Capabilities (i atributs extesos)

¿Com és possible que un usuari no-administrador pugui canviar-se la seva pròpia contrasenya si no té permisos per poder editar l'arxiu "/etc/shadow"? ¿Com és possible que un usuari no-administrador pugui fer "pings" si per això cal obrir "sockets" de xarxa, cosa que només ho pot fer un administrador?

La forma tradicional de fer-ho possible era fent servir l'anomenat "bit SETUID". No obstant, poc a poc el bit "SETUID" està sent sustituït per una altra tècnica: les "capabilities" del kernel. Aquesta tècnica consisteix en assignar a un determinat programa (passwd, ping, etc) uns determinats "permisos d'administració" molts concrets (les "capabilities") que, un cop assignades, permeten a qualsevol usuari "normal" executar aquest programa i realitzar les tasques que necessitin els permisos d'administració assignats ¡sense escalar privilegis!

La gràcies de les "capabilities" és que n'hi ha moltes, i cadascuna només permet la realització de tasques d'administració molt concretes i cap altra. D'aquesta manera, si un programa té assignada la "capability A" però no la "B", quan un usuari normal executi aquest programa, només podrà realitzar les tasques d'administració relacionades amb la "capability A" però no pas amb les relacionades amb la "B". Per tant, aquest és un sistema d'assignació de permisos selectiu (depenent del programa en particular que estiguem estudiant), que evita l'ús de sudo o similars i, per tant, la possible presa de control del sistema total (com a molt, un "hacker" només tindria els permisos corresponents a les "capabilities" del programa "hackejat")

En principi, una "capability" no distingeix usuaris: està associada al programa de forma que qualsevol usuari tindrà per defecte la possibilitat d'executar el programa en qüestió amb les "capabilities" que tingui assignades sense cap restricció. De totes formes, amb l'ajuda de PAM es pot restringir l'ús de les "capabilities" presents en un binari per només oferir-les a determinats usuaris "normals" i no a uns altres. Això ho estudiarem més endavant.

Per conèixer les "capabilities" que ofereix el kernel actual als diferents programes, es pot executar la comanda *man capabilities*. De la llista que apareix allà, destacarem:

CAP_DAC_OVERRIDE	No té en compte els permisos de lectura, escriptura i execució
CAP_KILL	Deixa enviar qualsevol tipus de senyal a qualsevol altre procés
CAP_NET_BIND_SERVICE	Deixa associar un socket a un número de port menor que 1024
CAP_NET_RAW	Deixa generar paquets de tipus RAW o PACKET
CAP_NET_ADMIN	Deixa configurar les tarjes de xarxa, administrar el tallafocs i modificar les taules d'enrutament, entre altres
CAP_SYS_TIME	Deixa canviar la data del sistema i del rellotge RTC
CAP_SYS_ADMIN	Deixa (des)muntar sistemes de fitxers, gestionar la memòria swap, gestionar el sistema de quotes, establir el nom del host, etc
CAP_SYS_NICE	Deixa disminuir el valor NICE d'ell mateix o d'altres processos. També permet establir l'afinitat CPU i prioritats de temps real per ell i altres processos
CAP_SYS_MODULE	Deixa carregar i descarregar mòduls del kernel
CAP_SYS_RESOURCE	No té en compte el sistema de quotes ni els límits pam_limit
CAP_SYS_CHROOT	Deixa executar *chroot*
CAP_SYS_PACCT	Deixa executar *acct*
CAP_SYSLOG	Deixa realitzar operacions privilegiades sobre el registre del sistema
CAP_WAKE_ALARM	Deixa activar events que despertin el sistema

Cal saber també que cada procés té tres conjunts diferents de "capabilities" associades:

*Les de tipus "effective" : Són les que defineixen què pot fer realment un procés

*Les de tipus "permitted" : Són les que un procés pot arribar a utilitzar en el cas de que ho demani (fent servir les crides al sistema apropiades). És a dir, no permeten al procés fer res fins que aquest

no hagi demanat "formalment" al kernel poder-les fer servir. Això permet desenvolupar programes amb "capabilities" sensibles que només es tornen "effective" durant el temps que realment es necessitin

*Les de tipus "inheritable" : Són les que poden ser heredades pels eventuals processos fills del procés actual (dins del seu respectiu conjunt de capabilities "permitted")

Altres detalls:

- Les "capabilities" no funcionen amb enllaços, només amb binaris pròpiament dits

- Les "capabilities" no funcionen amb scripts. Una solució seria assignar la "capability" desitjada a l'intèrpret en sí (/bin/bash, /usr/bin/python, etc) però això és un potencial perill perquè llavors tots els scripts tindrien aquesta "capability" assignada. La solució més habitual en aquests casos és escriure un executable a part (en C, per exemple), donar-li la "capability" que volguem i llavors cridar-lo des de l'script.

- Les "capabilities" s'emmagatzemen dins del propi fitxer binari en un espai reservat pels anomenats "atributs extesos" (veure *man xattr* , *man lsattr* i *man chattr*). No obstant, si copiem un binari amb capabilities integrades a un altre sistema/partició diferent, aquestes capabilities es perden; aquest fet és volgut per una qüestió de seguretat (més en general, el que passa és que es perden les capabilities quan un fitxer canvia de número d'inode).

- Les "capabilities" es deshabiliten automàticament si el procés fa ús de les variables d'entorn LD_LIBRARY_PATH o LD_PRELOAD. Això és perquè un possible atacant podria substituir alguna de les llibreries estàndar del sistema per una altra amb codi maliciós i provocar així que els executables amb alguna "capability" perillosa invoquessin a aquesta llibreria impostora.

Per saber les "capabilities" que té assignat un determinat programa (és a dir,un binari executable), només cal executar la comanda:

getcap /ruta/programa

També es poden consultar les "capabilities" de tots els programes ubicats en un carpeta concreta (i de forma recursiva) amb la comanda:

getcap -r /ruta/carpeta

Per establir "capabilities" noves s'ha d'utilitzar la comanda *setcap*. S'ha d'especificar també el tipus de "capability" que volem que sigui ("e" per "effective", "p" per "permitted" i "i" per "inheritable"...cada "capability" pot ser dels tres tipus a la vegada). Per exemple...:

sudo setcap cap_net_raw=eip $(which ping)

...fa que la "capability" CAP_NET_RAW estigui assignada a l'executable ping i que sigui de tipus "effective", "permitted" i "inheritable". El símbol "=" fa que se sobreescriguin totes les possibles "capabilities" que tingués prèviament assignades el binari. Si només volem afegir o treure una "capability" concreta, es poden fer servir els símbols "+" o "-", respectivament.

Es pot assignar més d'una "capability" a la vegada al mateix binari separant-les per comes, així:

sudo setcap cap_net_admin+i,cap_net_raw=eip $(which ping)

Per esborrar totes les "capabilities" d'un executable (no es poden eliminar "capabilities" concretes), cal usar el paràmetre -r, així:

sudo setcap -r $(which ping)

Existeixen altres comandes relacionades amb la lectura de "capabilities" que es poden instal.lar si es necessiten gràcies al paquet "libcap-ng-utils" (a Ubuntu i Fedora). Concretament:

netcap : Mostra la llista de processos (PID, PPID, nom) fent servir sockets (de tipus TCP, UDP, RAW o PACKET, indicant a més el nº de port associat) que tinguin alguna "capability" (i mostra quines)

pscap : Mostra la llista de processos (PID, PPID, nom, usuari) amb alguna "capability" (i mostra quines)

getpcaps PID: Indica les "capabilities" pel procés concret indicat

EXERCICIS:

1.-a) Obre un terminal i executa això: *cp $(which ping) anotherping* . Tot seguit executa això *./anotherping 8.8.8.8* ¿Què passa?

b) Comprova les "capabilities" de l'executable ping i de l'executable *anotherping*. A partir d'aquí, dedueix la raó del vist a l'apartat anterior

c) Esborra l'executable *anotherping* i ara prova això: *sudo cp --preserve=xattr $(which ping) anotherping* ¿Ara funciona ./anotherping 8.8.8.8? Per què?

2.-a) Assigna la "capability" adient (en mode efectiu, permissiu i heredable) a la comanda *date* per tal de poder canviar l'hora del sistema sense haver de ser root. Comprova que ho has aconseguit provant d'executar la comanda *date -s 2:00* com a usuari normal

b) Assigna la "capability" adient (en mode efectiu, permissiu i heredable) a la comanda *ncat* per tal de poder posar a l'escolta un socket amb nº de port menor que 1024 sense haver de ser root. Comprova que ho has aconseguit provant d'executar la comanda *ncat -l -p 80* com a usuari normal

c) Assigna la "capability" adient (en mode efectiu, permissiu i heredable) a la comanda *find* per tal de poder recòrrer qualsevol ruta sense tenir en compte els permisos de lectura/execució de subcarpetes. Comprova que ho has aconseguit provant d'executar la comanda *find /etc -name "*" > /dev/null* com a usuari normal sense obtenir cap error.

Les "capabilities" no són més que un tipus d'"atributs extesos". Els "atributs extesos" són un conjunt de metadades emmagatzemades dins del propi fitxer al qual fan referència (veure *man xattr*). S'anomenen "extesos" perquè els atributs "estàndar" són els ja coneguts: permisos, propietari, etc però els primers (guardats en una altra zona del fitxer) inclouen altra tipus d'informació (la qual pot estar perfectament buida també).

NOTA: Cal saber, abans de res, però, si un sistema de fitxers determinat suporta els "atributs extesos" o no (està clar que si hem pogut realitzar els exercicis anteriors és que sí però de totes formes no va mal saber-ho). D'entrada, els sistemes de fitxers com el XFS o el BTRFS per disseny els suporten sempre i respecte el sistema Ext4, es pot comprovar (suposant que el sistema de fitxers es trobi a /dev/sda1) si la sortida de la comanda *tune2fs -l /dev/sda1 | grep "Default mount options"* mostra el valor "user_xattr". També es podria fer comprovant que apareix aquest valor a l'arxiu "/proc/fs/ext4/sda1/options"

En general, aquesta informació pot ser qualsevol però ha d'estar etiquetada dins d'una de les següents quatre categories: "user" (on un usuari estàndar pot introduïr dades personalitzades amb senzilles comandes que de seguida veurem), "system" (on es troben les ACL que estudiarem pròximament), "security" (on es troben les "capabilities" o també les regles AppArmor/SELinux que estudiarem pròximament) i "trusted".

Per introduir/modificar una metadada de la categoria "user" a un fitxer es pot fer:

setfattr -n user.nomMetadada -v "valorMetadata" /ruta/fitxer

Per esborrar una metadada es pot fer:

setfattr -x user.nomMetadada /ruta/fitxer

Per veure el nom i valor de totes les metadades de la categoria "user" existents en un fitxer es pot fer:

getfattr -d /ruta/fitxer

Per veure el nom i valor de totes les metadades de la categoria indicada existents al fitxer es pot fer:

getfattr -d -m {system|security|trusted|.} /ruta/fitxer*

Per veure els noms de totes les metadades de la categoria indicada existents al fitxer es pot fer:

getfattr -m {system|security|trusted|user|.} /ruta/fitxer*

Per veure el valor d'una determinada metadada existent al fitxer es pot fer:

getfattr -n categoria.nomMetadata /ruta/fitxer

La comanda *getfattr* també té el paràmetre -R per tal de veure les metadades de forma recursiva si s'indica la ruta d'una carpeta en comptes de la d'un fitxer.

3.-a) Executa *echo "Hola" > unfitxer.txt* . Executa *md5sum unfitxer.txt* . Afegeix al propi fitxer "unfitxer.txt" el resultat de la comanda anterior com a valor d'una metadada anomenada "user.md5sum" . Comprova amb *getfattr* que s'hagi introduït bé. Ara modifica el contingut d'"unfitxer.txt" com vulguis i torna a executar la comandes *md5sum unfitxer.txt* (haurà d'haver canviat el hash). Tot seguit executa la comanda *getfattr* adient per comprovar si la metadada "user.md5sum" ha canviat o no. Veient el que veus, ¿per a què creus que serviria en general aquest truc que acabes de fer en aquest exercici?

b) ¿Per què creus que es fa servir molt més la comanda *getcap* en comptes de la comanda *getfattr -d -m security* per tal d'esbrinar les "capabilities" que porta incorporades un determinat binari ?

D'altra banda, existeixen un petit conjunt d'atributs extesos (llistats a *man chattr)* que són de tipus "flag" (és a dir, que només ocupen un bit, "activat" o "desactivat", en comptes de les parelles nom<->valor que hem vist fins ara), els quals, però, només es troben definits a sistemes de fitxers Ext4. De tots els atributs d'aquest tipus que hi ha (els anomenarem "atributs físics"), els únics que ens interessaran són:

NOTA: Actualment els atributs físics també es troben definits a sistemes XFS i BTRFS, encara que no tots

i : Un fitxer amb aquest atribut no podrà ser modificat ni esborrat ni renombrat ni es podrà crear cap enllaç cap a ell. Només l'usuari root (o un procés amb la capability CAP_LINUX_IMMUTABLE) pot establir o treure aquest atribut

a : Un fitxer amb aquest atribut només podrà ser obert en mode "append" per escriure-hi en ell. És a dir, no podrà ser sobreescrit. Només l'usuari root (o un procés amb la capability CAP_LINUX_IMMUTABLE) pot establir o treure aquest atribut

A : Un fitxer amb aquest atribut no actualitzarà el seu atribut atime quan sigui accedit (seria equivalent a l'opció de muntatge "noatime", però només per a aquest fitxer)

S : Si un fitxer té aquest atribut, els canvis en el seu contingut es faran de forma síncrona al disc (seria equivalent a l'opció de muntatge "sync", però només per a aquest fitxer)

D : Si un directori té aquest atribut, els canvis en el seu contingut es faran de forma síncrona al disc (seria equivalent a l'opció de muntatge "dirsync", però només per a aquest directori)

Per veure els atributs físics d'un fitxer es pot utilitzar la comanda *lsattr {/ruta/fitxer|/ruta/carpeta}* (si s'indica una carpeta es veuran els atributs físics de tots els fitxers inclosos en ella). En aquest darrer cas, amb el paràmetre *-R* es pot obtenir la informació de forma recursiva per tots els fitxers inclosos a les subcarpetes; amb el paràmetre *-a* s'inclouen també els arxius ocults. El paràmetre *-d* serveix per mostrar els atributs físics de la carpeta pròpiament dita.

Per canviar els atributs físics d'un fitxer o carpeta (en aquest cas, opcionalment, de forma recursiva amb el parámetre *-R*) es pot usar la comanda *chattr {+|-|=}x {fitxer|carpeta}* on "x" representa l'atribut i els símbols "+","-" i "=" signifiquen "afegir", "treure" o "sobreescriure els atributs anteriors", respectivament.

4.-a) Executa *echo "Hola" > unfitxer.txt* i comprova amb la comanda *lsattr* quins atributs físics del llistats als paràgrafs blaus anteriors té per defecte "unfitxer.txt"

b) Estableix l'atribut "a" a "unfitxer.txt" i comprova-ho amb *lsattr*. Ara intenta executar la comanda *echo "Hola" > unfitxer.txt* ¿Pots? I si executes *echo "Hola" >> unfitxer.txt* , pots? Per què?

c) Treu l'atribut "a" anterior i ara estableix l'atribut "i" a "unfitxer.txt" Ara intenta executar la comanda *echo "Hola" >> unfitxer.txt* ¿Pots? I si intentes esborrar el fitxer, pots?