External Identities

On operating system level and for Web applications

Jan Pazdziora
jpazdziora@redhat.com
Platform Security Readiness, Red Hat

PV157 Autentizace a řízení přístupu @ FI MU
May 2020 / online edition

Users in computer systems

- Operating systems
  - Workloads for different users are represented by their processes.
- Applications, Web applications
  - One process (from operating system point of view) can carry workload for multiple different remote users.

○ Note: open circles mark tasks or questions to figure out.

User identities in POSIX/Linux systems

- After logon, process (like `bash`) is started with particular user and group id (UID, GID).
  - Log in to aisa.
  - Figure out your own UID. (Try command `id`.)
- UID is an integer, system tools try to show respective username.
  - As what user does the ntpd process on aisa run? (Try command `ps`. It may require some parameters.)
  - As what UID does the ntpd process run? (Try command `id`.)

Identity lookup

- Traditionally, user identities were stored in `/etc/passwd`.
  - Check the content of `/etc/passwd` on aisa. Is ntpd listed there?
  - Is your username listed there?
- With Name Service Switch (NSS) in glibc, other sources are possible, configured in `/etc/nsswitch.conf`.
  - What are the `passwd` sources on aisa?
  - Run `getent passwd $FRIENDSLOGIN` to see friend's entry.
  - Matching implementation in respective `libnss_*` shared library.
    - Where are these libraries located on aisa? Where are they on nymfe machines?
- Bonus question: is it possible to run process with UID not listed in any of the sources?

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/server immediatelly after being added to central database.
- Dependent on network availability of the external source.
  - Caching potentially to the rescue.
- During/after logon with username, UID of the user needs to be determined, to be used for the session processes.

Not just UIDs

- User group identities and membership also need to be captured and resolved.
  - What is your group on aisa?
  - What are all user groups available on aisa?

- 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 — proving and verifying 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.
  - Check `auth` lines in `/etc/pam.d/sshd` on aisa. How does aisa authenticate users?
  - Compare to a nymfe machine.

Local authentication

- Password-based authentication:
  - Traditionally matched password against hash in `/etc/shadow`.
  - Can you find hashed root password in that file on aisa?
  - In PAM-based systems, implemented by `pam_unix.so`.
    - Where is it installed on aisa? Where is it on nymfe machines?
- SSH public key authentication:
  - SSH daemon consults `.ssh/authorized_keys` in user's home.
  - Use `ssh-keygen` to get yourself a fresh key pair.
  - Configure account on aisa to be able to log in with this key, without password, using `ssh -i ...`.

Local authentication, with external sources

- Goal: avoid copying authentication data (password hashes) around.
- Password-based authentication:
  - Some NSS sources return the shadow entry. Insecure.
    - Why is it insecure?
  - Safer approach is to run authentication against the external system (LDAP bind, Kerberos “kinit”).
    - What mechanism is used on aisa? And on a nymfe?

Authentication based on trust

- User can authenticate against third party and bring a proof of identity.
- Kerberos:
  - User obtains a ticket granting ticket (TGT) from Key Distribution Center (KDC) by authenticating to it.
  - It can happen automatically upon logging in.
    - Log in to aisa with password, avoid using public key this time: `ssh -o 'PubkeyAuthentication no' ...`
    - On aisa, run `klist`.
    - Log in from aisa to nymfe, again with `ssh -o 'PubkeyAuthentication no'`. Were you asked for password?
    - End the session on nymfe with `exit` and back on aisa, run `klist` again. What do you observe?

Authentication based on trust (cont'd)

- In fi.muni.cz network, you can get TGT with `kinit`.
  - Run as `KRB5_TRACE=/dev/stderr kinit` to see under the hood.
  - Clear credential cache with `kdestroy -A`.
- When accessing service, service ticket for that service is obtained from KDC based on the TGT.
  - Without user re-authenticating again.
  - Did a service ticket get created for you when logging in from aisa to nymfe?
- Service decrypts the service ticket, verifies timestamp, uses principal information found.
- Based on principal, user identity (and UID) is looked up.

Authorization, access control

- Authorizing 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.
    - Try `echo test > test; chmod u-r test; cat test`
  - POSIX' UID/GID of processes and objects on filesystem — shared namespace of UIDs.
  - Users' process can access their own data (UIDs match).
    - Can that be a problem?

Further isolation options

- SELinux
  - Access control mechanism independent of UIDs/GIDs.
  - Targeted policy: processes differentiated by their SELinux types.
  - Multi-Level Security (MLS) adds 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 with external sources

- Language / framework-specific authentication modules.
- Authentication by front-end server:
  - For example `mod_auth`* for Apache HTTP Server.
  - Application gets passed result of authentication: `REMOTE_USER`.
- Open new private window in your browser.
- Open network traffic logging (in Firefox it's in Tools > Web Developer > Network).
- Go to https://fadmin.fi.muni.cz/auth/. Observe what happened.
- Try again in new private window, this time cancel the logon attempt.

Authentication based on trust

- Multiple possible protocols:
  - 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).
- If you have account in Fedora Account System:
  - Get TGT: `kinit <yourfasloginname>@FEDORAPROJECT.ORG`
  - List it: `klist`
  - Make HTTP request to Fedora's build system logon location:
    `curl -v --anyauth -u : https://koji.fedoraproject.org/koji/login/`
  - Observe the first response status 401 and header
    `www-authenticate: Negotiate`
  - Notice request repeated with header `Authorization: Negotiate ...`
  - List Kerberos credential cache again: `klist`

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.
- OpenID claims carry signed information about user identity and attributes (OpenID Connect Scopes).
- Initial setup needed on OpenID Provider and the Relying Party.
- Tip: Apache HTTP Server module: `mod_auth_openidc`.
- Go to https://pv157-demo.fi.muni.cz/ and try OpenID Connect.
  - In Web Developer console, observe the HTTP redirects that happen.
  - Can you setup your own web application with similar federated authentication?

Conclusion

- In networked setups, identities are often managed in external, often distributed services.
- On POSIX operating system level, process UIDs 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.