Exercise: SELinux - Creating a Custom Policy for echodaemon

Steps

Before we do this exercise, we need to start up echodaemon on the RickMorty system if it's not started already.

Wait about a minute for the rickmorty system to fully start / resume from the managedsave.

Now, confirm that your system has received the script from the sync.

```
ls -l /sync/bin/start-rickmorty-echodaemon.sh
```

If it has, make sure that the SSH program is answering on RickyMorty:

```
ssh -o StrictHostKeyChecking=false -p 22222 -i /sync/.rickmortykey root@rickmorty
```

Exit that SSH session. Now, run the start-rickmorty-echodaemon.sh script:

```
/sync/bin/start-rickmorty-echodaemon.sh
```

1. On the RickdickulouslyEasy (`rickmorty`) CTF machine, there's a strange service running on port `22000`.

   Connect to the service with netcat:

   ```
   nc rickmorty 22000
   ```

2. Type a few characters and hit `Enter` to receive a rejection "That's not it! Go away!" message.

3. See if you can guess the password. Take a look at Jay's t-shirt on his bio picture on page ix of this book:

   https://www.amazon.com/Stealing-Network-How-Own-Continent/dp/1931836051

4. The password is at the bottom of this page.

5. You'll see the `root` user's SSH private key file from this system displayed on the screen. Use an editor to write this private key into a file in your

current directory named `rickmorty-key`, run a `chmod go-rwx` on that file, then `ssh` into the `rickmorty` machine:

```
nano rickmorty-key
chmod go-rwx rickmorty-key
ssh -p 22222 -i rickmorty-key root@rickmorty
```

6. Let's figure out which program just let us in:

   ```
   netstat -vantp | grep 22000
   ```

7. OK - this was `echodaemon`. Let's block this attack by creating an SELinux policy for the `echodaemon` program.

8. First, if you've been logged out of the `rickmorty` virtual machine, log back in:

   ```
   ssh -p 22222 -i rickmorty-key root@rickmorty
   ```

9. Note that if `echodaemon` is running, it's running with the type `unconfined_service_t`.

   ```
   ps -efZ | grep [e]chodaemon
   ```

10. Let's make a directory in which to work on our policy:

    ```
    cd ; mkdir selinux ; cd selinux
    ```

11. Also, since we can't cat out a file right now when we want to, run this please:

    ```
    alias cat="grep -v asdfghjkl"
    ```

12. First, we'll run `sepolicy` generate to create a simple dummy policy:

    ```
    sepolicy generate --init /usr/local/sbin/echodaemon
    ```

13. This created:

    ```
    echodaemon.te - a type enforcement file with rules
    echodaemon.if - an interface file describing domain transitions into echodaemon_t
    echodaemon.fc - a file context list, stating how the echodaemon binary should be labele
    ```

14. Compile the policy package for `echodaemon`:

    ```
    make -f /usr/share/selinux/devel/Makefile echodaemon.pp
    ```

15. Ignore all the "duplicate definition" errors. Install the `echodaemon` policy package that created:

    ```
    semodule -i echodaemon.pp
    ```

16. The `echodaemon` policy will start out disabled and in permissive (not enforced, but logging) mode. Enable the policy, but keep it in permissive mode so we can get the AVC log messages about what actions `echodaemon` takes.

```
semodule -e echodaemon
```

17. Stop the `echodaemon` program if it's running.

    ```
    kill `ps -ef | grep [e]chodaemon | awk '{print $2}'`
    ```

18. Now tell SELinux to label the `echodaemon` binary with its new `echodaemon_exec_t` type:

    ```
    restorecon /usr/local/sbin/echodaemon
    ```

19. Observe the new file context on `/usr/local/sbin/echodaemon`:

    ```
    ls -lZ /usr/local/sbin/echodaemon
    ```

20. Start the `echodaemon` program.

    ```
    systemctl start echodaemon
    ```

21. Make sure `echodaemon` is now running with the type `echodaemon_t`.

    ```
    ps -efZ | grep [e]chodaemon
    ```

22. Now, let's exercise the `echodaemon`.

    Note: Any time we make a change, we'll need to restart the `echodaemon` program.

    ```
    kill `ps -ef | grep [e]chodaemon | awk '{print $2}'`
    systemctl start echodaemon
    ```

23. Exercising the Confined Program - Round 1

    Connect to the `echodaemon` from the `rickmorty` system:

    ```
    nc 127.0.0.1 22000
    ```

24. Type any string except for the magic password you typed to see the SSH key file.

    ```
    abc
    ```

25. Now end this netcat session with `Ctrl-C`.

26. Now check out the AVC messages that SELinux logged by running `ausearch`:

    ```
    ausearch -m avc -ts recent | grep echodaemon
    ```

27. Let's pass these log messages straight into `audit2allow`, asking it to append additional rules onto our type enforcement rules file:

    ```
    ausearch -m avc -ts recent | grep echodaemon | audit2allow -R >>echodaemon.te
    ```

28. Remove the permissive line from `echodaemon.te` now, if you haven't done this already:

    ```
    sed -i 's/^permissive/#permissive/' echodaemon.te
    ```

29. Rebuild the `echodaemon` policy package:

    ```
    make -f /usr/share/selinux/devel/Makefile echodaemon.pp
    ```

30. Now upgrade/re-install the `echodaemon` policy package:

    ```
    semodule -i echodaemon.pp
    ```

31. It's time for the first good test. Let's see if our custom SELinux policy will allow the normal behavior we exercised previously. We want to make sure that we can get `echo`'d text. We are not testing the attack yet.

32. Restart the `echodaemon`:

    ```
    kill `ps -ef | grep [e]chodaemon | awk '{print $2}'`
    systemctl start echodaemon
    ```

33. Connect to the `echodaemon` from the `rickmorty` system:

    ```
    nc 127.0.0.1 22000
    ```

34. Type any string except for the magic password you typed to see the SSH key file.

    ```
    def
    ```

35. SELinux isn't allowing us the `echodaemon` program to send us any text back - neither the password request, not the rejection.

36. End this netcat session with `Ctrl-C`.

37. Now check for a new AVC error:

    ```
    ausearch -m AVC -ts recent | grep echodaemon
    ```

38. We have new AVC messages - continue below where we'll repeat the `audit2allow` step.

39. If you don't have any new AVC messages, please checkout out the troubleshooting section Dealing with Silent Block Rules below, then come back here.

40. Adding to the Profile

    Our profile isn't going to catch everything on its first round.

    Let's pass these log messages straight into `audit2allow`, asking it to append additional rules onto our type enforcement rules file:

    ```
    ausearch -m avc -ts recent | grep echodaemon | audit2allow -R >>echodaemon.te
    ```

41. Rebuild the `echodaemon` policy package:

    ```
    make -f /usr/share/selinux/devel/Makefile echodaemon.pp
    ```

42. Install the upgraded `echodaemon` policy package:

    ```
    semodule -i echodaemon.pp
    ```

43. Restart the `echodaemon`:

    ```
    kill `ps -ef | grep [e]chodaemon | awk '{print $2}'`
    systemctl start echodaemon
    ```

44. Connect to the `echodaemon` from the `rickmorty` system:

    ```
    nc 127.0.0.1 22000
    ```

45. Type any string except for the magic password you typed to see the SSH key file.

    ```
    ghi
    ```

46. SELinux isn't allowing us the `echodaemon` program to send us any text back - neither the password request, not the rejection.

47. End this netcat session with `Ctrl-C`.

48. When we check for a new AVC error, we don't find any new ones. To see this, run a `date` command:

    ```
    date
    ```

49. Now change your `ausearch` command so that it finds things only from the last couple minutes - you'll need to replace this time:

    ```
    ausearch -m AVC --start '15:22:07' | grep echodaemon
    ```

50. Now let's try exercising the program, so you can trigger new AVC errors.

51. Connect to the `echodaemon` from the rickmorty system:

    ```
    nc 127.0.0.1 22000
    ```

52. Type any string except for the magic password you typed to see the SSH key file.

    ```
    jkl
    ```

53. SELinux isn't allowing us the `echodaemon` program to send us any text back - neither the password request, not the rejection.

54. End this netcat session with `Ctrl-C`.

55. We'll now have new log messages. Let's pass them straight into `audit2allow`, asking it to append additional rules onto our type enforcement rules file:

    ```
    ausearch -m avc -ts recent | grep echodaemon | audit2allow -R >>echodaemon.te
    ```

56. Rebuild the `echodaemon` policy package:

    ```
    make -f /usr/share/selinux/devel/Makefile echodaemon.pp
    ```

57. Install the upgraded `echodaemon` policy package:

    ```
    semodule -i echodaemon.pp
    ```

58. Restart the `echodaemon`:

    ```
    kill `ps -ef | grep [e]chodaemon | awk '{print $2}'`
    systemctl start echodaemon
    ```

59. Connect to the `echodaemon` from the `rickmorty` system:

    ```
    nc 127.0.0.1 22000
    ```

60. Type any string except for the magic password you typed to see the `/etc/shadow` file.

    ```
    mno
    ```

61. We now see "That's not it! Go away!" echoed back. So our program now works. Hit `Ctrl-C`.

62. If you're at this point, `echodaemon` is letting you enter a single line of text (not the secret key), then giving you a reply. If that's not happening, raise your left hand.

    Now it's time for the moment of truth. Let's make sure the module blocks the backdoor.

63. Connect to the `echodaemon` from the `rickmorty` system:

    ```
    nc 127.0.0.1 22000
    ```

64. Type the magic password you typed to see the SSH key file and make sure it doesn't show up.

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

66. Suspend the virtual machines:

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

Troubleshooting

Dealing with Silent Block Rules

We have a silent block rule that is flagged `dontaudit`. Let's disable the `dontaudit` rules:

```
semodule --disable_dontaudit --build
```

Note

We deactivated the `dontaudit` rules because one was blocking normal program behavior, but not logging an error to the audit log. Once you've created a policy that works, you can re-enable the `dontaudit` rules by running:

```
semodule --build
```

Another cool trick for searching logs is checkpointing - from the `ausearch` man page:

```
ausearch --checkpoint /etc/audit/auditd_checkpoint.txt -i
_au_status=$?
if test ${_au_status} eq 10 -o ${_au_status} eq 11 -o ${_au_status} eq 12 ; then
    ausearch --checkpoint /etc/audit/auditd_checkpoint.txt --start checkpoint -i
fi
```

Password Hint

The password is: `gotroot`