0x12. Web stack debugging #2

DevOpsSysAdminScriptingDebugging

- By: Sylvain Kalache, co-founder at Holberton School
- Weight: 1
- Ongoing second chance project - started Oct 10, 2022 6:00 AM, must end by Oct 15, 2022 6:00 AM
- An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 6.0/6 mandatory & 1.0/3 optional
- **Altogether: 133.33%**
  - Mandatory: 100.0%
  - Optional: 33.33%
  - Calculation: 100.0% + (100.0% * 33.33%) == **133.33%**

Concepts

For this project, we expect you to look at this concept:

- Web stack debugging

Requirements

General

- All your files will be interpreted on Ubuntu 14.04 LTS
- All your files should end with a new line
- A README.md file at the root of the folder of the project is mandatory
- All your Bash script files must be executable
- Your Bash scripts must pass `Shellcheck` without any error
- Your Bash scripts must run without error
- The first line of all your Bash scripts should be exactly `#!/usr/bin/env bash`
- The second line of all your Bash scripts should be a comment explaining what is the script doing

Tasks

0. Run software as another user

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

The user `root` is, on Linux, the “superuser”. It can do anything it wants, that’s a good and bad thing. A good practice is that one should never be logged in the `root` user, as if you fat finger a command and for example run `rm -rf /`, there is no comeback. That’s why it is preferable to run as a privileged user, meaning that the user also has the ability to perform tasks that the `root` user can do, just need to use a specific command that you need to discover.

For the containers that you are given in this project as well as the checker, everything is run under the `root` user, which has the ability to run anything as another user.

Requirements:

- write a Bash script that accepts one argument
- the script should run the `whoami` command under the user passed as an argument
- make sure to try your script by passing different users

Example:

```
root@ubuntu:~# whoami
root
root@ubuntu:~# ./0-iamsomeoneelse www-data
www-data
root@ubuntu:~# whoami
root
root@ubuntu:~#
```

Repo:

- GitHub repository: `alx-system_engineering-devops`
- Directory: `0x12-web_stack_debugging_2`
- File: `0-iamsomeoneelse`

Done! Help Check your code Get a sandbox QA Review

1. Run Nginx as Nginx
mandatory

Score: 100.0% (*Checks completed: 100.0%*)

The `root` user is a superuser that can do anything on a Unix machine, the top administrator. Security wise, you must do everything that you can to prevent an attacker from logging in as `root`. With this in mind, it's a good practice not to run your web servers as `root` (which is the default for most configurations) and instead run the process as the less privileged `nginx` user instead. This way, if a hacker does find a security issue that allows them to break-in to your server, the impact is limited by the permissions of the `nginx` user.

Fix this container so that Nginx is running as the `nginx` user.

Requirements:

- `nginx` must be running as `nginx` user
- `nginx` must be listening on all active IPs on port `8080`
- You cannot use `apt-get remove`
- Write a Bash script that configures the container to fit the above requirements

After debugging:

```
root@ab6f4542747e:~# ps auxff | grep ngin[x]
nginx      884  0.0  0.0  77360  2744 ?        Ss   19:16   0:00 nginx: master proces
s /usr/sbin/nginx
nginx      885  0.0  0.0  77712  2772 ?        S    19:16   0:00  \_ nginx: worker pr
ocess
nginx      886  0.0  0.0  77712  3180 ?        S    19:16   0:00  \_ nginx: worker pr
ocess
nginx      887  0.0  0.0  77712  3180 ?        S    19:16   0:00  \_ nginx: worker pr
ocess
nginx      888  0.0  0.0  77712  3208 ?        S    19:16   0:00  \_ nginx: worker pr
ocess
root@ab6f4542747e:~#
root@ab6f4542747e:~# nc -z 0 8080 ; echo $?
0
root@ab6f4542747e:~#
```

Repo:

- GitHub repository: `alx-system_engineering-devops`
- Directory: `0x12-web_stack_debugging_2`
- File: `1-run_nginx_as_nginx`

Done! Help Check your code Get a sandbox QA Review

2. 7 lines or less

#advanced

Score: 33.33% (*Checks completed: 33.33%*)

Using what you did for task #1, make your fix short and sweet.

Requirements:

- Your Bash script must be 7 lines long or less
- There must be a new line at the end of the file
- You respect Bash script requirements
- You cannot use `;`
- You cannot use `&&`
- You cannot use `wget`
- You cannot execute your previous answer file (Do not include the name of the previous script in this one)

Repo:

- GitHub repository: `alx-system_engineering-devops`
- Directory: `0x12-web_stack_debugging_2`
- File: `100-fix_in_7_lines_or_less`