FreeIPA Training Series

SSSD and SUDO integration

Pavel Březina

12-20-2012

Introduction to SUDO

- SUDO is a utility that allows a user to run a program as a different user (typically as root)
- Typical usage is to grant user the privilege to run a program with root permissions without the knowledge of root's password
- Who can run what on behalf of whom is specified by rules

SUDO rules sources

- Rules can be stored in file (/etc/sudoers) or in LDAP
- SUDO will look up rules in sources specified in /etc/nsswitch.conf, the database is called sudoers
- /etc/sudoers is used mainly for local users
- LDAP is used for centralized identity management

Motivation

- We use FreeIPA or other LDAP-based identity management solution
- We want to distribute SUDO rules across all machines in the domain – we will use LDAP as a source for SUDO rules
- Everything works fine... until the LDAP or network goes down

Motivation

- Because we can't access the LDAP server, we can't use SUDO
- Because we can't use SUDO we need to get the guy who knows root password to run network diagnostic tools...
- ...or we can use SSSD as a middleman between LDAP and SUDO
- SSSD will cache the rules and make SUDO work when offline

LDAP schema for SUDO rules

- sudoUser
  - Which user does this rule apply to
- sudoHost
  - Which machine does this rule apply to
- sudoCommand
  - Which command is the user allowed to run
- sudoNotBefore, sudoNotAfter
  - Defines the time span during which the rule is applied
- For more attributes please see `man sudoers.ldap`

SUDO rules examples

```
# this is a special rule that contains default options that are inherited by all rules
dn: cn=defaults,ou=sudorules,dc=example,dc=com
objectClass: sudoRole
cn: defaults
sudoOption: !requiretty

# allow to run all command on pbrezina.example.com by pbrezina
dn: cn=pbrezina-allow-all,ou=sudorules,dc=example,dc=com
objectClass: sudoRole
cn: pbrezina-allow-all
sudoHost: pbrezina.example.com
sudoUser: pbrezina
sudoCommand: ALL

# sudoHost is mandatory attribute for all rules that are not cn=defaults
```

SUDO rules in FreeIPA

- FreeIPA supports serving SUDO rules
- Traditional schema has limitations
- For better manageability FreeIPA uses a custom schema
- For compatibility with clients FreeIPA translates custom schema via a special compat tree
  - ou=sudoers,dc=example,dc=com
  - Not readable using anonymous bind
- SSSD does not support FreeIPA schema yet, only the standard schema exposed via compat tree

Configuring SUDO to work with SSSD

- You need to configure SUDO to use “sss” source for sudoers database in /etc/nsswitch.conf
- That's it!
- For example:

  sudoers: sss

  - This will force SUDO to use SSSD as its only data source

  sudoers: files sss

  - SUDO will use both /etc/sudoers and SSSD

Configuring SSSD to cache SUDO rules

- Add “sudo” to the “services” option in the [sssd] section of /etc/sssd/sssd.conf
- When using LDAP as backend
  - That's it!
- When using FreeIPA as backend
  - SSSD doesn't support FreeIPA as SUDO provider yet
  - You need to use FreeIPA provider for identity and LDAP provider for SUDO
  - You need to use authenticated channel to access SUDO rules on FreeIPA LDAP

Example configuration - SSSD with LDAP

```
[sssd]
config_file_version = 2
services = nss, pam, sudo
domains = EXAMPLE

[domain/EXAMPLE]
id_provider = ldap
ldap_uri = ldap://example.com
```

Example configuration - SSSD with FreeIPA server

```
[sssd]
config_file_version = 2
services = nss, pam, sudo
domains = EXAMPLE

[domain/EXAMPLE]
# standard FreeIPA configuration
id_provider = ipa
ipa_domain = example.com
ipa_server = ipa.example.com
ldap_tls_cacert = /etc/ipa/ca.crt

# configure SUDO and GSSAPI authentication
sudo_provider = ldap
ldap_uri = ldap://ipa.example.com
ldap_sudo_search_base = ou=sudoers,dc=example,dc=com
ldap_sasl_mech = GSSAPI
ldap_sasl_authid = host/hostname.example.com
ldap_sasl_realm = EXAMPLE.COM
krb5_server = ipa.example.com
```

How SSSD caches rules

- Keeping cached rules consistent with LDAP is critical
- SSSD performs three types of updates:
  - Full refresh
  - Smart refresh
  - Rules refresh
- SSSD stores all rules that apply to the machine

Full refresh

- Replace all cached rules with those currently available in LDAP server
- It is used to delete rules that are no longer present in the LDAP server
- Full refresh may be:
  - Periodical – once in several hours
  - Out of band – on demand of rules refresh

Smart refresh

- Smart refresh aims to keep the cache growing
- It periodically stores rules that are new or modified in the LDAP server
- It will **never delete** any rule from the cache
  - As a consequence, it will not detect change in sudoHost attribute such that the rule does no longer apply to the machine

Rules refresh

- When user runs SUDO, SSSD tries to refresh all rules that are expired and applies to this user
- Its purpose it to delete rules that are no longer present in the LDAP server so SSSD will not grant more permission that defined
- If any rule is deleted from the cache
  - SSSD will perform out of band full refresh
  - Because more rules that are not yet expired may have been deleted

Caching mechanisms summary

	Full refresh	Smart refresh	Rules refresh
When (default)	every 6 hours or when a rule is deleted from the cache	every 15 minutes	when user runs SUDO, rules expire after 90 minutes
Why	keep the cache consistent	store new rules	do not grant user more privilege
Operations	insert, delete	insert, modify	modify, delete
Expected traffic	large	small	small
Configuration option	ldap_sudo_full_ refresh_interval	ldap_sudo_smart_ refresh_interval	entry_cache_sudo_ timeout

Obtaining debugging information

- Enable SUDO log
  - In /etc/sudo.conf:
    - Debug sudo /var/log/sudo_debug all@debug
- Increase SSSD debug level
  - In /etc/sssd/sssd.conf in each section ([sssd], [sudo], ...)
    - debug_level = level
    - 0x3ff0 is very verbose level that will usually give us enough information

Additional information

- SSSD manual pages
  - sssd.conf
  - sssd-ldap
  - sssd-sudo
- SUDO manual pages
  - sudoers
  - sudoers.ldap