

# Sécurité applicative

Laboratoire 7

# Laboratoire 7

Adrien Voisin, Bastien Bodart

IR313 - Henallux 2021-2022

## 1 Introduction

Ce laboratoire aura pour but la mise en place de monitoring permettant la détection d'activité suspecte de la part d'applications.

UTILISEZ TOUJOURS UNE VM POUR RÉALISER LES EXERCICES DE CE COURS, MÊME SI VOUS VOUS SERVEZ DE VOTRE PROPRE PC.

## 2 DNS Monitoring

Mettez en place un service (daemon) utilisant tcpdump qui va monitorer toutes les requêtes DNS (udp 53). Configurer les permissions pour que ce service ne soit pas lancé par root!

### Lancer tcpdump sans être root

Par défaut, vous n'avez pas les droits pour lancer une capture :

```
(user@host) - [~/Documents/Labo7]  
$ tcpdump  
tcpdump: eth0: You don't have permission to capture on that device  
(socket: Operation not permitted)
```

Sur Google, on tape « run tcpdump as non root ».

On tombe sur :

<https://gist.github.com/zapstar/3d2ff4f345b43ce7918889053503ef84>

```

<> noroot_tcpdump.sh
1  #!/usr/bin/env bash
2
3  # NOTE: This will let anyone who belongs to the 'pcap' group
4  # execute 'tcpdump'
5
6  # NOTE2: User running the script MUST be a sudoer. It is
7  # convenient to be able to sudo without a password.
8
9  sudo groupadd pcap
10 sudo usermod -a -G pcap $USER
11 sudo chgrp pcap /usr/sbin/tcpdump
12 sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
13 sudo ln -s /usr/sbin/tcpdump /usr/bin/tcpdump

```

Attention, notez que, sur Kali, ce n'est pas « sbin » comme sur le GitHub, mais bien « bin ».

On peut s'en assurer avec la commande « locate » :

```

$ locate tcpdump
/etc/apparmor.d/usr.bin.tcpdump
/etc/apparmor.d/local/usr.bin.tcpdump
/usr/bin/tcpdump
/usr/share/bash-completion/completions/tcpdump
/usr/share/doc/tcpdump
/usr/share/doc/tcpdump/NEWS.Debian.gz
/usr/share/doc/tcpdump/README.Debian
/usr/share/doc/tcpdump/README.md.gz
/usr/share/doc/tcpdump/changelog.Debian.gz
/usr/share/doc/tcpdump/changelog.gz
/usr/share/doc/tcpdump/copyright
/usr/share/doc/tcpdump/examples
/usr/share/doc/tcpdump/examples/atime.awk
/usr/share/doc/tcpdump/examples/packetdat.awk
/usr/share/doc/tcpdump/examples/send-ack.awk
/usr/share/doc/tcpdump/examples/stime.awk
/usr/share/man/man8/tcpdump.8.gz
/usr/share/mime/application/vnd.tcpdump.pcap.xml
/usr/share/zsh/functions/Completion/Unix/_tcpdump
/var/cache/apt/archives/tcpdump_4.99.1-2_amd64.deb
/var/lib/dpkg/info/tcpdump.conf files
/var/lib/dpkg/info/tcpdump.list
/var/lib/dpkg/info/tcpdump.md5sums
/var/lib/dpkg/info/tcpdump.postinst
/var/lib/dpkg/info/tcpdump.postrm
/var/lib/dpkg/info/tcpdump.preinst
/var/lib/dpkg/info/tcpdump.prerm

```

Remplissez à présent un script avec les commandes trouvées sur GitHub :

```
GNU nano 5.4                                ex1.sh *
#!/usr/bin/env bash

# NOTE: This will let anyone who belongs to the 'pcap' group
# execute 'tcpdump'

# NOTE2: User running the script MUST be a sudoer. It is
# convenient to be able to sudo without a password.

sudo groupadd pcap
sudo usermod -s -G pcap $USER
sudo chgrp pcap /usr/bin/tcpdump
sudo setcap cap_net_raw,cap_net_admin=eip /usr/bin/tcpdump
```

```
(user@host)~[~/Documents/Labo7]
$ chmod +x ex1.sh
```

Et on l'exécute avec notre user :

```
# ./ex1.sh
```

A présent, on peut lancer tcpdump en tant qu'utilisateur :

```
(user@host)~[~/Documents/Labo7]
$ tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

### Le transformer en daemon

Dans « /lib/systemd/system », on crée un service. Par exemple : « myprog.service ».

```
GNU nano 5.4                                myprog.service
[Unit]
After=network.target

[Service]
Restart=always
RestartSec=30
Environment="TCPDUMP_FORMAT=%Y-%m-%d_%H-%M"
#ExecStartPre=/bin/mkdir -p /var/log/tcpdumpd/
ExecStart=/usr/bin/tcpdump -i eth0 port 53 -w '/var/log/tcpdumpd/myfile.pcap' -U
#ExecStop=/bin/kill -s QUIT $MAINPID

[Install]
WantedBy=multi-user.target
```

La commande la plus importante ici est : « ExecStart=/usr/bin... »

Le « -U » sert à écrire dynamiquement, dès qu'il reçoit quelque chose à écrire, il l'écrit.

On démarre ensuite notre service :

```
(user@host)-[/lib/systemd/system]
$ systemctl start myprog.service
```

Lancer le service au boot du système

```
(user@host)-[/lib/systemd/system]
$ systemctl enable myprog.service
Created symlink /etc/systemd/system/multi-user.target.wants/myprog.service → /lib/systemd/system/myprog.service.

$ systemctl status myprog.service
● myprog.service
   Loaded: loaded (/lib/systemd/system/myprog.service; disabled; vendor preset: disabled)
   Active: active (running) since Thu 2021-12-02 12:10:03 CET; 6min ago
     Main PID: 3105 (tcpdump)
       Tasks: 1 (limit: 4633)
      Memory: 1.1M
         CPU: 7ms
    CGroup: /system.slice/myprog.service
            └─3105 /usr/bin/tcpdump -i eth0 port 53 -w /var/log/tcpdumpd/myfile.pcap -U
```

```
(user@host)-[/lib/systemd/system]
$ nslookup google.com
Server:         10.101.250.9
Address:        10.101.250.9#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.179.174
Name:   google.com
Address: 2a00:1450:400e:80d::200e
```

Et enfin, on peut le visualiser dans Wireshark

```
(user@host)-[/lib/systemd/system]
$ wireshark /var/log/tcpdumpd/myfile.pcap
```

L'interface graphique de Wireshark se lance avec les paquets récoltés précédemment.

### 3 strace

*strace* est un outil permettant le monitoring des appels systèmes d'un processus. Utilisez cet outil *strace* afin d'identifier tous les appels systèmes invoqués par les exécutables du laboratoire précédent.

Il est également possible d'attacher *strace* à un processus en cours. Écrivez un script d'audit qui va générer une trace des appels systèmes pour un processus en cours d'exécution. Le résultat sera stocké dans `/var/log/strace/<PID>.log`.

#### Installation

```
(user@host)-[/lib/systemd/system]
$ sudo apt-get install strace
```

#### Exemples d'utilisation

- L'option : `-P PATH` : Tracer les accès au chemin.

Exemple :

```
(user@host)-[/lib/systemd/system]
$ sudo strace -P /etc/ld.so.cache ls /var/local
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=84062, ... }) = 0
mmap(NULL, 84062, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f75a713f000
close(3) = 0
+++ exited with 0 +++
```



Analyse de l'exécutable « curious » du laboratoire précédent :

```
(user@host)~[~/Documents/Labo 6/lab6]
$ strace ./curious
execve("./curious", ["/curious"], 0x7ffdc650810 /* 54 vars */) = 0
brk(NULL) = 0x555ded11b000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=84140, ...}) = 0
mmap(NULL, 84140, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fdc36a30000
close(3) = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0\0\0\1\0\0\0\200\177\2\0\0\0\0" ..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1839168, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdc36a2e000
mmap(NULL, 1852480, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fdc36869000
mprotect(0x7fdc3686f000, 1658880, PROT_NONE) = 0
mmap(0x7fdc3686f000, 1347584, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7fdc3686f000
mmap(0x7fdc369d8000, 307200, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x16f000) = 0x7fdc369d8000
mmap(0x7fdc36a24000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1ba000) = 0x7fdc36a24000
mmap(0x7fdc36a2a000, 13376, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fdc36a2a000
close(3) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdc36867000
arch_prctl(ARCH_SET_FS, 0x7fdc36a2f540) = 0
mprotect(0x7fdc36a24000, 12288, PROT_READ) = 0
mprotect(0x555deca00000, 4096, PROT_READ) = 0
mprotect(0x7fdc36a6f000, 4096, PROT_READ) = 0
munmap(0x7fdc36a30000, 84140) = 0
rt_sigaction(SIGINT, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7fdc368a5ef0}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
rt_sigaction(SIGQUIT, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7fdc368a5ef0}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
rt_sigprocmask(SIG_BLOCK, [CHLD], [], 8) = 0
mmap(NULL, 36864, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fdc36a3c000
rt_sigprocmask(SIG_BLOCK, ~[], [CHLD], 8) = 0
clone(child_stack=0x7fdc36a44ff0, flags=CLONE_VM|CLONE_VFORK|SIGCHLD) = 1942
munmap(0x7fdc36a3c000, 36864) = 0
rt_sigprocmask(SIG_SETMASK, [CHLD], NULL, 8) = 0
wait4(1942, ls: cannot open directory '/home/client/lost+found': Permission denied
[!WIFEXITED(s) && WEXITSTATUS(s) == 1], 0, NULL) = 1942
rt_sigaction(SIGINT, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7fdc368a5ef0}, NULL, 8) = 0
rt_sigaction(SIGQUIT, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7fdc368a5ef0}, NULL, 8) = 0
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=1942, si_uid=1000, si_status=1, si_utime=0, si_stime=0} ---
exit_group(0) = ?
+++ exited with 0 +++
```

## Traçage d'un processus en cours d'exécution

### Option : -p

Exemple :

On run Firefox

Puis :

```
(user@host)~[~/lib/systemd/system]
$ ps -o | grep firefox
3660 ?      00:00:05 firefox-esr
```

Vous verrez une notification indiquant que strace s'est attaché au processus, puis les appels de trace système seront affichés.

```
(user@host)~[~/Lib/systemd/system]
$ sudo strace -p 3660
strace: Process 3660 attached
restart_syscall(<... resuming interrupted read ...>) = 1
read(28, "\372", 1)                                = 1
futex(0x7f537fe083f0, FUTEX_WAKE_PRIVATE, 1) = 1
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
poll([{fd=4, events=POLLIN}], {fd=5, events=POLLIN}, {fd=7, events=POLLIN}, {fd=28, events=POLLIN}], 4, 0) = 0 (Timeout)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
poll([{fd=4, events=POLLIN}], {fd=5, events=POLLIN}, {fd=7, events=POLLIN}, {fd=28, events=POLLIN}], 4, 0) = 0 (Timeout)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
poll([{fd=4, events=POLLIN}], {fd=5, events=POLLIN}, {fd=7, events=POLLIN}, {fd=28, events=POLLIN}], 4, 0) = 0 (Timeout)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
poll([{fd=4, events=POLLIN}], {fd=5, events=POLLIN}, {fd=7, events=POLLIN}, {fd=28, events=POLLIN}], 4, -1) = 1 ([{fd=28,
read(28, "\372", 1)                                = 1
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
poll([{fd=4, events=POLLIN}], {fd=5, events=POLLIN}, {fd=7, events=POLLIN}, {fd=28, events=POLLIN}], 4, 0) = 0 (Timeout)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
poll([{fd=4, events=POLLIN}], {fd=5, events=POLLIN}, {fd=7, events=POLLIN}, {fd=28, events=POLLIN}], 4, 0) = 0 (Timeout)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
poll([{fd=4, events=POLLIN}], {fd=5, events=POLLIN}, {fd=7, events=POLLIN}, {fd=28, events=POLLIN}], 4, 0) = 0 (Timeout)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
recvmsg(4, {msg_namelen=0}, 0)                     = -1 EAGAIN (Resource temporarily unavailable)
```

Créer un script dont l'objectif est de générer une trace des appels système pour un processus. Son résultat sera stocké dans : « /var/log/[pid].log ».

Code :

```
GNU nano 5.4 ex2.sh
#!/bin/bash

echo Démarrage

if test $# -ne 1
then
    exit 1
fi

process=$1

pid=$(pidof $process)
echo le pid est : $pid

fileName=$(echo $pid".log")

mkdir -p /var/log/strace/

sudo touch /var/log/strace/$fileName

sudo strace -p $pid -o /var/log/strace/$fileName ls

echo terminé
```

Le rendre exécutable :





## 4 auditd

Mettez en place le daemon *auditd* sur votre machine. Faites en sorte de l'ajouter aux options du kernel dans GRUB (voir dans */etc/default/grub*). Ajoutez des règles permettant de monitorer :

- les accès en lecture, écriture et modification de permissions de */etc/group,passwd,shadow,sudoers*
- les accès en lecture et écriture sur le dossier */etc/security*
- l'ensemble des appels systèmes effectués par l'utilisateur 1000
- les appels systèmes *open* et *chmod*

Essayez de réfléchir à d'autres règles qui vous semblent adéquates. Ajoutez également un règle pour empêcher la modification de la configuration d'audit.

Vérifiez votre configuration en effectuant quelques modifications et appels, et vérifiez ensuite avec *ausearch* et *aureport*

Documentation :

- [https://wiki.archlinux.org/title/Audit\\_framework](https://wiki.archlinux.org/title/Audit_framework)
- <https://connect.ed-diamond.com/GNU-Linux-Magazine/glmfhs-093/journalisez-les-actions-de-vos-utilisateurs-avec-auditd>
- <https://www.redhat.com/sysadmin/configure-linux-auditing-auditd>

### Installation

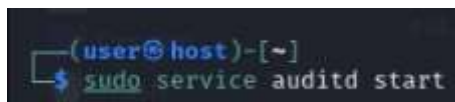
```
#sudo apt-get install auditd
```

### Configuration

Modifier « */boot/grub/grub.cfg* » et ajouter :

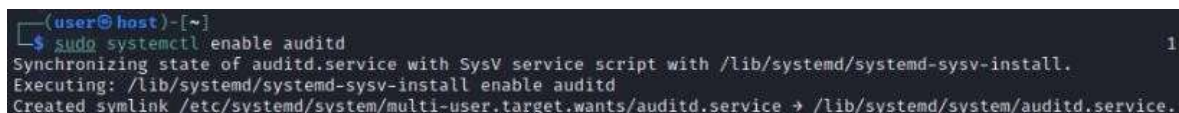
```
#audit=1
```

Démarrer le service :



```
(user@host)-[~]
$ sudo service auditd start
```

Activez le daemon « *auditd* », pour qu'il puisse démarrer au démarrage :



```
(user@host)-[~]
$ sudo systemctl enable auditd
Synchronizing state of auditd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable auditd
Created symlink /etc/systemd/system/multi-user.target.wants/auditd.service → /lib/systemd/system/auditd.service.
```

- les accès en lecture, écriture et modification de permissions de */etc/group,passwd,shadow,sudoers*

```
#man auditctl
```

```
-p [r|w|x|a]
Describe the permission access type that a file system watch will trigger on. r=read, w=write,
x=execute, a=attribute change. These permissions are not the standard file permissions, but rather
the kind of syscall that would do this kind of thing. The read & write syscalls are omitted from
this set since they would overwhelm the logs. But rather for reads or writes, the open flags are
looked at to see what permission was requested.
```

Créer la règle

```
(user@host)-[~]
$ sudo auditctl -w /etc/passwd -p rwa
```

Voir les règles

```
(user@host)-[~]
$ sudo auditctl -l
-w /etc/passwd -p rwa
```

Supprimer toutes les règles

```
(user@host)-[~]
$ sudo auditctl -D
No rules
```

Vérifier que la règle est monitorée :

On ouvre un autre terminal et on tape dans « /var/log/audit » :

```
#tail -f audit.log
```

Quand on tape par exemple « cat /etc/passwd » on voit sur le nouveau terminal que l'accès est bien monitoré :

```
type=SYSCALL msg=audit(1639244273.117:1364): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=7ffd603eb44
8 a2=0 a3=0 items=1 ppid=1014 pid=2768 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=100
0 fsgid=1000 tty=pts0 ses=2 comm="cat" exe="/usr/bin/cat" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=openat AUID
="user" UID="user" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=CWD msg=audit(1639244273.117:1364): cwd="/home/user"
type=PATH msg=audit(1639244273.117:1364): item=0 name="/etc/passwd" inode=2099622 dev=08:01 mode=0100644 ouid=0 ogid
=0 rdev=00:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00000000 OUID="root"
type=PROCTITLE msg=audit(1639244273.117:1364): proctitle=636174002F6574632F706173737764
```

OK ! Maintenant qu'on sait que ça fonctionne, on ajoute les autres règles demandées :

```
(user@host)-[~]
$ sudo auditctl -w /etc/group -p rwa

(user@host)-[~]
$ sudo auditctl -w /etc/shadow -p rwa

(user@host)-[~]
$ sudo auditctl -w /etc/sudoers -p rwa
```

Vérification :

```
(user@host)-[~]
$ sudo auditctl -l
-w /etc/passwd -p rwa
-w /etc/group -p rwa
-w /etc/shadow -p rwa
-w /etc/sudoers -p rwa
```

On essaye de lire shadow :

```
type=SYSCALL msg=audit(1639244584.357:1522): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=55906842c27
0 a2=0 a3=0 items=1 ppid=1014 pid=2885 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=100
0 fsgid=1000 tty=pts0 ses=2 comm="nano" exe="/usr/bin/nano" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=openat AU
ID="user" UID="user" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=CWD msg=audit(1639244584.357:1522): cwd="/home/user"
type=PATH msg=audit(1639244584.357:1522): item=0 name="/etc/shadow" inode=2097310 dev=08:01 mode=0100664 ouid=0 ogid
=42 rdev=00:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0000000000000000 OUID="root" OGID="shadow"
type=PROCTITLE msg=audit(1639244584.357:1522): proctitle=6E616E6F002F6574632F736861646F77
█
```

Pour que les règles soient persistantes, il faut les copier dans le fichier :

« /etc/audit/rules.d/audit.rules »

- les accès en lecture et écriture sur le dossier /etc/security

```
(user@host)-[~]
$ sudo auditctl -w /etc/security -p rw
```

```
(user@host)-[/etc/security]
$ ls
access.conf  group.conf  limits.d    namespace.d  opasswd      sepermit.conf  user_map.conf
faillock.conf limits.conf namespace.conf namespace.init pam_env.conf  time.conf
```

J'accède au dossier, et on peut voir que ma commande « ls » a bien été détectée :

```
type=SYSCALL msg=audit(1639244784.385:1566): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=55a3ce59249
0 a2=90800 a3=0 items=1 ppid=1014 pid=2902 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid
=1000 fsgid=1000 tty=pts0 ses=2 comm="ls" exe="/usr/bin/ls" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=openat AU
ID="user" UID="user" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=CWD msg=audit(1639244784.385:1566): cwd="/etc/security"
type=PATH msg=audit(1639244784.385:1566): item=0 name="." inode=2097237 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:
00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0000000000000000 OUID="root" OGID="root"
type=PROCTITLE msg=audit(1639244784.385:1566): proctitle=6C73002D2D636F6C6F72306175746F
█
```



Si je fais un « `ls` » avec le compte `root`, il est également monitoré :

```
type=SYSCALL msg=audit(1639244903.753:1622): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=56373b980de
0 a2=90800 a3=0 items=1 ppid=2907 pid=2918 audit=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts
0 ses=2 comm="ls" exe="/usr/bin/ls" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=openat AUID="user" UID="root" GID
="root" EUID="root" SUID="root" FSUID="root" EGID="root" SGID="root" FSGID="root"
type=CWD msg=audit(1639244903.753:1622): cwd="/etc/security"
type=PATH msg=audit(1639244903.753:1622): item=0 name="." inode=2097237 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:
00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PROCTITLE msg=audit(1639244903.753:1622): proctitle=6C73002D2D636F6C6F723D6175746F002D6C
type=SYSCALL msg=audit(1639244903.753:1623): arch=c000003e syscall=192 success=no exit=-61 a0=7ffe8b445830 a1=7f4d0a
353753 a2=56373b9906a0 a3=ff items=1 ppid=2907 pid=2918 audit=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fs
gid=0 tty=pts0 ses=2 comm="ls" exe="/usr/bin/ls" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=lgetxattr AUID="user
" UID="root" GID="root" SUID="root" FSUID="root" EGID="root" SGID="root" FSGID="root"
type=CWD msg=audit(1639244903.753:1623): cwd="/etc/security"
type=PATH msg=audit(1639244903.753:1623): item=0 name="limits.d" inode=2097242 dev=08:01 mode=040755 ouid=0 ogid=0 r
dev=00:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PROCTITLE msg=audit(1639244903.753:1623): proctitle=6C73002D2D636F6C6F723D6175746F002D6C
type=SYSCALL msg=audit(1639244903.753:1624): arch=c000003e syscall=191 success=no exit=-61 a0=7ffe8b445830 a1=56373a
5c4ff3 a2=0 a3=0 items=1 ppid=2907 pid=2918 audit=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pt
s0 ses=2 comm="ls" exe="/usr/bin/ls" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=getxattr AUID="user" UID="root"
GID="root" EUID="root" SUID="root" FSUID="root" EGID="root" SGID="root" FSGID="root"
type=CWD msg=audit(1639244903.753:1624): cwd="/etc/security"
type=PATH msg=audit(1639244903.753:1624): item=0 name="limits.d" inode=2097242 dev=08:01 mode=040755 ouid=0 ogid=0 r
dev=00:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PROCTITLE msg=audit(1639244903.753:1624): proctitle=6C73002D2D636F6C6F723D6175746F002D6C
type=SYSCALL msg=audit(1639244903.753:1625): arch=c000003e syscall=191 success=no exit=-61 a0=7ffe8b445830 a1=56373a
5c500b a2=0 a3=0 items=1 ppid=2907 pid=2918 audit=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pt
s0 ses=2 comm="ls" exe="/usr/bin/ls" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=getxattr AUID="user" UID="root"
```

...

- l'ensemble des appels systèmes effectués par l'utilisateur 1000

```
#auditctl -a always,exit -S all -F uid=1000
```

```
(user@host)-[~]
$ sudo auditctl -a always,exit -S all -F uid=1000
```

```
(user@host)-[~]
$ sudo auditctl -l
-a always,exit -S all -F uid=1000
```



A présent tout ce que je fais avec mon utilisateur qui a un pid de 1000, à savoir « user » est monitoré.

```
(root@host)-[/var/log/audit]
# tail -f audit.log | grep uid=1000
```

Résultat

```
(root@host)-[/var/log/audit]
# tail -f audit.log | grep uid=1000
type=SYSCALL msg=audit(1639343397.515:2030): arch=c000003e syscall=7 success=yes exit=0 a0=7ffe5bf62f70 a1=1 a2=0 a3=
=8 items=0 ppid=1128 pid=1471 audit=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1
000 tty=pts0 ses=2 comm="nano" exe="/usr/bin/nano" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=poll AUID="user" U
ID="user" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=SYSCALL msg=audit(1639343397.519:2031): arch=c000003e syscall=7 success=yes exit=0 a0=7ffe5bf62f70 a1=1 a2=0 a3
=0 items=0 ppid=1128 pid=1471 audit=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1
000 tty=pts0 ses=2 comm="nano" exe="/usr/bin/nano" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=poll AUID="user" U
ID="user" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=SYSCALL msg=audit(1639343397.519:2032): arch=c000003e syscall=1 success=yes exit=5 a0=1 a1=55b204706450 a2=5 a3
=7f1ae2354767 items=0 ppid=1128 pid=1471 audit=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1
000 fsgid=1000 tty=pts0 ses=2 comm="nano" exe="/usr/bin/nano" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=write A
UID="user" UID="user" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=SYSCALL msg=audit(1639343397.519:2033): arch=c000003e syscall=13 success=yes exit=0 a0=14 a1=7ffe5bf62da0 a2=0
a3=8 items=0 ppid=1128 pid=1471 audit=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid
=1000 tty=pts0 ses=2 comm="nano" exe="/usr/bin/nano" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=rt_sigaction AUI
D="user" UID="user" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=SYSCALL msg=audit(1639343397.519:2034): arch=c000003e syscall=1 success=yes exit=12 a0=1 a1=55b204706450 a2=c a
3=8 items=0 ppid=1128 pid=1471 audit=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid
=1000 tty=pts0 ses=2 comm="nano" exe="/usr/bin/nano" subj=unconfined key=(null)ARCH=x86_64 SYSCALL=write AUID="user"
UID="user" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
```

- les appels systèmes open et chmod

Test avec Kill :

J'utilise mon script précédent :

```
(user@host)-[~/Documents/Labo7]
$ ./ex2.sh mousepad
Démarrage
le pid est : 4659
```

Je crée une règle et je tue mousepad

```
(user@host)-[~/Documents/Labo7]
$ sudo auditctl -a exit,always -F arch=b64 -S kill -k test_kill
(user@host)-[~/Documents/Labo7]
$ kill 4659
```

### Analyse :

```

type=USER_END msg=audit(1639252650.006:2885): pid=4702 uid=1000 auid=1000 ses=2 subj=unconfined msg=op=PAM:session
_close grantors=pam_permit,pam_unix acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/3 res=succes
s'UID="user" AUID="user"
type=CRED_DISP msg=audit(1639252650.006:2886): pid=4702 uid=1000 auid=1000 ses=2 subj=unconfined msg=op=PAM:setcre
d grantors=pam_permit acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/3 res=success'UID="user" A
UID="user"
type=SYSCALL msg=audit(1639252650.010:2887): arch=c000003e syscall=62 success=no exit=-3 a0=ffffeda8 a1=0 a2=0 a3=2e
fd30 item=0 ppid=931 pid=3820 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=
1000 tty=pts3 ses=2 comm="zsh" exe="/usr/bin/zsh" subj=unconfined key="test_kill"ARCH=x86_64 SYSCALL=kill AUID="use
r" UID="user" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=PROCTITLE msg=audit(1639252650.010:2887): proctitle="/usr/bin/zsh"
type=SYSCALL msg=audit(1639252650.014:2888): arch=c000003e syscall=62 success=yes exit=0 a0=1266 a1=0 a2=0 a3=8 item
s=0 ppid=931 pid=3820 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=
pts3 ses=2 comm="zsh" exe="/usr/bin/zsh" subj=unconfined key="test_kill"ARCH=x86_64 SYSCALL=kill AUID="user" UID="u
ser" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=OBJ_PID msg=audit(1639252650.014:2888): opid=4710 oauid=1000 ouid=1000 oses=2 obj=unconfined ocomm="zsh"OAUID=
"user" OUID="user"
type=PROCTITLE msg=audit(1639252650.014:2888): proctitle="/usr/bin/zsh"
type=SYSCALL msg=audit(1639252650.014:2889): arch=c000003e syscall=62 success=no exit=-3 a0=1266 a1=0 a2=0 a3=8 item
s=0 ppid=931 pid=3820 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=
pts3 ses=2 comm="zsh" exe="/usr/bin/zsh" subj=unconfined key="test_kill"ARCH=x86_64 SYSCALL=kill AUID="user" UID="u
ser" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=PROCTITLE msg=audit(1639252650.014:2889): proctitle="/usr/bin/zsh"
type=SYSCALL msg=audit(1639252650.042:2890): arch=c000003e syscall=62 success=yes exit=0 a0=1267 a1=0 a2=0 a3=8 item
s=0 ppid=931 pid=3820 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=
pts3 ses=2 comm="zsh" exe="/usr/bin/zsh" subj=unconfined key="test_kill"ARCH=x86_64 SYSCALL=kill AUID="user" UID="u
ser" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=OBJ_PID msg=audit(1639252650.042:2890): opid=4711 oauid=1000 ouid=1000 oses=2 obj=unconfined ocomm="zsh"OAUID=
"user" OUID="user"
type=PROCTITLE msg=audit(1639252650.042:2890): proctitle="/usr/bin/zsh"
type=SYSCALL msg=audit(1639252650.042:2891): arch=c000003e syscall=62 success=no exit=-3 a0=1267 a1=0 a2=0 a3=8 item
s=0 ppid=931 pid=3820 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=
pts3 ses=2 comm="zsh" exe="/usr/bin/zsh" subj=unconfined key="test_kill"ARCH=x86_64 SYSCALL=kill AUID="user" UID="u
ser" GID="user" EUID="user" SUID="user" FSUID="user" EGID="user" SGID="user" FSGID="user"
type=PROCTITLE msg=audit(1639252650.042:2891): proctitle="/usr/bin/zsh"

```