SSSD and OpenSSH Integration

Jan Cholasta
01-04-2013
Introduction to OpenSSH

- OpenSSH is an implementation of the SSH protocol
  - Provides both server (sshd) and client (ssh)
- SSH allows secure access to resources on a remote system
  - Most commonly access to remote shell
- Both users and hosts are authenticated
  - Users are authenticated by the server (sshd)
  - Hosts are authenticated by the client (ssh)
User authentication in `sshd`

- SSH supports multiple mechanisms for authenticating users
  - Password authentication, public key authentication, GSSAPI authentication, …
- Public key authentication uses digital signatures to verify the user's identity
  - The user's private key is stored on the client (`~/.ssh/id_rsa`)
  - The user's public keys are stored on the server (`~/.ssh/authorized_keys`)
Host authentication in ssh

- Only public key authentication is supported for authenticating hosts
  - The host's private key is stored on the server (`/etc/ssh/ssh_host_rsa_key`)
  - Host names and their respective public keys are stored on the client (`~/.ssh/known_hosts`)
- When `ssh` connects to an unknown host, its identity must be manually verified by the user
  - When verified, the host is automatically added to the `known_hosts` file by `ssh`
Motivation

- OpenSSH can be already used with SSSD for user authentication
  - Password authentication is handled by PAM
  - Kerberos authentication is handled by GSSAPI
- However, OpenSSH does public key authentication on its own
  - No centralized management of public keys
  - No host authentication without user interaction
- Make public key authentication work with identity information stored in FreeIPA
SSH public keys in FreeIPA

- SSH public keys in FreeIPA are stored in LDAP attribute `ipaSshPubKey`
- User and host LDAP entries with object classes `ipaSshUser` and `ipaSshHost` can contain the attribute
- It is possible to configure SSSD to use a different attribute for SSH public keys
  - Configuration option `ldap_user_ssh_public_key`
  - Configuration option `ipa_host_ssh_public_key`
SSH public keys in FreeIPA example

- Set user's public key using `ipa` command:
  
  ```
  $ ipa user-mod user -sshpubkey='ssh-rsa AAAA...'
  ```
  
  (see “SSH Public Keys in FreeIPA” slides for more information)

- A user's LDAP entry with a SSH public key:
  
  ```
  dn: uid=user, cn=accounts, dc=example, dc=com
  objectClass: posixAccount
  objectClass: ipaSshUser
  ...
  uid: user
  ipaSshPubKey: ssh-rsa AAAAB3NzaC1yc2E......
  ...
Configure OpenSSH to work with SSSD (1)

- **Configure** `sshd` in `/etc/ssh/sshd_config`
  - Use PAM for password authentication
    ```
    UsePAM yes
    ```
  - Make sure we do not `kinit` ourselves and let SSSD do it
    ```
    KerberosAuthentication no
    ```
  - Get `authorized_keys` from SSSD
    ```
    AuthorizedKeysCommand
    /usr/bin/sss_ssh_authorizedkeys
    ```
    *(note that `AuthorizedKeysCommand` is available only in patched OpenSSH – available in RHEL and Fedora)*
- **Restart** `sshd`
Configure OpenSSH to work with SSSD (2)

- Configure `ssh` in `/etc/ssh/ssh_config`
- Get `known_hosts` from SSSD

```plaintext
GlobalKnownHostsFile /var/lib/sss/pubconf/known_hosts
ProxyCommand /usr/bin/sss_ssh_knownhostsproxy -p %p %h
```
Configure SSSD to work with OpenSSH

- Configure SSSD in `/etc/sssd/sssd.conf`
  - Append `ssh` to the `services` line in the `[sssd]` section
  - Create an empty `[ssh]` section if it does not exist
- Restart SSSD
User public key authentication with SSSD

1. sshd receives a public key authentication request
2. sshd executes `sss_ssh Authorizedkeys <user>`
3. `sss_sshAuthorizedkeys` asks SSSD to get the user's public keys from FreeIPA server
4. `sss_sshAuthorizedkeys` prints the public keys in `authorized_keys` format to its standard output
5. sshd reads and processes the output as if it was an actual `authorized_keys` file

(note that this requires patched OpenSSH - available in RHEL and Fedora)
Host authentication with SSSD

1. User executes `ssh <host>`

2. `ssh` executes `sssd_ssh_knownhostsproxy -p 22 <host>` to connect to the host

3. `sssd_ssh_knownhostsproxy` asks SSSD to get the host's public keys from FreeIPA server (LDAP, *not* DNS!)

4. SSSD adds the host's name and public keys to `/var/lib/sss/pubconf/known_hosts`

5. `sssd_ssh_knownhostsproxy` connects to the host and pipes all communication through its standard I/O

6. `ssh` processes SSSD `known_hosts` the same way as any other `known_hosts` file
Debugging the OpenSSH configuration

- Check that the required options have correct values in `sshd_config` and `ssh_config`.

- Debug `sshd`
  
  ```bash
  # /usr/sbin/sshd -D -ddd -p <port>
  (you must use the full path!)
  ```

- Debug `ssh`
  
  ```bash
  # ssh -vvv -p <port> -l <login> <host>
  ```
Debugging the SSSD configuration (1)

• Check that the `ssh` service is enabled in `sssd.conf` and the `sssd_ssh` process is running

• Check SSSD debug logs
  • Set the `debug_level` option in `[ssh]` and `[domain/<domain>]` sections in `sssd.conf`
  • Restart SSSD
  • Inspect `sssd_ssh.log` and `sssd_<domain>.log` in `/var/log/sssd`
Debugging the SSSD configuration (2)

- Run `sss_sshAuthorizedkeys` manually
  
  ```bash
  $ sss_sshAuthorizedkeys --debug 10 <user>
  ```

  - You should get a list of public keys for the user and no error messages
  - Check SSSD debug logs
Debugging the SSSD configuration (3)

• Run `sss_ssh_knownhostsproxy` manually

  • (opt.) Set `ssh_hash_known_hosts` option to `false` in the `[ssh]` section of `sssd.conf` and restart SSSD

    $ sss_ssh_knownhostsproxy --debug 10 -p <port> <host>

  • You should get a hello message from the server and no error messages (exit with Ctrl+C)

  • Check if `/var/lib/sss/pubconf/known_hosts` was updated with the correct information for the host

  • Check SSSD debug logs
Additional information

- OpenSSH manual pages
  - `sshd(8), sshd_config(5), ssh(1), ssh_config(5)`

- SSSD manual pages
  - `sssd.conf(5), sssd-ldap(5), sssd-ipa(5), sss_ssh Authorizedkeys(1), sss_ssh_knownhostsproxy(1)`

- “SSH Public Keys in FreeIPA” slides