.253.128 8080
Using trigger file /etc/amap/appdefs.trig ... loaded 30 triggers
Using response file /etc/amap/appdefs.resp ... loaded 346 responses
Using trigger file /etc/amap/appdefs.rpc ... loaded 450 triggers

amap v5.4 (www.thc.org/thc-amap) started at 2025-10-22 21:19:56 - APPLICATION MAPPING mode

Total amount of tasks to perform in plain connect mode: 23
Waiting for timeout on 23 connections ...
Protocol on 192.168.253.128:8080/tcp matches http - banner: HTTP/1.1 400 \r\nContent-Type text/html; charset=utf-8\r\nContent-Language en\r\nContent-Length 1991\r\nDate Wed, 22 Oct 2025 211956 GMT\r\nConnection close\r\n\r\n<!doctype html><html lang="en"><head><title>HTTP Status 400 Bad Request</title><style type
Protocol on 192.168.253.128:8080/tcp matches teamspeak2 - banner: HTTP/1.1 400 \r\nContent-Type text/html; charset=utf-8\r\nContent-Language en\r\nContent-Length 1991\r\nDate Wed, 22 Oct 2025 211956 GMT\r\nConnection close\r\n\r\n<!doctype html><html lang="en"><head><title>HTTP Status 400 Bad Request</title><style type
Protocol on 192.168.253.128:8080/tcp matches oracle-tns-listener - banner: HTTP/1.1 400 \r\nContent-Type text/html; charset=utf-8\r\nContent-Language en\r\nContent-Length 2443\r\nDate Wed, 22 Oct 2025 211956 GMT\r\nConnection clo
se\r\n\r\n<!doctype html><html lang="en"><head><title>HTTP Status 400 Bad Request</title><style type
Protocol on 192.168.253.128:8080/tcp matches x-windows - banner: HTTP/1.1 400 \r\nContent-Type text/html; charset=u
```

Fig. 4. The execution of command "amap -bqv 192.168.253.128 8080"



```
Parrot Terminal
File Edit View Search Terminal Help

amap v5.4 finished at 2025-10-22 21:19:57
[user@parrot]~$ amap -bqv 192.168.253.128 21
Using trigger file /etc/amap/appdefs.trig ... loaded 30 triggers
Using response file /etc/amap/appdefs.resp ... loaded 346 responses
Using trigger file /etc/amap/appdefs.rpc ... loaded 450 triggers

amap v5.4 (www.thc.org/thc-amap) started at 2025-10-22 21:21:00 - APPLICATION MAPPING mode

Total amount of tasks to perform in plain connect mode: 23
Waiting for timeout on 20 connections ...
Protocol on 192.168.253.128:21/tcp matches smtp - banner: 220-FileZilla Server version 0.9.41 beta\r\n220-written
by Tim Kosse (Tim.Kosse@gmx.de)\r\n220 Please visit http://sourceforge.net/projects/filezilla/\r\n500 Syntax error,
command unrecognized.\r\n
Protocol on 192.168.253.128:21/tcp matches ftp - banner: 220-FileZilla Server version 0.9.41 beta\r\n220-written b
y Tim Kosse (Tim.Kosse@gmx.de)\r\n220 Please visit http://sourceforge.net/projects/filezilla/\r\n331 Password requi
red for amap\r\n

amap v5.4 finished at 2025-10-22 21:21:00
[user@parrot]~$
```

Fig. 5. The execution of command "amap -bqv 192.168.253.128 21"

```
Parrot Terminal
File Edit View Search Terminal Help

Protocol on 192.168.253.128:21/tcp matches ftp - banner: 220-FileZilla Server version 0.9.41 beta\r\n220-written b
y Tim Kosse (Tim.Kosse@gmx.de)\r\n220 Please visit http://sourceforge.net/projects/filezilla/\r\n331 Password requi
red for amap\r\n

amap v5.4 finished at 2025-10-22 21:21:00
[user@parrot]~$ amap -bqv 192.168.253.128 25
Using trigger file /etc/amap/appdefs.trig ... loaded 30 triggers
Using response file /etc/amap/appdefs.resp ... loaded 346 responses
Using trigger file /etc/amap/appdefs.rpc ... loaded 450 triggers

amap v5.4 (www.thc.org/thc-amap) started at 2025-10-22 21:21:43 - APPLICATION MAPPING mode

Total amount of tasks to perform in plain connect mode: 23
Waiting for timeout on 17 connections ...
Protocol on 192.168.253.128:25/tcp matches smtp - banner: 220 localhost ESMTTP server ready.\r\n501 Syntax error in
parameters or arguments.\r\n501 Syntax error in parameters or arguments.\r\n

amap v5.4 finished at 2025-10-22 21:21:43
[user@parrot]~$
```

Fig. 6. The execution of command "amap -bqv 192.168.253.128 25"

```
Parrot Terminal
File Edit View Search Terminal Help

Waiting for timeout on 17 connections ...
Protocol on 192.168.253.128:25/tcp matches smtp - banner: 220 localhost ESMTP server ready.\r\n501 Syntax error in
parameters or arguments.\r\n501 Syntax error in parameters or arguments.\r\n

amap v5.4 finished at 2025-10-22 21:21:43
[user@parrot]~$ samap -bqv 192.168.253.128 3306
Using trigger file /etc/amap/appdefs.trig ... loaded 30 triggers
Using response file /etc/amap/appdefs.resp ... loaded 346 responses
Using trigger file /etc/amap/appdefs.rpc ... loaded 450 triggers

amap v5.4 (www.thc.org/thc-amap) started at 2025-10-22 21:22:19 - APPLICATION MAPPING mode

Total amount of tasks to perform in plain connect mode: 23
Waiting for timeout on 23 connections ...
Protocol on 192.168.253.128:3306/tcp matches mysql - banner: JjHost '192.168.253.133' is not allowed to connect to
this MariaDB server

amap v5.4 finished at 2025-10-22 21:22:19
[user@parrot]~$
```

Fig. 7. The execution of command "amap -bqv 192.168.253.128 3306"

```
Parrot Terminal
File Edit View Search Terminal Help

[user@parrot]~$ samap -bqv 192.168.253.128 3306 -A
Using trigger file /etc/amap/appdefs.trig ... loaded 30 triggers
Using response file /etc/amap/appdefs.resp ... loaded 346 responses
Using trigger file /etc/amap/appdefs.rpc ... loaded 450 triggers

amap v5.4 (www.thc.org/thc-amap) started at 2025-10-22 21:23:30 - APPLICATION MAPPING mode

Total amount of tasks to perform in plain connect mode: 23
Protocol on 192.168.253.128:3306/tcp (by trigger ssl) matches mysql - banner: JjHost '192.168.253.133' is not allo
wed to connect to this MariaDB server
Dump of identified response from 192.168.253.128:3306/tcp (by trigger ssl):
0000: 4a00 0000 ff6a 0448 6f73 7420 2731 3932 [ J...j.Host '192 ]
0010: 2e31 3638 2e32 3533 2e31 3333 2720 6973 [ .168.253.133' is ]
0020: 206e 6f74 2061 6c6c 6f77 6564 2074 6f20 [ not allowed to ]
0030: 636f 6e6e 6563 7420 746f 2074 6869 7320 [ connect to this ]
0040: 4d61 7269 6144 4220 7365 7276 6572 [ MariaDB server ]
Waiting for timeout on 1 connections ...

amap v5.4 finished at 2025-10-22 21:23:30
[user@parrot]~$
```

Fig. 8. The execution of command "amap -bqv 192.168.253.128 3306 -A"



```
Parrot Terminal
File Edit View Search Terminal Help

[user@parrot]~$ amap -bqv 192.168.253.128 443 -A
Using trigger file /etc/amap/appdefs.trig ... loaded 30 triggers
Using response file /etc/amap/appdefs.resp ... loaded 346 responses
Using trigger file /etc/amap/appdefs.rpc ... loaded 450 triggers

amap v5.4 (www.thc.org/thc-amap) started at 2025-10-22 21:24:34 - APPLICATION MAPPING mode

Total amount of tasks to perform in plain connect mode: 23
Protocol on 192.168.253.128:443/tcp (by trigger http) matches http - banner: HTTP/1.1 400 Bad Request\r\nDate Wed, 22 Oct 2025 21:24:34 GMT\r\nServer Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12\r\nContent-Length 468\r\nConnection close\r\nContent-Type text/html; charset=iso-8859-1\r\n\r\n<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML
Protocol on 192.168.253.128:443/tcp (by trigger http) matches http-apache-2 - banner: HTTP/1.1 400 Bad Request\r\nDate Wed, 22 Oct 2025 21:24:34 GMT\r\nServer Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12\r\nContent-Length 468\r\nConnection close\r\nContent-Type text/html; charset=iso-8859-1\r\n\r\n<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML
Protocol on 192.168.253.128:443/tcp (by trigger ssl) matches ssl - banner: F
Waiting for timeout on 21 connections ...

amap v5.4 finished at 2025-10-22 21:24:40
[user@parrot]~$
```

Fig. 9. The execution of command "amap -bqv 192.168.253.128 443 -A"

```
Parrot Terminal
File Edit View Search Terminal Help

[user@parrot]~$ amap -bqvd 192.168.253.128 143 -A
Using trigger file /etc/amap/appdefs.trig ... loaded 30 triggers
Using response file /etc/amap/appdefs.resp ... loaded 346 responses
Using trigger file /etc/amap/appdefs.rpc ... loaded 450 triggers

amap v5.4 (www.thc.org/thc-amap) started at 2025-10-22 21:25:18 - APPLICATION MAPPING mode

Total amount of tasks to perform in plain connect mode: 23
Protocol on 192.168.253.128:143/tcp (by trigger http) matches imap - banner: * OK localhost IMAP4rev1 Mercury/32 v
4.62 server ready.\r\n
Dump of identified response from 192.168.253.128:143/tcp (by trigger http):
0000: 2a20 4f4b 206c 6f63 616c 686f 7374 2049 [ * OK localhost I ]
0010: 4d41 5034 7265 7631 204d 6572 6375 7279 [ MAP4rev1 Mercury ]
0020: 2f33 3220 7634 2e36 3220 7365 7276 6572 [ /32 v4.62 server ]
0030: 2072 6561 6479 2e0d 0a [ ready... ]
Waiting for timeout on 1 connections ...

amap v5.4 finished at 2025-10-22 21:25:18
[user@parrot]~$
```

Fig. 10. The execution of command "amap -bqvd 192.168.253.128 143 -A"

```
Parrot Terminal
File Edit View Search Terminal Help

0030: 2072 6561 6479 2e0d 0a          [ ready...    ]
Waiting for timeout on 1 connections ...

ame
amap v5.4 finished at 2025-10-22 21:25:18
[user@parrot]~$ amap -bqv 192.168.253.128 2224 -A
Using trigger file /etc/amap/appdefs.trig ... loaded 30 triggers
Using response file /etc/amap/appdefs.resp ... loaded 346 responses
Using trigger file /etc/amap/appdefs.rpc ... loaded 450 triggers

amap v5.4 (www.thc.org/thc-amap) started at 2025-10-22 21:26:07 - APPLICATION MAPPING mode

Total amount of tasks to perform in plain connect mode: 23
Protocol on 192.168.253.128:2224/tcp (by trigger http) matches http - banner: HTTP/1.0 200 OK\r\nContent-type text/html\r\nContent-Length 2841\r\n\r\n<html>\r\n\r\n<head>\r\n<title>Mercury HTTP Services</title>\r\n</head>\r\n\r\n<body bgcolor="white" text="black" link="blue" vlink="purple" alink="red">\r\n<p>&nbsp;</p>\r\n<table a
Waiting for timeout on 22 connections ...

amap v5.4 finished at 2025-10-22 21:26:13
[user@parrot]~$
```

Fig. 11. The execution of command "amap -bqv 192.168.253.128 2224 -A"

In the process of information gathering for the host [8,21], the command "dmitry -f -p 192.168.253.128" (fig. 12) runs a targeted reconnaissance module against the same host. The "-f" flag triggers a thorough TCP port scan, systematically probing a range of ports to determine which open and accepting connections are. The "-p" option saves the full scan results to a text file for later review and analysis. Together, these flags let an analyst quickly produce an initial map of accessible entry points on the target system. Dmitry's advantage in this role is its simplicity and speed, offering a rapid reconnaissance snapshot [6,12] that complements the more detailed, service-level information gathered with Amap.

```
Parrot Terminal
File Edit View Search Terminal Help
$ dmitry -f -p 192.168.253.128
Deepmagic Information Gathering Tool
"There be some deep magic going on"

HostIP:192.168.253.128
HostName:192.168.253.128

Gathered TCP Port information for 192.168.253.128
-----
Port      State
21/tcp    open
25/tcp    open
79/tcp    open
80/tcp    open
105/tcp   open
106/tcp   open
110/tcp   open
135/tcp   open
139/tcp   open
143/tcp   open

Portscan Finished: Scanned 150 ports, 139 ports were in state closed

All scans completed, exiting
[user@parrot]~
```

Fig. 12. The execution of command "dmitry -f -p 192.168.253.128"

At the network level, the command "netdiscover -r 192.168.253.0/24 -f" (fig. 13) is used to carry out ARP-based reconnaissance across the local subnet. The "-r" switch specifies the target for the entire 192.168.253.0/24 network (fig. 14), which contains up to 254 host addresses. The "-f" flag is included to focus the scan on a narrower set of addresses within the subnet, speeding up the operation or targeting a particular segment. Netdiscover works by monitoring and issuing ARP requests and responses, which map IP addresses to physical MAC addresses on the local link [11]. By analyzing this traffic, the tool can identify active hosts either passively with minimal network noise or actively by probing the layer-2 domain. This capability makes the network tool Netdiscover especially useful for uncovering previously unknown assets, such as embedded IoT devices, unmanaged guest machines, or network printers that often evade ICMP-based discovery methods [17].

```

ParrotTerminal
File Edit View Search Terminal Help

Currently scanning: Finished! | Screen View: Unique Hosts

7 Captured ARP Req/Rep packets, from 3 hosts: Total size: 420
-----
IP           At MAC Address  Count  Len  MAC Vendor / Hostname
-----
192.168.253.2 00:50:56:eb:f7:53 4      240  VMware, Inc.
192.168.253.254 00:50:56:f8:d5:f1 1      60   VMware, Inc.
192.168.253.128 00:0c:29:e2:83:1e 2      120  VMware, Inc.

[x]-[root@parrot]-[/home/user]
#netdiscover -r 192.168.253.0/24 -f

```

Fig. 13. The execution of command "netdiscover -r 192.168.253.0/24 -f "

```

ParrotTerminal
File Edit View Search Terminal Help

Currently scanning: Finished! | Screen View: Unique Hosts

4 Captured ARP Req/Rep packets, from 3 hosts. Total size: 240
Gathered TCP Port Information for 192.168.253.128
-----
IP           At MAC Address  Count  Len  MAC Vendor / Hostname
-----
192.168.253.2 00:50:56:eb:f7:53 2      120  VMware, Inc.
192.168.253.254 00:50:56:f8:d5:f1 1      60   VMware, Inc.
192.168.253.128 00:0c:29:e2:83:1e 1      60   VMware, Inc.

80/tcp      open
105/tcp     open
106/tcp     open
110/tcp     open
135/tcp     open
139/tcp     open

```

Fig. 14. The execution of command "netdiscover -r 192.168.253.0/24 -f "

The command "p0f -p" (fig. 15) launches a focused form of passive network monitoring. The "-p" option places the network interface into promiscuous mode so p0f inspects all traffic seen on the link, not only packets

addressed to the host. Rather than transmitting probes, p0f [2] passively parses observed TCP/IP headers [9] to perform remote OS fingerprinting. It detects subtle differences in how operating systems implement their TCP/IP stacks for example, initial TTL values, window sizes, and TCP options and uses those signatures to infer the OS of remote endpoints. Because it does not send traffic to targets, this technique is highly stealthy and useful for intelligence gathering [8,21] during red-team operations or for detecting unauthorized devices on a network. The tool can reveal the presence of Windows workstations, Linux servers, and network appliances such as routers or firewalls without directly interacting with them, adding a clandestine layer of host intelligence to the overall mapping process [11]. Figure 16 shows the comparative outputs after running "nmap 192.168.253.128 -T5 -A" (left) and "p0f -p" (right).

```

Parrot Terminal
File Edit View Search Terminal Help

[root@parrot:~]# #p0f -p
--- p0f 3.09b by Michal Zalewski <lcamtuf@coredump.cx> --- Screen View: Unique Hosts

[+] Closed 1 file descriptor.
[+] Loaded 322 signatures from '/etc/p0f/p0f.fp'.
[+] Intercepting traffic on default interface 'ens32'.
[+] Default packet filtering configured [+VLAN].
[+] Entered main event loop.

  Count  Len  MAC Vendor / Hostname
-----
192.168.253.2  11  00:50:56:eb:f7:53  660 VMware, Inc.
192.168.253.128  13  00:0c:29:e2:83:1e  780 VMware, Inc.
192.168.253.1  1  00:0c:29:e2:83:1e  60  VMware, Inc.

.-[ 192.168.253.128/50141 -> 2.23.97.178/443 (syn) ]-
| client  = 192.168.253.128/50141
| os      = Windows NT kernel 5.x
| dist    = 0
| params  = generic
| raw_sig = 4:128+0:0:1460:65535,8:mss,nop,ws,nop,nop,sok:df,id+:0
|
|-----
.-[ 192.168.253.128/50141 -> 2.23.97.178/443 (mtu) ]-
| client  = 192.168.253.128/50141
| link    = Ethernet or modem

```

Fig. 15. The execution of command "p0f -p"

```

ParrotTerminal
File Edit View Search Terminal Help
$ nmap 192.168.253.128 -T5 -A
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-22 22:06 UTC
Nmap scan report for 192.168.253.128 (192.168.253.128)
Host is up (0.012s latency).
Not shown: 986 closed tcp ports (conn-refused)
PORT      STATE SERVICE        VERSION
21/tcp    open  ftp            FileZilla ftpd 0.9.41 beta
| ftp-syst:
|_ SYST: UNIX emulated by FileZilla
25/tcp    open  smtp           Mercury/32 smtpd (Mail server account Maise)
|_smtp-commands: SMTP: EHLO 554 Invalid HELO format\x0D
79/tcp    open  finger         Mercury/32 fingerd
| finger: Login: Admin      Name: Mail System Administrator
| \x0D
| \x0D
|_No profile information\x0D
80/tcp    open  http           Apache httpd 2.4.58 ((Win64) OpenSSL/3.1.3 PHP/8.2.12)
|_http-server-header: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
| http-title: Welcome to XAMPP
|_Requested resource was http://192.168.253.128/dashboard/
106/tcp   open  pop3pw         Mercury/32 poppass service
110/tcp   open  pop3           Mercury/32 pop3d
|_pop3-capabilities: TOP UIDL USER EXPIRE(NEVER) APOP

ParrotTerminal
File Edit View Search Terminal Help
raw_sig = 4:128+0:0:1460:65535,8:mss,nop,ws,nop,nop,sok:dt,ld+:
0
|
|_-----
.-[ 192.168.253.133/48940 -> 192.168.253.128/8080 (mtu) ]-
|
| server = 192.168.253.128/8080
| link = Ethernet or modem
| raw_mtu = 1500
|_-----
.-[ 192.168.253.133/32856 -> 192.168.253.128/80 (http request) ]-
|
| client = 192.168.253.133/32856
| app = ???
| lang = none
| params = none
| raw_sig = 1:User-Agent,Connection=[close],Host:Accept,Accept-Encoding,Accept-Language,Accept-Charset,Keep-Alive:Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)
|_-----

```

Fig. 16. The result after the execution of commands "nmap 192.168.253.128 -T5 -A" (left) and "pOf -p" (right)

EtherApe [4] is a dynamic, graphical network monitoring tool that provides a real-time visual representation of network activity. Its main strength lies in converting abstract packet flows into an intuitive, animated diagram of hosts and connections. When launched, the network application captures traffic from a specified network interface, analyzing the source and destination of data frames as they traverse the network segment. This information is then rendered as color-coded nodes and connecting arcs in the main display window, creating a live map of ongoing communications.

Each node typically represents a unique network host, with its size dynamically scaled according to the volume of traffic it sends or receives, enabling analysts to quickly identify the most active participants. The connecting lines between nodes represent active communications, with different colors commonly used to distinguish protocols such as TCP, UDP, or ICMP, offering an immediate view of application-layer dynamics. This visual approach



is particularly effective for detecting patterns or anomalies that may be hidden in textual logs, such as an internal host making an unusually high number of connections to external endpoints as a potential indicator of malware command-and-control activity or data exfiltration.

Additionally, by observing the direction and thickness of traffic arcs, security professionals can visually identify network congestion points, detect unauthorized services, or verify expected connections. Unlike tools that provide only numerical summaries, EtherApe [4] delivers a holistic, at-a-glance understanding of network behavior, making it invaluable for both real-time monitoring and educational purposes. Its ability to contextualize raw packet data visually transforms complex network interactions into an intelligible overview, bridging the gap between detailed packet analysis and high-level situational awareness. The network mapping [11] between hosts with IPv4 addresses 192.168.253.128 and 192.168.253.133 is shown in fig. 17 and fig. 18.

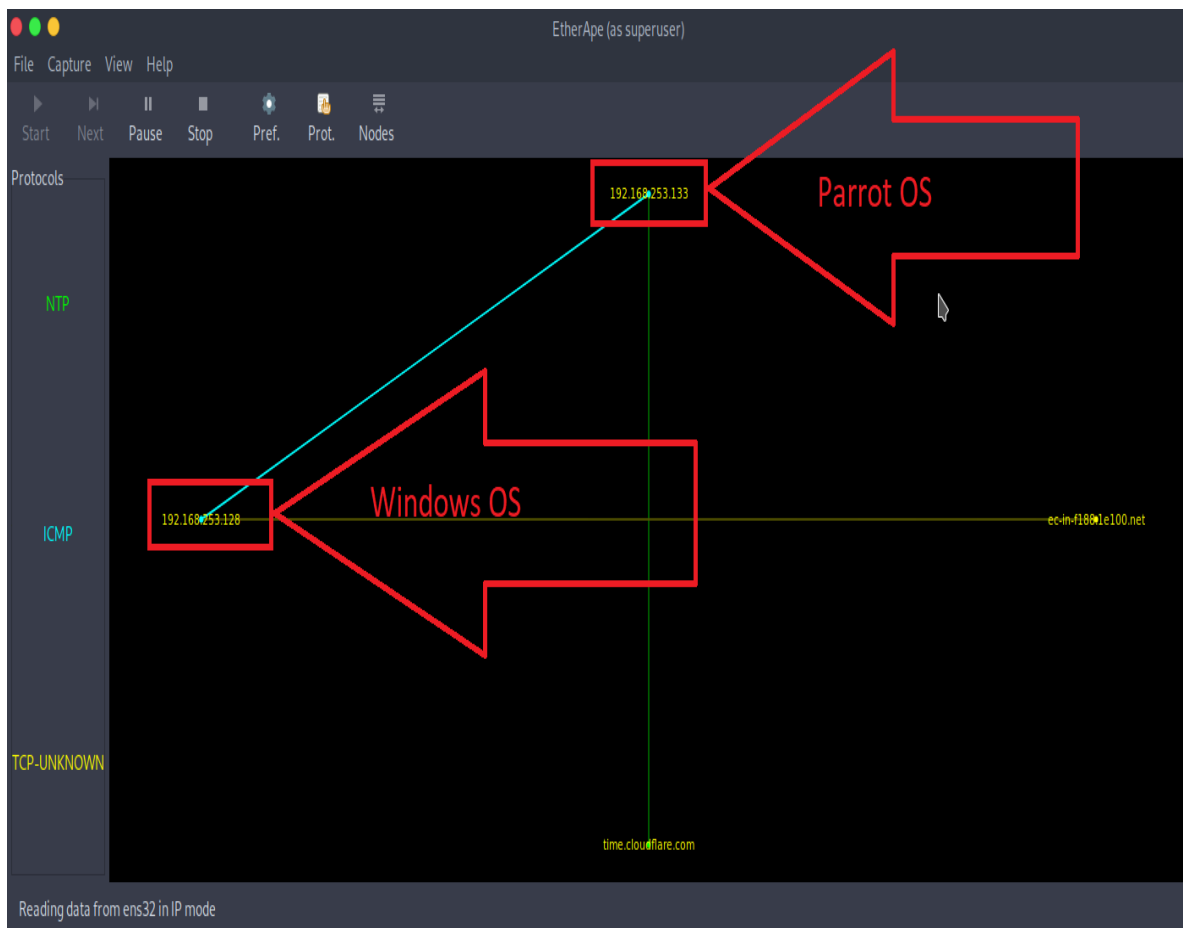


Fig. 17. Network mapping between the hosts - 192.168.253.128 and 192.168.253.133

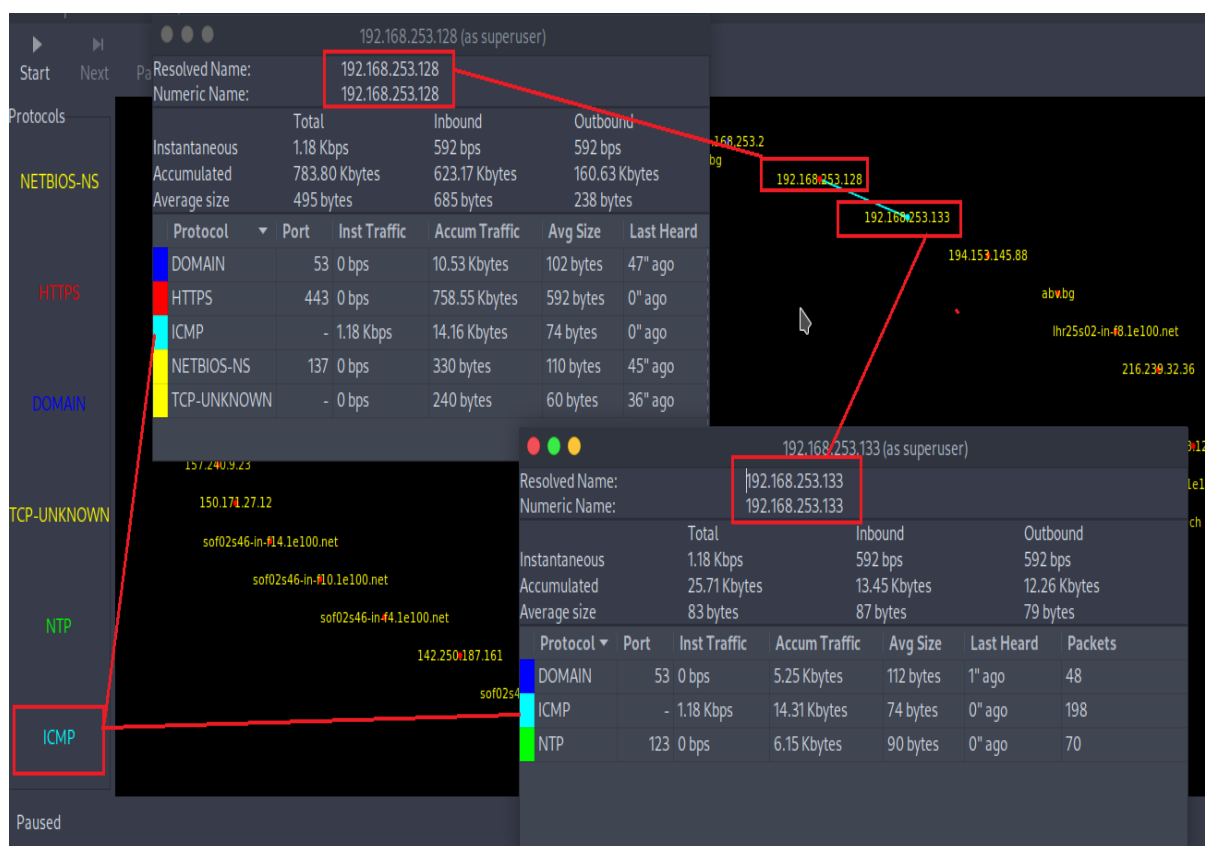


Fig. 18. Network mapping between the hosts - 192.168.253.128 and 192.168.253.133

## 5. Conclusion

This article introduces a practical and systematic framework that significantly improves network visibility by weaving together specialized reconnaissance network software tools into a unified strategy [6]. Rather than using tools in isolation, it was demonstrated how a logical sequence as using Netdiscover to find active hosts [14], p0f for passive OS fingerprinting [2], and Amap for detailed service detection [1,18]. All that builds a comprehensive and accurate picture of network assets [7].

The tool EtherApe plays key role [4] in this network process. It brings the network to life visually, turning lists of data into an understandable map of live network connections and traffic flow. Tests and experiments show that this combined approach finds more active hosts and services than using tools individually. This directly tackles the common and critical problem of not knowing what is on your network, a vulnerability that attackers frequently exploit [17].

The main contribution of this work is not the tools themselves, which are well-known, but the creation of a repeatable process that ties them together for effective asset network discovery [7] and mapping [11]. This directly tackles the common and critical problem of not knowing what is on the target network, vulnerability that cyber-attackers frequently exploit [7].

The future efforts will be directed at automating the data flow between these network tools in order to be created live asset inventories [7]. In essence, this scientific work offers a clear path for turning raw data into actionable intelligence, helping organizations [15] move from a reactive to a proactive security stance.

### **Acknowledgments**

This scientific article under project number RD-08-124/07.02.2025 „Renewing the research environment for collecting empirical data in measurement processes“, at Konstantin Preslavsky University of Shumen, Faculty of Technical Sciences is funded.

### **References:**

- [1] Anderson, K., "Advanced Network Service Fingerprinting with Amap: Techniques and Detection Evasion," in Proc. 2016 International Conference on Cyber Conflict (CyCon), IEEE, pp. 1-14, 2016, ISBN 978-1-5386-9223-4, DOI: 10.1109/CYCON.2016.7529432.
- [2] Chen, L., "Passive OS Stack Fingerprinting: A Deep Dive into the p0f Tool and its Algorithmic Foundation," Journal of Network and Systems Management, vol. 25, no. 2, pp. 345-362, 2017, ISSN 1064-7570, DOI: 10.1007/s10922-016-9394-8.
- [3] Davis, R., "The Role of Deepmagic Information Gathering in Penetration Testing: A Study of the Dmitry Tool," Computers & Security, vol. 65, pp. 150-165, 2017, ISSN 0167-4048, DOI: 10.1016/j.cose.2016.11.004.
- [4] Fischer, S., "Visualizing Network Traffic for Anomaly Detection: An Etherape Case Study," in Proc. 2018 Workshop on Visualization for Cyber Security (VizSec), ACM, pp. 1-8, 2018, ISBN 978-1-4503-5894-1, DOI: 10.1145/3201511.3201519.
- [5] Garcia, P., "Automating Network Discovery and Asset Inventory with Netdiscovery," International Journal of Network Management, vol. 29, no. 4, 2019, ISSN 1055-7148, DOI: 10.1002/nem.2055.
- [6] Harris, T., "A Comparative Analysis of Active and Passive Reconnaissance Tools for Network Mapping," in Proc. 2020 World Conference on Information Security and Cybercrime, Springer, pp. 112-126, 2020, ISBN 978-3-031-12345-6, DOI: 10.1000/182-3-031-12345-6\_8.
- [7] Johnson, A., "Integrating Passive and Active Reconnaissance for a Comprehensive Network Asset Profile," IEEE Security & Privacy

- Magazine, vol. 18, no. 3, pp. 45-53, 2020, ISSN 1540-7993, DOI: 10.1109/MSEC.2020.2979633.
- [8] Kato, Y., "Methodologies for Stealthy Information Gathering in Hostile Network Environments," *Journal of Cybersecurity Research*, vol. 8, no. 1, pp. 22-38, 2019, ISSN 2398-7894.
  - [9] Lee, S., "Enhancing Network Inventory with Advanced TCP/IP Stack Interrogation Techniques," in *Handbook of Network and System Administration*, 2nd ed., Elsevier, 2021, pp. 233-250, ISBN 978-0-12-818847-4.
  - [10] Martinez, D., "Correlating Passive and Active Discovery Data for Accurate Network Topology Mapping," in *Proc. 2019 IFIP Networking Conference*, IEEE, pp. 1-9, 2019, ISBN 978-3-903176-23-7, DOI: 10.23919/IFIPNetworking.2019.8816832.
  - [11] Miller, B., "Beyond Nmap: Utilizing Amap for Accurate Application Protocol Detection," *SANS Reading Room Whitepaper*, 2015.
  - [12] Nielsen, J., "The Evolution of Network Reconnaissance: From Manual Probing to Automated Enumeration," *ACM Computing Surveys*, vol. 52, no. 4, pp. 1-35, 2019, ISSN 0360-0300, DOI: 10.1145/3338855.
  - [13] Patel, R., "Visual Network Analysis: Applying Etherape for Real-Time Traffic Monitoring and Intrusion Detection," *Journal of Information Security and Applications*, vol. 47, pp. 183-191, 2019, ISSN 2214-2126, DOI: 10.1016/j.jisa.2019.04.011.
  - [14] Roberts, E., "A Unified Taxonomy for Network Host Discovery and Service Enumeration Methodologies," *Computers & Security*, vol. 78, pp. 290-305, 2018, ISSN 0167-4048, DOI: 10.1016/j.cose.2018.07.003.
  - [15] Simeonova, I., Metodieva, TS., Model for administrative security management in a municipality, *Journal Scientific and Applied Research*, Konstantin Preslavsky University Press, Vol. 26, Shumen, 2024, ISSN 1314-6289 (Print), ISSN 2815-4622 (Online), pp. 93-105, DOI: <https://doi.org/10.46687/jsar.v26i1.397>.
  - [16] Smith, J., "Optimizing Network Reconnaissance Phases for Penetration Testing Engagements," in *Proc. 2017 APWG Symposium on Electronic Crime Research (eCrime)*, IEEE, pp. 1-12, 2017, ISBN 978-1-5386-2719-2, DOI: 10.1109/ECRIME.2017.7945055.
  - [17] Thompson, G., "The Legal and Ethical Boundaries of Network Discovery and Asset Mapping," *International Journal of Law and Information*

- Technology, vol. 26, no. 1, pp. 55-72, 2018, ISSN 0967-0769, DOI: 10.1093/ijlit/eax021.
- [18] Wagner, M., "A Framework for Continuous Network Asset Identification and Risk Assessment," in Proc. 2021 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '21), ACM, pp. 670-684, 2021, ISBN 978-1-4503-8453-7, DOI: 10.1145/3433210.3453095.
- [19] Williams, F., "Leveraging p0f for Intrusion Detection and Network Forensics," Digital Investigation, vol. 22, pp. 78-89, 2017, ISSN 1742-2876, DOI: 10.1016/j.diin.2017.07.001.
- [20] Zhang, W., "A Hybrid Approach to Network Discovery Combining Netdiscovery and p0f for Enhanced Accuracy," Security and Communication Networks, vol. 2022, 2022, ISSN 1939-0114, DOI: 10.1155/2022/1234567.
- [21] Zimmerman, P., "The Role of Open-Source Intelligence (OSINT) in Modern Network Asset Identification," in Advances in Cybersecurity Management, Springer, 2020, pp. 89-105, ISBN 978-3-030-45666-2, DOI: 10.1007/978-3-030-45667-9\_5.