Un paseo pola seguridade de GNU/Linux

Eloy Pérez González

Whoami

- Membro de Hackliza 

- Mástodon: @zer1t0@defcon.social

Que imos ver?

- Autenticación
- Autorización
- ~~Seguridade en redes~~
- ~~Hardening~~

Obxetivo

- Coñecer un pouco mais os internals de GNU/Linux
- Saber para que serve cada mecanismo de seguridade

Autenticación

- Demostrar que somos quen dicimos ser
  - Identificarse
  - Dar unha proba da identificación (contrasinal, certificado, DNI)

- Como funciona isto en GNU/Linux?

Identificación

- Nomes de usuario
- Identificadores de usuario = uid
- Exemplo: ada ↔ 1002

Fontes de Identificación

- /etc/passwd

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

  ada:x:1002:1003::/home/ada:/bin/bash

- Directorio externo
  - Microsoft Active Directory
  - FreeIPA

Autenticación

- Que pasa se como root facemos setuid(9999)? (usuario non existente)
  - a) Non se pode porque o usuario non existe.
  - b) Non se pode, hai que pasar un contrasinal como segundo parámetro deste xeito setuid(9999, “P4ssw0rd”)?
  - c) Éxito

Autenticación

- Que pasa se como root facemos setuid(9999)? (usuario non existente)
  - a) Non se pode porque o usuario non existe.
  - b) Non se pode, hai que pasar un contrasinal como segundo parámetro deste xeito setuid(9999, “P4ssw0rd”)?
  - c) Éxito

Autenticación no kernel

- O kernel Linux so coñece uids
- “Autenticación” por privilexios
  - usuario con CAP_SETUID (xeralmente root)
  - Usando setuid

Autenticación en user space

- *Contrasinal, clave privada, certificado, PIN, etc…*
- *Moi complexo*
- *Solución: PAM (Pluggable Authentication Modules)*

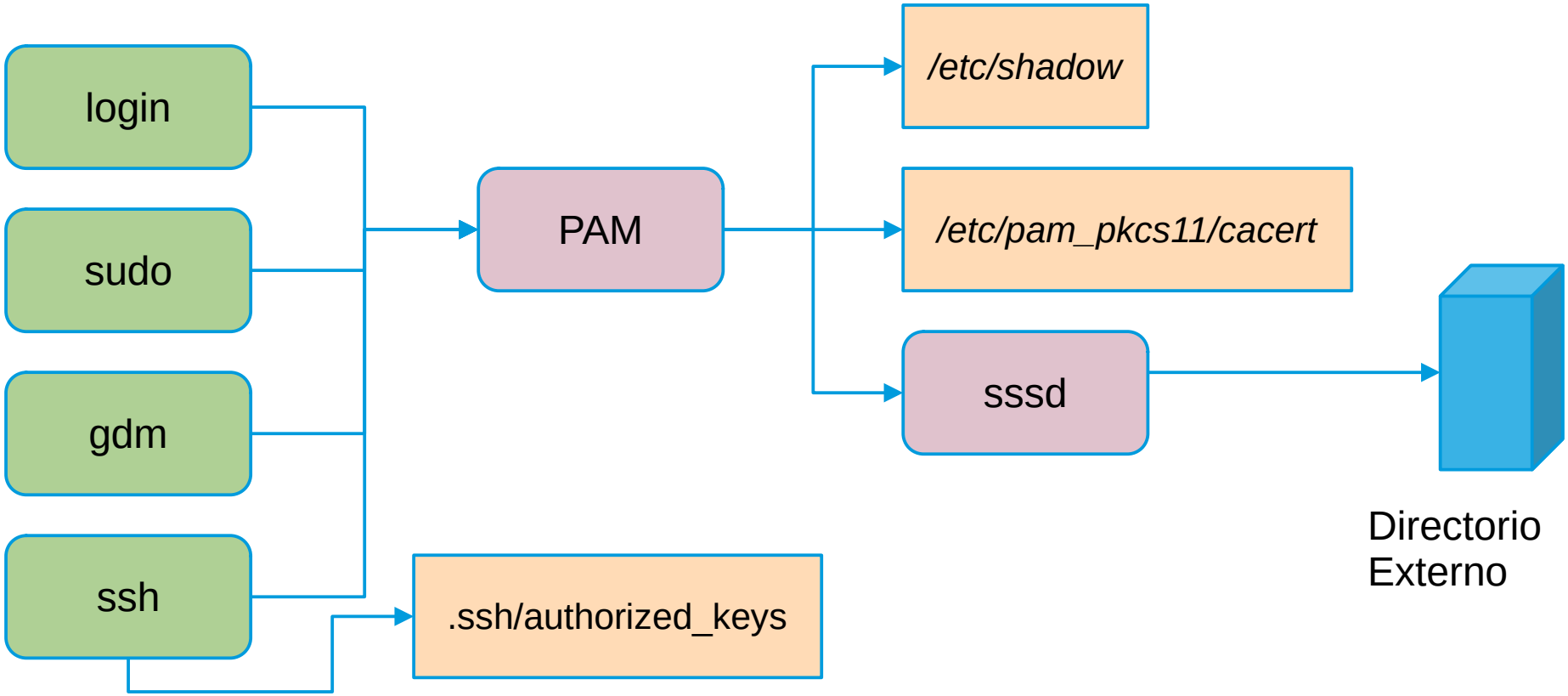
PAM

- Libraría de autenticación extensible por módulos
- Permite:
  - Autenticación
  - Cambio de credenciais
  - Creación do entorno de usuario

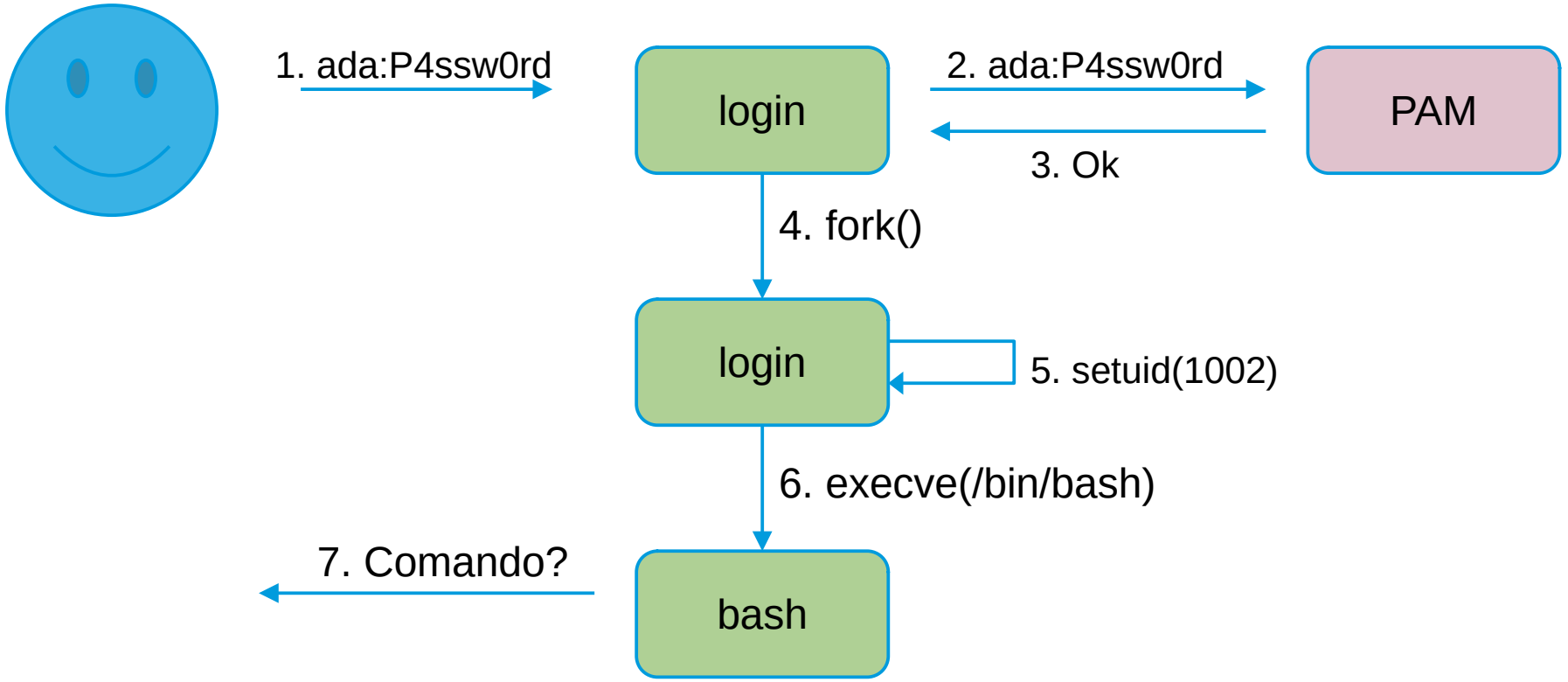
Módulos PAM

- Contrasinais
  - pam_unix.so => /etc/shadow
  - pam_sss.so => sssd => Directory server
- Certificados
  - pam_pkcs11.so
- PIN
  - pam_pwdfile.so ??
- ...

Esquema autenticación



Exemplo de autenticación



Autorización

- Permitir a alguén (sabendo quen é) facer algo
- Maiormente a nivel de kernel Linux
  - Algunhas excepcións como reglas sudo

Autorización

- DAC (Discretionary Access Control)
  - Cada usuaria pode cambiar os permisos dos seus obxetos
  - Exemplo: Permisos de ficheiros
- MAC (Mandatory Access Control)
  - Os permisos os decide unha entidade central
  - Capabilities, SELinux, etc

Uids de usuario

- uid ou ruid (Real uid) → O uid real
- euid (Effective uid) → O que se comproba ao facer operacións
- suid (Saved uid) → Para retornar do euid
- fsuid (Filesystem uid) → O que se comproba ao facer operacións sobre ficheiros (igual que o euid case sempre)

Grupos

- Permiten darlle permisos a varios usuarios á vez

- /etc/group

  root:x:0:

  ada:x:1002:

  sudo:x:27:ada

- Directorio externo

Pregunta sobre grupos

- Pódense engadir grupos a outros grupos en Linux?

Permisos de filesystem

- Linux usa VFS
  - Soporta varios filesystem: ext4, fat32, …
- Non tódolos filesystem son iguais
  - Pode non soportar certas operacións
- Filesystems especiais: procfs, securityfs, etc

Ejemplo procfs

```
$ ls -l /proc/3957/maps
-r--r--r-- 1 root root 0 may 24 09:35 /proc/3957/maps


$ cat /proc/3957/maps
cat: /proc/3957/maps: Permission denied
```

- Razón: Necesitanse permisos de PTRACE

Ejemplo securityfs

```
$ ls -l /sys/kernel/security/apparmor/profiles
-r--r--r-- 1 root root 0 may 24 09:04 /sys/kernel/security/apparmor/profiles

$ cat /sys/kernel/security/apparmor/profiles
cat: /sys/kernel/security/apparmor/profiles: Permission denied
```

- Razón: securityfs usa permisos adhoc

Puntos de montaxe

- Soportan varias opcións que limitan as operacións:
  - Read only
  - No executable
  - No setuid bit

```
$ findmnt /proc
TARGET SOURCE FSTYPE OPTIONS
/proc  proc   proc   rw,nosuid,nodev,noexec,relatime
```

Permisos de ficheiro

- Permisos básicos:
  - Ler
  - Escribir
  - Executar
- Permisos especiais:
  - setuid
  - setgid
  - Sticky bit
- Alcance:
  - Usuaria do ficheiro
  - Grupo do ficheiro
  - Outras usuarias

```
$ chmod 654 a.txt

$ ls -l a.txt
-rw-r-xr-- 1 ada ada 0 ago  7  2023 a.txt
```

setuid

- Permite poñer obter os permisos do propietario do ficheiro

```
$ sudo chmod u+s /usr/bin/id

$ ls -l /usr/bin/id
-rwsr-xr-x 1 root root 39432 feb  8 04:46 /usr/bin/id

$ id
uid=1002(ada) gid=1003(ada) euid=0(root)
groups=1003(ada),27(sudo)
```

Pregunta

```
$ id ada
uid=1002(ada) gid=1003(ada) groups=1003(ada),27(sudo),1005(it)

$ ls -l budget.txt
-rw----r-- 1 root it 8 may 24 09:15 budget.txt
```

- Que puede hacer ada sobre budget.txt?
  - a) Leer y escribir
  - b) Leer
  - c) Nada

Pregunta

```
$ id ada
uid=1002(ada) gid=1003(ada) groups=1003(ada),27(sudo),1005(it)

$ ls -l budget.txt
-rw----r-- 1 root it 8 may 24 09:15 budget.txt
```

- Que puede hacer ada sobre budget.txt?
  - a) Leer y escribir
  - b) Leer
  - c) Nada

Permisos de ficheiro

- Problema: moi limitado
- Solución: ACLs

ACLs de ficheiro

- Permisos con maior granularidade
- Permiten especificar outras usuarias e grupos

```
$ ls -l file-acls.txt
-rw-rw----+ 1 ada ada 0 ago  6 09:08 file-acls.txt

$ getfacl file-acls.txt
# file: file-acls.txt
# owner: ada
# group: ada
user::rw-
user:margaret:r--
group::rw-
group:managers:rw-
mask::rw-
```

Atributos de ficheiro

- Algúns atributos restrixen operacións:
  - Append
  - Immutable

```
$ lsattr
-----a--------e------- ./append.txt
----i---------e------- ./immutable.txt

$ echo "aaa" > append.txt
bash: append.txt: Operation not permitted
$ echo "aaa" >> append.txt
```

- So modificables por root ou CAP_LINUX_IMMUTABLE

Capabilities

- Problema en Linux: root o nada
- Solución: Capabilities

- Permisos para realizar operacións privilexiadas sen ser root.

- Por defecto:
  - root ten todas as capabilities.
  - O resto de usuarias non teñen ningunha.

Capabilities

- Hai bastantes:
  - CAP_NET_BIND_SERVICE : Escoitar debaixo do 1024
  - CAP_SETUID: Cambiar o uid
  - CAP_CHOWN: Cambiar a propietaria dun ficheiro
  - Moitas mais

- RTFM!! aka man capabilities

Capabilities

- Funcionan similar ao setuid..

```
$ nc -lvp 80
nc: Permission denied

$ sudo setcap cap_net_bind_service=ep /bin/nc.openbsd

$ nc -lvp 80
Listening on 0.0.0.0 80
```

Capabilities

- .. pero mais complexo ao ter varios conxuntos:
  - Effective
  - Permitted
  - Inheritable
  - Bounding
  - Ambient

Pregunta

- Pode root escoitar no porto 80 se lle quitamos CAP_NET_BIND_SERVICE?

LSM (Linux Security Modules)

- Funcionan por hooks no kernel de Linux
- Cada un ten as súas propias características

- Dous tipos: major e minor
  - Major: so se pode usar un destes (SELinux, AppArmor, Smack, Tomoyo)
  - Minor: compatibles co resto

LSM (Linux Security Modules)

- Modulos de seguridade incluidos no kernel:
  - **AppArmor**: Sistema MAC
  - **Bpf**
  - **Capability**: Capabilities
  - **Integrity**
  - **Loadpin**: Controla ficheros de kernel
  - **Lockdown**: Controla el acceso al kernel
  - **SafeSetId**: Controla setuid
  - **SELinux**: Sistema MAC
  - **Smack**: Sistema MAC
  - **Tomoyo**: Sistema MAC
  - **Yama**: Controla ptrace

LSM (Linux Security Modules)

- Podemos ver os activos no noso sistema

```
$ cat /sys/kernel/security/lsm
lockdown,capability,landlock,yama,apparmor
```

Namespaces

- Illamento de recursos do sistema operativo
  - Usados en container

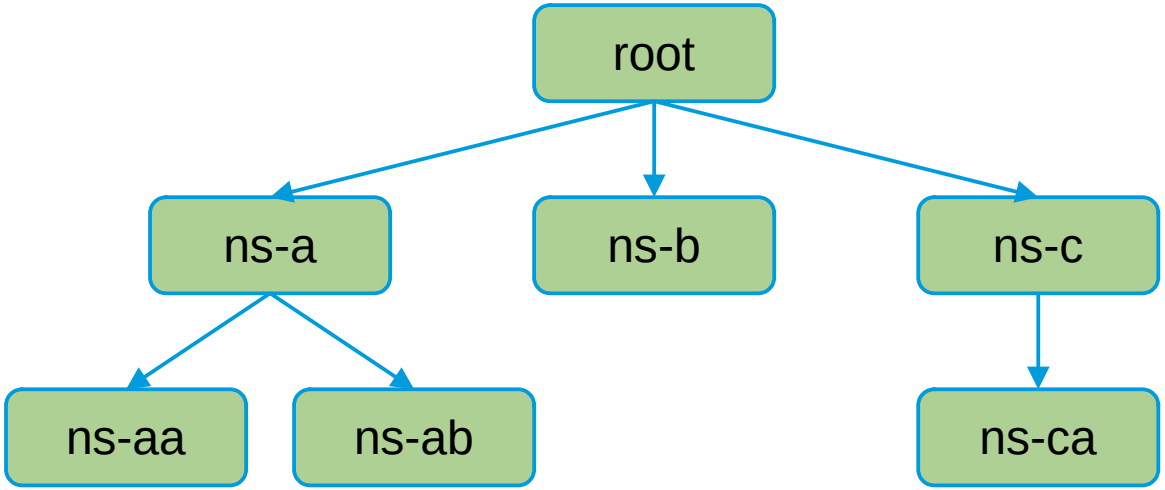
- Exemplo: Un proceso so pode ver as interfaces namespace de red ao que pertence.

Namespaces

- Varios tipos:
  - Cgroup
  - IPC
  - Network
  - Mount
  - PID
  - Time
  - User*
  - UTS

User namespace

- Namespace en árbore



User namespace

- Todo obxeto en Linux pertence a un user namespace
- O acceso ao obxeto ven determinado polo namespace
- Un usuario poder ser root nun namespace e non ter permisos noutro
- Usado en posman

chroot

- Cambia o directorio raíz
- Seguridade moi fráxil, é fácil escapar con privilexios

Seccomp

- Restricción de syscalls para un proceso
- Dous modos:
  - Strict → So permite read, write, _exit
  - Bpf → Permite crear filtros personalizados

Ata aquí chegamos

Graciñas