Control de Acceso

Identificación, Autenticación e Autorización

Exemplos Prácticos

- Atributos dos ficheiros
- Listas de control de acceso (ACL)
- Cambio de rol (sudo, runat)
- Límitación de acceso aos recursos (ulimit, cuotas de disco)
- Autenticación centralizada (Kerberos e Radius)

Atributos dos Ficheiros

Os atributos dos ficheiros limitan o que podemos facer con eles indicando certas características que o sistema de arquivos se encarga de forzar sin importar o grao de autorización do usuario. En Linux se xestionan cos comandos chattr e lsattr, mentres que en Windows se xestionan co comando attrib. Por exemplo

Linux

- Facemos un arquivo inmutable: **chattr +i arquivo**
- Podemos intentar eliminar o arquivo **rm arquivo**, movelo/renombralo (**mv arquivo destino**) ou modificalo e veremos que non é posible.
- Con **lsattr** obteremos os atributos que teñen os arquivos na carpeta actual
- Con **chattr -i** arquivo lle quitamos o atributo de inmutable
- Mais información: **man chattr**, **man lsattr**

Windows

- Abrimos un terminal
- Podemos ver os atributos de todos os arquivos escribindo **attrib** ou dun único ficheiro con **attrib ficheiro**
- Podemos ver a axuda do comando con **attrib /?**
- Podemos poñerlle ao arquivo o atributo de oculto con attrib +H arquivo
- Podemos quitarlle ao arquivo o atributo de oculot con attrib -H arquivo
- Mediante o botón de propiedades avanzadas do arquivo, se poden controlar outros atributos como **compresión** ou **cifrado**

Listas de Control de Acceso (ACL)

As ACL se utilizan en moitos ámbitos para controlar as autorizacións de acceso. Un dos máis comúns é o acceso aos elementos nos sistemas de arquivos. Windows xestiona as acl mediante o comando cacls mentres que Linux dispón dos comandos setfacl e getfacl.

Linux

- En Linux además dos permisos específicos do arquivo, tamén se aplican os permisos indicados nas ACL. Si os permisos non permiten o acceso, se comproban os permisos das ACL, si os permisos dan a autorización non é necesario.
- Comprobamos as ACL dun arquivo/directoro concreto **getfacl nome**.
- Indicamos que os membros do grupo www-data poden acceder a carpeta www para leer **setfacl -m g:www-data:rx**
- Eliminamos a ACL posta no punto anterior **setfacl -x g:www-data**
- Mais información **setfacl –help, getfacl –help, man setfacl, man getfacl**

Windows

- En Windows podemos controlar as ACL mediante a pestana de “Seguridade” das propiedades do arquivo ou co comando **cacls**
- Con **cacls arquivo** podemos ver as ACL atuais
- Con **cacls arquivo /G usuario:R** permitimos a usuario ler
- Con **cacls arquivo /G usuario:W** permitimos a usuario escribir
- Con **cacls arquivo /E /R Usuario**, eliminamos a acl para o usuario
- Tamén se pode expresar o permiso mediante descritores de seguridade expresados con ***SDDL*** (Security Description Definition Language) como as cadeas **ACE** (Access Control Entry): https://docs.microsoft.com/es-es/windows/win32/secauthz/ace-strings
- Máis información con **cacls /?**
- **cacls** está completamente substituída por **icacls**, que xestiona además o control de acceso obrigatorio

Cambio de rol: sudo e runas

Cando se traballa nun sistema sempre debemos seguir o **principio do mínimo privilexio**: *Calquera usuario ou sistema debe ter únicamente os mínimos privilexios necesarios para facer o seu traballo*.

Sen embargo, en algunhas ocasións é necesario poder realizar tarefas puntuais que requiren privilexios distintos dos que ten o usuario en curso. Para estos casos existen as ferramentas ***sudo*** (Linux) e ***runas*** (Windows).

sudo

sudo permite a un usuario *pertencente ao grupo sudoers* a execución de comandos baixo outro usuario segundo a configuración establecida en ***/etc/sudoers***. Este arquivo deberíamos editalo sempre co comando ***visudo*** xa que nos avisará de calquera erro de sintaxe e fará que os cambios sexan efectivos de inmediato. O formato é:

Usuario Host=(UsuarioDesexado:GrupoDesexado) [NOPASSWD:] Comandos Permitidos separados por comas

Como usuario podemos poñer o **nome**, ou **%grupo**

- **ALL** é un alias que significa “Todos”
- E posible definir alias para comandos (**Cmnd_Alias NOME_ALIAS = comando, comando, …**), para usuarios (**User_Alias NOME_ALIAS = usuario, usuario**…) ou para hosts (**Host_Alias NOME_ALIAS = host, host…**)
- Con **Defaults:ALL timestap_timeout=0** teremos que meter o noso contrasinal cada vez que usemos sudo, o tempo por defecto e de uns 3 minutos
- Con **Defaults:ALL passwd_tries = 2** limitamos a dous fallos na contrasinal antes de ter que escribir todo o comando de novo
- Con **Defaults logfile=/var/log/ficheiro** indicamos en que arquivo se gardará a auditoría de accesos con sudo
- Con **journalctl _COMM=sudo** veremos o historial de uso de sudo
- Con **usermod –lock –expiredate 1970-01-01 root** inhabilitamos o usuario root, o podemos reactivar con **usermod –unlock –expiredate 99999 root**

Cambio de rol: sudo e runas

Algúns sistemas como raspbian ou ubuntu delegan toda a administración do sistema en sudo, inhabilitando o usuario root. A idea é evitar de calquera maneira que un usuario traballe como root, e que poña “sudo” cando ten que realizar un traballo de administración. Sen embargo existen numerosas posibilidades de que unha mala configuración permita unha ***escalada de privilexios***.

Sempre se pode convertir un en root con ***sudo su –*** aínda que o usuario root esté “deshabilitado”.

RunAs

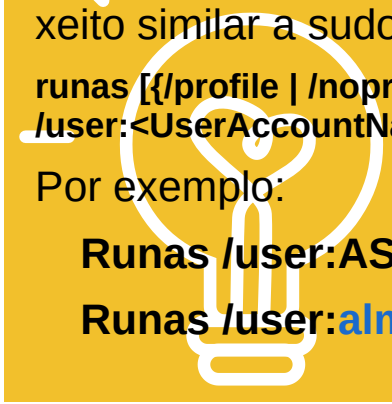
RunAs permite a un usuario executar utilidades e programas cos permisos de outro usuario, de xeito similar a sudo. O seu formato é:

runas [{/profile | /noprofile}] [/env] [{/netonly | /savecred}] [/smartcard] [/showtrustlevels] [/trustlevel] /user:<UserAccountName> "<ProgramName> <PathToProgramFile>"

Por exemplo:

Runas /user:ASIR\Administrator cmd

Runas /user:almudena@asir.iesrodeira.com “notepad carta.txt”



Control de Acceso aos recursos I - ulimit

Además da autorización para o uso de obxectos no sistema de arquivos é necesario controlar a autorización aos recursos que un usuario pode consumir: Cantidade de ram, número de procesos, uso de CPU, uso de disco… De non facelo, nos podemos exponer a que un usuario consuma demasiados recursos de xeito accidental ou maliciosa, por exemplo coa execución de ***fork_bombs.***

Mentres que nos sistemas monousuario como Windows isto non é un problema (un usuario se estaría facendo un DoS a si mesmo), en sistemas multiusuario como Linux ou Windows Server / Terminal Server pode ser un problema moi grave.

Linux dispón dun xeito moi completo de limitar os recursos que pode consumir un usuario mediante **ulimit** ou o uso de **cgroup e namespaces.**

cgroups (Control Groups) e namespaces

cgroups é unha característica do kernel Linux que limita, rexistra ou ailla o uso de recursos (CPU, memoria, uso de E/S, rede… etc) de un conxunto de procesos. Un cgroup é un grupo de procesos que se unen polo mesmo criterio e están asociados a un conxunto de límites. Os grupos son xerárquicos, de xeito que un grupo herda os límites do seu grupo pai. Os **namespaces** permiten separar grupos de procesos de xeito que os procesos dun namespace non poden ver os recursos de outro. Existen 6 tipos de namespaces: PID, Networks, UIDS, mounts, ipc, … cada proceso está unido a un namespace de cada grupo. Os cgroups e namespaces son a base dos contenedores como LXC ou Dockers. (man lsns, *proc*/PID/ns, man unshare, ip netns add, man nsenter)

Ulimit

O ficheiro /etc/sysctl.conf controla os límites por defecto para todo o sistema, podemos velos con sysctl -a (man sysctl) En **/etc/security/limits** podemos establecer os límites para os usuarios ou xestionalos con ulimit (man ulimit). Por exemplo, podemos establecer un límite de 1000 procesos para os usuarios dun grupo para evitar o efecto das fork_bombs.

Os límites “soft” se poden sobrepasar ata alcanzar o límite “hard”. Os límites “hard” non se poden sobrepasar.

Os límites se aplican no inicio de sesión.

Control de Acceso aos Recursos II – Cuotas de Disco

Mediante as cuotas de disco podemos restrinxir o uso de disco que poden facer os usuarios en canto a número de bloques (1k por bloque) e inodes (cada inode é un arquivo).

Os límites poden ser hard ou soft, o límite soft se pode sobrepasar so durante un periodo de gracia establecido e nunca sobrepasar o límite hard.

Os límites sempre son **por dispositivo** de bloques, que deben estar montados coas opcións

jqfmt=vsfv0,usrquota=ficheiro_quota_usuario,grpjquota=ficheiro_quota_grupo

man quotacheck, man quotaon, man quotaoff, man edquota, man quota, man repquota, man warnquota

Autenticación Centralizada: Kerberos

Kerberos almacena os obxectos a autenticar (usuarios, hosts e servizos) en unha base de datos. Estes obxectos reciben o nome de **principais** e terán asociada unha chave. No caso dos usuarios a chave será un texto manual, mentres que para os servicios e hosts o axeitado e crear unha chave longa ao azar que se almacenará logo na máquina cliente correspondente no seu ***keytab***, deste xeito tanto a máquina cliente como o servidor Kerberos coñecerán a chave.

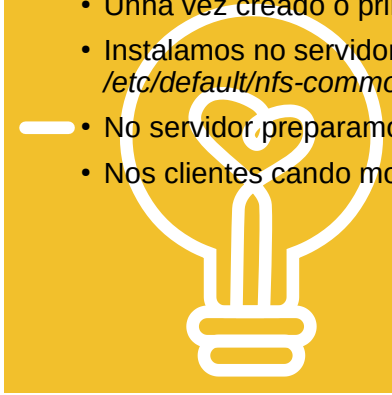
- Configurar hostname krbserver, e IP fixa segundo a política establecida
  - *Se suxire un script que configure calquera VM mediante SSH (bash vmname hostname fqdn ip)*
- Asegurarnos que o nome se resolve ben con hostname -s e hostname -f e facendo ping
  - Se aconsella engadir ao /etc/hosts do equipo de traballo.
- apt install krb5-admin-server krb5-kdc ntpdate
  - Preguntará polo realm e os servidores kerberos
  - ***man krb5.conf***
- A configuración está en */etc/krb5.conf.* Debemos editala e engadir o mapeo de dominio a realm (sección [domain_realm])
- krb5_newrealm, e editamos as acl para darlle permiso de administración ao principal root/admin (vim */etckrb5kdc/*kadm5.acl)
- Nos conectamos localmente ao administrador Kerberos, para poder crear o usuario administrador
  - listprincs
  - kadmin.local
  - addprinc root/admin
  - listprincs
- A partir deste momento poderemos conectarnos dende calquera cliente Kerberos co comando **kadmin**. Esto é práctico porque permitirá incorporar fácilmente ao cliente a chave dos principais creados para o mesmo no seu propio keytab.
- Os clientes deben ter instalado krb5-user e krb5-config. A configuración /etc/krb5.conf debe ser igual a do servidor kerberos
- Si queremos autenticar os usuarios contra Kerberos deberemos instalar libpam-krb5 e engadir a *etc pam.d* / common_auth a liña **session required pam_mkhomedir.so skel=/etc/skel/ umask=0022.** Como non temos información do usuario deberemos crear a liña correspondente en /etc/passwd, o máis simple é facelo con **adduser**

Autenticación Centralizada: Kerberos

Autenticando Servizos: NFS

NFS é o sistema máis empregado en Linux para a compartición de directorios e arquivos. E un sistema rapidísimo que se utiliza en diversos ámbitos. O deseño deste protocolo e moi antiguo, e debe parte da súa velocidade a falta de controis de autenticación dos usuarios que acceden aos recursos. En NFS, os usuarios teñen os privilexios no sistema de arquivos remoto que lles otorga a UID/GID que teñen no sistema local, o que supon un grave problema de seguridad paliado so polas opcións ***root_squash*** e ***all_squash***

- Para autenticar un servizo, o servizo ten que ser dado de alta en kerberos cun principal. A convención é o uso de *servizo/fqdn*. No noso caso **nfs/servidornfs**
- O alta do principal é mellor facelo dende o proveedor do servizo, xa que será máis simple engadir a súa chave kerberos ao *keytab*
- O principal debe ter unha password xerada automáticamente
- Os clientes deben tamén ser dados de alta en Kerberos. O principal no caso das máquinas ten habitualmente o nome *host/fqdn*. No noso caso **host/clientenfs**.Tamén conven dalas de alta dende cada cliente, e cunha password xerada automáticamente (***addprinc -randkey principal***)
- Unha vez creado o principal engadimos a súa chave ao keytab mediante ***ktadd principal***
- Instalamos no servidor ***nfs-kernel-server*** e nos clientes ***nfs-common***. Debemos editar */etc/default /nfs-kernel-server* e */etc/default/nfs-common* para o seu correcto funcionamento con NFSv4 nos clientes e no servidor.
- No servidor preparamos os shares NFSv4 con autenticación kerberos e reiniciamos o servizo *nfs-kernel-server.*
- Nos clientes cando montemos o recurso nfs especificaremos ***-o sec=krb5***



Autenticación Centralizada: Radius

Radius é un protocolo para implementar servizos ***AAA*** *(Authorization, Authentication and Accounting)* empregado masivamente, sobre todo nas redes WiFi con seguridade *WPA2 Enterprise*. *Radius* proporciona autenticación contra un servicio externo ou unha base de datos, autorización de uso dos recursos e mantén un rexistro de acceso e uso.

Existen interfaces de xestión como ***daloradius***, un panel web que permite configurar e xestionar un servidor Radius.

O protocolo RADIUS é o que empregan os Access Points (Puntos de acceso) para solicitar a autorización e autenticación ao servidor Radius. **Os clientes Radius son os puntos de acceso** (puntos de acceso WiFi, Switches, Routers … etc). O servidor Radius dispón de diversos módulos que lle proporcionan múltiples xeitos de xestionar a autenticación, como o uso de *Kerberos* ou *Active Directory*, entre outras.

- **apt install freeradius**
- Editamos */etc/freeradius/3.0/users* que é a base de datos de autenticación en texto plano que se utiliza por defecto e */etc/freeradius/3.0/clients.conf* para configurar os clientes (AP) que poden utilizar o servizo
- Paramos o servizo con **systemctl stop freeradius** e o lanzamos para “testing” con **freeradius -X**
- O servidor radius acepta peticións por defecto ao **porto 1812**. No cliente probamos con: **radtest usuario password ipservidorradius 1812 password-radius**. Debemos ter instalado no cliente **freeradius-utils**
- Paramos con control-c e xa podemos iniciar o servicio con normalidade con **systemctl start freeradius**