External Identities

On operating system level and for Web applications

Jan Pazdziora
jpazdziora@redhat.com
Identity Management Special Projects, Red Hat

PV157 Autentizace a řízení přístupu @ FI MU
15th May 2017

Users in computer systems

- Systems used by single person or few people.
  - Work laptop, home PC, ...
- Multiuser systems.
  - Remote sessions for large numbers of users.
- Systems running applications.
  - Web applications that support authentication but do not run processes for individual users.

User identities in POSIX/Linux systems

- After logon, process (like `/bin/bash`) is started with particular user and group id (UID, GID).
- Processes running with different UIDs differentiate different users.
- UID is an integer, system tools try to show respective username.

```
UID      PID PPID C STIME TTY   TIME CMD
root       1    0 0 Apr28 ?     00:00:10 /usr/lib/systemd/systemd --switched-root
root       2    0 0 Apr28 ?     00:00:00 [kthreadd]
[...]
dbus     767    1 0 Apr28 ?     00:00:05 /usr/bin/dbus-daemon --system --address=
root    1118    1 0 Apr28 ?     00:00:00 /usr/sbin/sshd
lightdm 1143    1 0 Apr28 ?     00:00:00 /usr/lib/systemd/systemd --user
[...]
bob     1243    1 0 Apr28 ?     00:00:00 /usr/lib/systemd/systemd --user
bob     1246 1243 0 Apr28 ?     00:00:00 (sd-pam)
bob     1249 1168 0 Apr28 ?     00:00:00 /bin/sh /etc/xdg/xfce4/xinitrc -- vt
bob     1625 1311 0 Apr28 ?     00:11:22 hexchat -e
bob     1641    1 2 08:25 ?     06:42:14 /usr/lib64/firefox/firefox
bob     1684    1 0 Apr28 ?     00:00:01 /usr/libexec/gconfd-2
bob     1819 1444 0 11:42 pts/1 00:00:00 bash
```

Identity lookup

- Traditionally, user identities were stored in `/etc/passwd`.

```
root:x:0:0:root:/root:/bin/bash
dbus:x:81:81:System message bus:/:/sbin/nologin
lightdm:x:991:987::/var/log/lightdm:/sbin/nologin
```

- GNU C Lib uses Name Service Switch (NSS), other sources are possible:
  - `/etc/nsswitch.conf`

```
passwd: files                # implements the /etc/passwd mechanism
```

```
passwd: db files nisplus nis
```

```
passwd: db sss
```

  - FYI: Matching implementation in respective shared library: `libnss_files.so.2`, `libnss_sss.so.2`, ...

- System can resolve username even if not listed locally in `/etc/passwd`:

```
# getent passwd bob
bob:*:1346800007:1346800007:Robert Chase:/home/bob:/bin/bash
```

Identity lookup from external sources

- No syncing of (potentially huge) user databases across machines.
- User record looked up on the fly.
- New user can log in to workstation immediatelly after being added to central database.
- Dependent on network availability of the external source.
  - Caching to the rescue.
- After (during) logon with username, the goal is to figure out the UID of the user, to run their processes with.

- System Security Services Daemon (SSSD).
  - In contemporary Linux distributions and FreeBSD.

Not just UIDs

- User group identities and membership also need to be captured and resolved.

```
# id bob
uid=1346800007(bob) gid=1346800007(bob)
          groups=1346800007(bob),1346800002(editors),1346800000(admins)
# getent group editors
editors:*:1346800002:bob,david
```

- Also hosts.
- Services.
- Domain Name System (DNS) records.
- They are essential for access control.

Authentication

- Operating system can run processes with any UID it choses.
- It needs to protect *itself* before starting processes with user's identity.
- Authentication in computer systems — verifying / proving identity.
  - Run locally.
  - Run locally, based on externally accessible database.
  - Based on trust.
- Configured on system level.
  - Sequence of Pluggable authentication modules (PAM) to try.
  - `/etc/pam.d/*` for individual services.
- Applications can use PAM exclusively or in addition to their own authentication steps.

Local authentication

- Password-based authentication:
  - Traditionally matched password against hash in `/etc/shadow`.

```
# ls -la /etc/shadow
----------. 1 root root 798 May 11 07:52 /etc/shadow
```

  - In PAM-based systems, implemented by `auth pam_unix.so`.
    - Uses setuid helper to verify password against the root-owned shadow file.
- SSH public key authentication:
  - SSH daemon consults `.ssh/authorized_keys` in user's home directory.
  - User lookup and determination of their home directory must have happened.
  - Tip: Use `ssh-keygen` to get yourself fresh key pair.

Local authentication, with external sources

- The goal is to avoid the need of copying authentication material (like password hashes) to machines.
- Password-based authentication:
  - Some NSS sources return the shadow entry.
    - Insecure.
  - Safer approach is to run authentication against the external system (LDAP bind, Kerberos “kinit”).
    - `pam_ldap.so`, `pam_sss.so`, … in `/usr/lib(64)/security`.
- SSH public key authentication:
  - OpenSSH supports `AuthorizedKeysCommand` — it can retrieve public keys by calling helper process.

Authentication based on trust

- User can authenticate against third party and bring a proof of identity.
- Kerberos:
  - Obtain ticket granting ticket (TGT) from Key Distribution Center (KDC) by authenticating to it.
    - It can happen automatically upon logging in to workstation/laptop.
  - The ticket is stored in local credentials (ticket) cache.
  - When accessing service, service ticket for that service is obtained from KDC based on the TGT.
    - Without user re-authenticating again.
  - Service decrypts the service ticket, verifies timestamp, uses principal information found.
  - Based on principal, user identity (and UID) is looked up.

Example of Kerberos workflow

```
client$ kdestroy -A
client$ klist
klist: Credentials cache keyring 'persistent:1346800007:1346800007' not found
client$ kinit bob
Password for bob@EXAMPLE.COM:
client$ klist
Ticket cache: KEYRING:persistent:1346800007:1346800007
Default principal: bob@EXAMPLE.COM
Valid starting       Expires              Service principal
04/28/2016 03:36:03  04/29/2016 03:35:59  krbtgt/EXAMPLE.COM@EXAMPLE.COM
client$ ssh bob@server.example.com
[bob@server]$ id
uid=1346800007(bob) gid=1346800007(bob) groups=1346800007(bob),1346800000(admins)
[bob@server]$ exit
Connection to server.example.com closed.
client$ klist
Ticket cache: KEYRING:persistent:1346800007:1346800007
Default principal: bob@EXAMPLE.COM
Valid starting       Expires              Service principal
04/28/2016 03:37:59  04/29/2016 03:37:56  host/server.example.com@EXAMPLE.COM
04/28/2016 03:37:57  04/29/2016 03:37:56  krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

Notes about Kerberos authentication

- Services have to be Kerberos principals:
  - `host/server.example.com`, `HTTP/www.example.com`, ...
  - Host's “password” is in `/etc/krb5.keytab`.
- Time needs to be synced.
- Authentication happens before identity (and UID) of user is determined.
- For users, it's effective single sign-on (SSO).
  - As opposed to “single password” nature of authentication with external sources (LDAP).
- Typically invoked via Generic Security Services API (GSS-API).
- When KDCs trust each other, they can issue service tickets based on TGT from the other forest and/or realm.
- Even hosts and services themselves can authenticate, not just users.

Authorization, access control

- Authorizing the session on operating system level:
  - PAM's account management group.
  - Configured in the same `/etc/pam.d/*` files as authentication.
  - Can invoke external sources of policies.
- Authorizing access to objects:
  - The `rwx` on/off bits allowing read, write, and execute for user, group, others.
  - POSIX' UID/GID of processes and objects on filesystem — shared namespace of UIDs.
  - Users' process can access their own data (UIDs match).
  - Problem: running software that should not have access to all user's data (daemons running as root, but also browsers, PDF viewers).

SELinux: processes

- Assign types to processes.
- Defined by SELinux policy, executed by kernel.
- Hint: use ps **Z** to display SELinux types of processes.

```
system_u:system_r:sshd_t:s0-s0:c0.c1023 root    /usr/sbin/sshd
un..._u:...:unconfined_t:s0-s0:c0.c1023 root    \_ sshd: root@pts/0
un..._u:...:unconfined_t:s0-s0:c0.c1023 root    | \_ -bash
guest_u:guest_r:guest_t:s0              tom     \_ sshd: tom@pts/1
guest_u:guest_r:guest_t:s0              tom     | \_ -bash
staff_u:staff_r:staff_t:s0-s0:c0.c1023  bob     \_ sshd: bob@pts/8
staff_u:staff_r:staff_t:s0-s0:c0.c1023  bob     | \_ -bash
system_u:system_r:sshd_t:s0-s0:c0.c1023 root    \_ sshd: alice [priv]
system_u:system_r:sshd_net_t:s0-s0:c0.c1023 sshd  \_ sshd: alice [net]
system_u:system_r:chkpwd_t:s0-s0:c0.c1023 root    \_ /usr/sbin/unix_chkpwd alice
system_u:system_r:NetworkManager_t:s0   root    /usr/sbin/NetworkManager --no-dae
system_u:system_r:dhcpc_t:s0            root    \_ /sbin/dhclient -d -q -sf /usr/
system_u:system_r:dnsmasq_t:s0          nobody  \_ /usr/sbin/dnsmasq --no-resolv
system_u:system_r:postfix_master_t:s0   root    /usr/libexec/postfix/master -w
system_u:system_r:postfix_qmgr_t:s0     postfix \_ qmgr -l -t unix -u
system_u:system_r:postfix_pickup_t:s0   postfix \_ pickup -l -t unix -u
```

SELinux: objects

- Assign types to files (and directories, network ports, sockets, ...).

```
-rw-r--r--. root  root   system_u:object_r:dnsmasq_etc_t:s0 dnsmasq.conf
drwxr-xr-x. root  root   system_u:object_r:dnsmasq_etc_t:s0 dnsmasq.d
drwxr-xr-x. root  root   system_u:object_r:NetworkManager_etc_t:s0 NetworkManager
drwxr-xr-x. root  root   system_u:object_r:postfix_etc_t:s0 postfix
----------. root  root   system_u:object_r:shadow_t:s0      shadow
drwxr-xr-x. root  root   system_u:object_r:etc_t:s0         ssh
drwxr-xr-x. root  root   system_u:object_r:cert_t:s0        ssl
```

- And to programs.

```
-rwxr-xr-x. root root system_u:object_r:sshd_exec_t:s0 /usr/sbin/sshd
-rwsr-xr-x. root root unconfined_u:object_r:chkpwd_exec_t:s0 /usr/sbin/unix_chkpw
```

SELinux: rules

- Explicit `allow` rules for processes of particular type to access only objects of given type.

```
# sesearch --allow -s sshd_t -t shadow_t
Found 2 semantic av rules:
   allow sshd_t file_type : filesystem getattr ;
   allow sshd_t file_type : dir { getattr search open } ;
# sesearch --allow -s chkpwd_t -t shadow_t
Found 1 semantic av rules:
   allow chkpwd_t shadow_t : file { ioctl read getattr lock open } ;
```

- Explicit transition rules to execute process with different type.

```
# sesearch --type -s sshd_t -t chkpwd_exec_t
Found 1 semantic te rules:
   type_transition sshd_t chkpwd_exec_t : process chkpwd_t;
```

- For example: network-listening SSH daemon (UID root, type sshd_t) is not allowed to read `/etc/shadow`; helper is enforced.

Further isolation options

- SELinux' Multi-Level Security (MLS)
  - Processes of the same UID and SELinux type can be differentiated by sensitivity and category.
- Namespacing support in Linux kernel
  - Mount (filesystems hierarchy)
  - Network (devices, IP addresses, routing)
  - Process IDs
  - User and group IDs
  - UTS (hostname, domainname)
  - IPC (SysV IPC, message queues)

Identities on the Web

- In Web applications, identity of user being served is rarely reflected by UID of process service the request.
- Simple cases:
  - List of users (login name, password) in application's database table.
  - Logon form, authentication by application code.
- Problems:
  - In large organizations where user identities are already managed in central directories, noone will maintain application-specific copies.
  - People do not want to remember another login and password.
  - Users prefer to log in once (say per day) — SSO needed.
  - With publicly accessible Web applications, multiple public identity sources required by users.

Authentication in Web applications

- Local authentication with external sources.
  - Language / framework-specific modules for connecting to other authentication sources.
  - Authentication by front-end server:
    - For example `mod_auth*` for Apache HTTP Server.
    - Application gets passed result of authentication: `REMOTE_USER`.
- Based on trust:
  - Kerberos (`HTTP/*` principals used)
  - Security Assertion Markup Language (SAML)
  - OpenID Connect
- After authentication, session is typically created, maintained by HTTP cookies.

Kerberos on the Web

- Based on HTTP status 401 for Negotiate/SPNEGO (RFC 4559).
- HTTP request to protected location:

```
GET /login HTTP/1.1
Host: www.example.com

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Negotiate
```

- Service ticket is obtained.
- Request retried with GSS-API data.

```
GET /login HTTP/1.1
Authorization: Negotiate YIICdwYJKoZIhvcSAQICAQBuggJmMIICYqADAgEFoQMCAQ...

HTTP/1.1 200 OK (or 302 Found)
WWW-Authenticate: Negotiate YIGZBgkqhkiG9xIBAgICAG+BiTCBhqADAgEFoQMCAQ+iejI
Set-Cookie: _session_id=de9be308edebb4432127fa39a10f4535; path=/; secure; H
```

Kerberos Web setup example

- Apache HTTP Server module: `mod_auth_gssapi`

```
<Location /login>
  AuthType GSSAPI
  AuthName "Kerberos Login"
  GssapiCredStore keytab:/etc/http.keytab
  require valid-user
</Location>
```

- Keytab file contains key(s) for `HTTP/www.example.com` principal.
- With GSS-Proxy, keytab file does not need to be accessible by the Apache HTTP Server processes.
- The Web server trusts information in the service ticket.

- After authentication, additional lookup of group membership or user attributes might be needed for application's access control.
  - OS-level setup (like SSSD) can be used.

SAML

- Identity Provider (IdP) — server that issues SAML assertions, based on user authentication.
- Service Provider (SP) — trusts the assertions.
  - Typically web server interested in having users authenticated by IdP.
- SAML assertions carry signed (by IdP) information about user identity.
  - And their attributes, like group memberships or roles.
- Metadata of IdP and SP need to be exchanged beforehand, when configuring the setup.
- Multiple HTTP redirects will be seen as user's browser is redirected to IdP to authenticate, and then back with the result.
- Tip: Apache HTTP Server module: `mod_auth_mellon`.

OpenID Connect

- OpenID Provider — server that issues claims, based on user authentication.
- Relying Party — trusts the claims.
  - Typically web server interested in having users authenticated by OpenID provider.
  - Mobile applications also supported.
- OpenID claims carry signed information about user identity.
  - And their attributes (OpenID Connect Scopes).
- Initial setup needed on OpenID Provider and the Relying Party.
- Multiple HTTP redirects will be seen as user's browser is redirected to OpenID Provider to authenticate, and then back with the result.
- Tip: Apache HTTP Server module: `mod_auth_openidc`.

Further resources

- FreeIPA server is integration of identity management solutions.
- SSSD is OS-level service for identity, authentication, and authorization operations.
  - Interoperability with FreeIPA, but also Active Directory servers, and general LDAP.
- Both are open source.
- Bc. and diploma topics available.

Conclusion

- In networked setups, identities are often managed in external, often distributed services.
- On POSIX operating system level, process UIDs and SELinux types represent the user identity.
- On the Web, unstructured, textual information is often used (username, groups).
- Multiple protocols for external authentication (with SSO) and authorization.
- Depending on protocol
  - The order of identity lookup and authentication can differ.
  - User attributes come included in ticket/assertion/claim or might need to be looked up.

References

- research.redhat.com
- diplomky.redhat.com
- pagure.io/SSSD/sssd/
- www.freeipa.org