Linux Authentication and Access Control

Roy McElmurry

/etc/passwd

```
username:password_hash:uid:gid:gecos:home_dir:login_program
```

Examples:

```
root:x:0:0:root:/root:/bin/bash
roy:x:1000:1000:Roy McElmurry,,,:/home/roy:/bin/bash
```

- Username: the plaintext user name you chose
- Password: a hash of the password you chose
- Uid: a unique identifier used to identify you by the system
- Gid: the id of your primary unix group, this is applied to all your files by default
- Gecos: extra information about you, typically a comma separated list of name, building/room number, phone number and other contact info
- Home: Where in the file system to point the working directory on login
- Program: What program to run once you login

/etc/shadow

```
username:$id$salt$hash(password,salt):t1:t2:t3:t4:t5:t6
```

- Username: the plaintext user name you chose
- Password: a hash of the password you chose
  - Id: an identifier for the hash method used
    - 1=md5, 2=blowfish, 5=sha-256, 6=sha-512
  - Salt: very similar to a nonce
  - Hash: A hash of the password concatenated with the salt
  - “!”: no password, “!!”: password expired, “*”: account is locked
- T1: days since epoch of last password change
- T2: days until password change is allowed
- T3: days until password change is required
- T4: days to warn of expiration
- T5: days before account is inactive
- T6: days since epoch when account expires

/etc/securetty

- Cannot usually log in as root remotely
- Can only do this from access points defined in the /etc/securetty
- Use **su** to promote yourself to superuser status

PAM (Pluggable Authentication Modules)

- Authentication schemes are varied and ever-changing
- PAM allows programs to use trusted modules for authentication
- PAM allows programs to easily mix and match various schemes for authentication
- PAM moves the authentication scheme out of the program and puts it in the hand of the system administrator

PAM Config Example

```
cat /etc/pam.d/sudo

#%PAM-1.0

@include common-auth

@include common-account

session required pam_permit.so

session required pam_limits.so
```

```
cat /etc/pam.d/common-auth
# /etc/pam.d/common-auth - authentication
# settings common to all services
# This file is included from other service-
# specific PAM config files,
# and should contain a list of the
# authentication modules that define
# the central authentication scheme for use
# on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).
# The default is to use the
# here are the per-package modules (the
"Primary" block)
auth    [success=1 default=ignore]
pam_unix.so nullok_secure
auth    requisite           pam_deny.so
auth    required            pam_permit.so
auth    optional
pam_ecryptfs.so unwrap
auth    optional            pam_cap.so
```

PAM modules

- PAM provides four management service interfaces
  - Auth: determines if the user is who they say they are
  - Account: determines if the user is allowed to use this service
  - Password: provides a way for the user to change their authentication credentials
  - Session: performs actions before and after the user is authenticated

PAM module examples

- pam_unix.so
  - Uses standard system call to read /etc/passwd and /etc/shadow
- pam_ftp.so
  - Interprets the user from an ftp request by splitting on @
- pam_krb5.so
  - Performs kerberos authentication for the user
- pam_ldap.so
  - Performs authentication against an ldap server

PAM module examples

- pam_chroot.so
  - Virtualize the root directory for user
- pam_env.so
  - Set environment variables for session
- pam_limits.so
  - Impose resource limits on a user's session
- pam_permit.so, pam_deny.so
  - Automatically accept or deny users

Access Control

- Authenticating users is not enough
- We need to enforce access restrictions on users
- A few basic models
  - Attribute-based access control: Users must prove that they possess the necessary attributes needed (anonymous authorization)
  - Discretionary access control: The owner decides who can do what (UNIX)
  - Mandatory access control: Specific access must be given to users for each resource (highly sensitive data)
  - Role-based access control: Transactions are allowed only if performed by someone in an allowed role (SQL)

ACL (Access Control List)

- Each object is bundled with a list of permission
- Each permission is a tuple of the form (user, operation)
- ACLs can be applied to anything
  - Files
  - Network ports
  - Domain connections

UNIX Permission Classes

- Owner: the creator of the file or directory
- Group: a set of users
- Other: everyone else

UNIX Permissions

- Read (1)
- Write (2)
- Execute (4)
- Typically these are written out in octal, one bit for each

Examples:

```
-rw-r--r--
drwxr-xr-x
-r-x------
```

Examples:

```
-rwsr--r-x
drwxr-sr-x
lr-x---r-t
```

UNIX Special Bits

- Setuid: When used in owner class elevates the user to the owner's privilege level for the duration of the process
  - What security issues arise?
- Setgid: Same, but when used in the group class
- Sticky bit: If specified, the OS will keep the executable code in memory for fast execution
  - What security issues arise?

SQL Roles and Users

```
create role staff;
create role adviser in role staff inherit;
create role teacher in role staff inherit;

create user crystal in role adviser password 'cat%m0nster';
create user marty in role teacher password 'fl#!';
```

- Think of roles as groups
- Users are just special cases of roles that can log in

SQL Grant

```
grant select, update on Courses, Faculty to staff;

grant select on Enrollment to teacher with grant option;

grant select, update, insert, delete on Enrollment to advisor;
```

- There are several privilege levels, the most common are
  - Select (read)
  - Update
  - Insert
  - Delete
- Can grant one or more privileges to any role
- Can give the user the ability to also grant privileges
- The system tracks that role x gave role y privileges to z

SQL Revoke

```
revoke delete on Enrollment from crystal;

revoke insert on Enrollment from adviser cascade;
```

- To take away privileges you just revoke them from a particular role
- Revoking privileges can be tricky and have unexpected consequences
- Revoking privileges may not prevent a user from performing such actions