Exercise: seccomp via Docker

Steps

In this exercise, we'll foil the trojan horse's backdoor with `seccomp`, using Docker.

On the RickdickulouslyEasy (`rickmorty`) CTF machine, there's a strange service running on port `22001`.

1. Connect to the service with netcat:

   ```
   nc rickmorty 22001
   ```

2. Type a few characters and hit enter to receive a "No - disconnect and try again" message.

3. See if you can guess the password. You may want to search for movie quotes. Make sure not to capitalize your answer or add any extra spaces to the beginning or end. The password is at the bottom of this page.

4. You'll see the `root` user's SSH private key file from this system displayed on the screen. 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:

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

5. Let's block this attack by running that program with a custom Seccomp profile.

6. First, log into the rickmorty virtual machine if you've logged out:

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

7. Take a look at the `docker ps` listing - notice that port `22001` is actually routing to the same port number on a Docker container called `echoforeground`. That container is running `echofg`.

   ```
   docker ps
   ```

8. Let's build a new Docker image for the `echoforeground` program, but have this one run `strace` to get a list of system calls that `echoforeground`

uses. We'll make the `entrypoint`, the program the container uses as PID 1, this:

```
strace -ff /usr/local/sbin/echoforeground 22001
```

9. Switch into `/root/docker`:

   ```
   cd /root/docker
   ```

10. On this system, the `Dockerfile` is a symbolic link to `Dockerfile-echoforeground`. Let's switch the symbolic link to point to `Dockerfile-strace` instead.

    ```
    ln -nsf Dockerfile-strace Dockerfile
    ```

11. Look at the difference between the two Dockerfiles by running:

    ```
    diff Dockerfile-echoforeground Dockerfile-strace
    ```

12. And now let's build a new image:

    ```
    docker build -t strace-echofg-image .
    ```

13. When you `strace` your program, you'll want to run the Docker container without any security confinement, so you get the program fully exercised and don't have the chance to miss anything. We'll thus use `--security-opt label=disable` to prevent SELinux from acting, as well as `--security-opt seccomp=unconfined` to stop the default seccomp profile from acting. Finally, we'll use `--cap-add ALL` to prevent root capability confinement from blocking anything.

14. To avoid colliding with the current use of port `22001`, we'll remap the host's port `22002` to the container's port `22001`. For our `strace`-profiling container, we'll thus run this:

    ```
    docker run -itd --name strace-echoforeground -p 22002:22001 --security-opt label=disabl
    --security-opt seccomp=unconfined --cap-add ALL strace-echofg-image
    ```

15. Run `docker ps` to see output similar to this:

    ```
    docker ps
    CONTAINER ID        IMAGE                 COMMAND                  CREATED
    e83b8f878f3c        strace-echofg-image   "/usr/local/sbin/e..."   14 seconds ago
    d79bfd70fa7e        echoforeground        "/usr/local/sbin/e..."   Up 30 minutes
    ```

16. Now, let's connect to our `strace`-ing container from the `rickmorty` machine:

    ```
    nc 127.0.0.1 22002
    ```

17. Type anything but the proper password, so we don't get into the trojan horse code, then hit `Ctrl-C` to end the connection.

18. Next, gather the `strace` output from the `strace-echoforeground` container via docker logs:

```
docker logs strace-echoforeground | grep -v strace > strace-for-echofg
```

19. Now, let's parse that `strace` output into a `json` file that tells Docker what system calls should be permitted:

    ```
    ./parse-strace-to-json.sh strace-for-echofg >seccomp-echofg
    ```

20. Let's see how many syscalls are permitted in this new allowlist:

    ```
    grep name seccomp-echofg | wc -l
    ```

21. According to my syscall table, there are 314 system calls available before we use this allowlist. Our allowlist lets us restrict the program to well under 10% of those total 314 calls.

22. Now let's run a container from the same `echoforeground` that was already here running the service on TCP port `22001`. We'll place this one on TCP port `22003`. Here's a reference:

    ```
    Port    Name/Memo
    ----    ---------
    22000   echodaemon, for SELinux exercise
    22001   echoforeground in Docker, w/o confinement
    22002   strace for echoforeground in Docker
    22003   echoforeground confined w/ seccomp
    ```

23. Start the container, using the new `seccomp` filter:

    ```
    docker run -itd --name confined-echofg -p 22003:22001 --security-opt no-new-privileges
    --security-opt seccomp=seccomp-echofg --restart always echoforeground
    ```

24. Now connect to `localhost` port `22003` to connect to the `seccomp`-confined `echoforeground`.

    ```
    nc 127.0.0.1 22003
    ```

25. Let's make sure the program works by answering incorrectly and hitting `Ctrl-C` to terminate the connection.

26. Next, make sure that you've defeated the backdoor by reconnecting and entering the password.

    If you don't get an SSH key file back for the correct password, you're successful.

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

28. Suspend the virtual machines:

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

Password Hint

The password is: `mattersmost`