Introducing SSSD: You Should See Polyscheme PAM

by Lawrence Kearney

The ever increasing adoption of Linux in enterprise data centres has brought some of the scaling limitations of the Name Service Switch (NSS) and Pluggable Authentication Module (PAM) framework to the forefront for service implementers and system administrators. One of the most prominent symptoms of these limitations is the reduction of performance when servers are connected to heavily loaded user stores. Furthermore, these stores are becoming more specialised, such as those used for applications and identity management, and are increasing in numbers themselves. In summary, large numbers of servers are needed for large numbers of users across many stores.

The System Security Services Daemon, or SSSD, resulted from the effort to address some of these emerging issues. It's important to note that the SSSD extends NSS and PAM, it does not replace it. Providing the ability for a Linux server to securely use multiple user stores, improve authentication performance and increase features.

Specifically, the SSSD can provide robust authentication and caching services for thousands of users in different stores concurrently using a unified configuration, and integrate natively with those stores. The caching model not only reduces server authentication workloads significantly but also allows for true offline authentication services.

So why is the SSSD not at the centre of all the Linux administrator water cooler discussions? Well, basically because hardly anyone has heard of it. The primary reasons for this being that it has been quietly slipped into Linux distributions over the past two years and it has that horrible, horrible name.

If you want to know how to do things using the SSSD against both eDirectory and Active Directory back ends, please read on.

SSSD History

The beginnings of the SSSD lie in an open source project named FreeIPA (Identity, Policy and Audit). The primary goal of the project is to provide a truly interoperable identity management, authentication and authorisation solution for Linux, UNIX, Windows and Apple computers. The FreeIPA effort is actually a collection of projects designed to provide a solution framework, which includes, well, a FreeIPA client. Red Hat is a primary

	Relevant version information
v1.9	Active Directory identity provider and UID-GID/SID mapping capabilities added
v1.11	Enhancements in Active Directory access control added
v1.12	CIFS integration and group policy support added.

sponsor and development contributor to the parent project. Their development focus was to provide a client that was narrower in scope than the open source parent, but could still provide authentication and integration enhancements.

The features in 1.9, v1.11 and 1.12 are remarkable because they reduce or can even eliminate the need for windbindd. The Windows Berkeley Internet Name Domain daemon (windbindd) is used to provide access to NTFS shares and can be used to map Windows Security Identifiers (SID) to posix User identifiers (UID) and group identifiers (GID). ID mapping, share access and migration from configurations using windbindd will be the subject of future articles.

Preparing for the SSSD

The SSSD provides an extremely flexible and comprehensive service. As of this writing its most recent features include SID mapping and CIFS share integration. This article details the use of the SSSD for authentication and authorisation configurations on the RHEL 6.5 and SLES 11.2 platforms. Both use similar SSSD software versions.

That said, every Linux user must have a UID and a primary GID to successfully authenticate. Technically, UIDs and GIDs are numeric values that are resolved to user and group names by the system (NSS actually), and if a user only belongs to one group, that group is their primary GID.

The first objective is to decide where the UID and GID information will come from. These values can be stored in local and remote user stores. Both Novell Open Enterprise Server for Linux and Windows Server 2008 R2 have the ability to store and reference UID and GID attribute/value

pairs. Using the Linux User Management tools and the Identity Management for Unix role, respectively. Locally, the SSSD can create valid users and groups and store them in its own secure cache as well.

Consider your use case(s)

1. Will users be authenticating to an application running on the server?

   If so the successful comparison of the user credentials and group memberships may be all that is required to authenticate and authorise the user.

2. Will users be authenticating to an application running on the server that requires them to have UIDs and GIDs?

   Believe it or not this use case does occur. Consider using dynamic UID/GID mapping configurations and setting login shell values to "/bin/false" for these users. This would prevent authentication to the local server and the creation of home directories.

3. Will users be authenticating to the Linux server to access local resources (file systems, middle tier application configurations, sudo, autofs, etc.)?

   If so ensuring users have valid attribute values populated for login shells and home directories, in addition to their UIDs and GIDs, even if allocated dynamically, would be a requirement.

Software: the needed and the handy

What software is involved with this newfangled SSSD stuff? Like anything else, it depends.

LDAP only configurations: The most up-to-date LDAP libraries and a version of the openLDAP client (which is very useful for testing and trouble shooting) for your platform is recommended.

Active Directory configurations: When using LDAP/kerberos or the SSSD Active Directory identity provider, the most recent kerberos client, kerberos libraries and samba libraries for your platform are required to initially configure the SSSD (technically for domain joining). However, leaving them installed and configured will be useful for troubleshooting problems later.

Linux servers: Minimally the sssd and sssd-client packages. Recommended packages include sssd-tools and ldb-tools which provide the ability to use the SSSD to manage local accounts and the SSSD cache file records. If you implement SSSD on a large scale the ability to manage cache records without invalidating or deleting the entire cache for a user store can be useful.

The SSSD cache files use the "LDAP like Database" (LDB) file format which is identifiable by the file name extension of ".ldb". It is interesting to note that samba databases use the "Trivial Database" (TDB) file format for storing simple key/value pair records, and these two databases share a relationship. LDB introduces a LDAP like hierarchy to data stored using the linear TDB method. The result is the presentation of organised data records without a database application being required.

Modifying the NSS and PAM configuration files for a Linux server can be an unforgiving experience if done incorrectly. Fortunately both the RHEL and SLES platforms have tools that will make the required changes to these files in a way the system expects. They are named "authconfig" and "pam-config" and they are both SSSD aware on platforms SSSD is available on.

After installing the SSSD software, use the appropriate commands to add it to the NSS and PAM configuration files for your platform.

RHEL:

```
authconfig --enablesssd --enablesssdauth
--enablemkhomedir -updateall
```

SLES:

```
pam-config --add --sss --mkhomedir
```

Implementing the SSSD

So, the software is installed and we're ready to configure the SSSD. All SSSD configuration directives are managed in one file, "/etc/sssd/sssd.conf".

The SSSD configuration file structure:

[sssd] services = domains =userstore1	Global parameters, domain and monitored service responder identification
[nss], [pam], [sudo]… reconnection_retries = filter_users =	Monitored service responder parameters
[domain/userstore1] id_provider =ldap auth_provider =ldap	SSSD domain parameters

The minimum configuration requires a defined domain in the [sssd] section, and a configured "SSSD domain".

* SSSD Domain = Identity Provider + Authentication provider

Linux

Working with identity providers

Depending on the SSSD identity provider(s) used, and there are more than presented in this article, carefully consider whether the configuration directives used by the provider fulfill your use case and security needs. Many will implement features native to a particular store. Such as store auto-discovery, fail over, account expiration and password policy support to name but a few. Alternatively, even if connecting to a proprietary store using the LDAP identity provider, SSSD configuration directives could still be used to implement select features native to that store.

Novell eDirectory via LDAP

Because Novell eDirectory rarely incorporates a kerberos service (but it can), nearly all connectivity to these stores will use the LDAP identity provider. The SSSD does not operate without security so the server TLS/SSL infrastructure is assumed to be present and LDAP searches (using the ldapsearch utility perhaps) are successful using that security.

The example uses a local copy of the public key certificate from the eDirectory Certificate Authority, strict cipher requirements and incorporates local and remote accounts managed by the SSSD.

The configuration (as shown in figure 1) explained:

- The LOCAL domain, then the LDAP1 domain is searched for users
- The NSS and PAM responders allow three user login attempts per session (but never return "root")
- Implements a UID/GID range for local accounts and remote UID/GID values are not mapped
- All users are cached on query and user ID case is ignored
- Uses anonymous LDAP credentials for eDirectory access

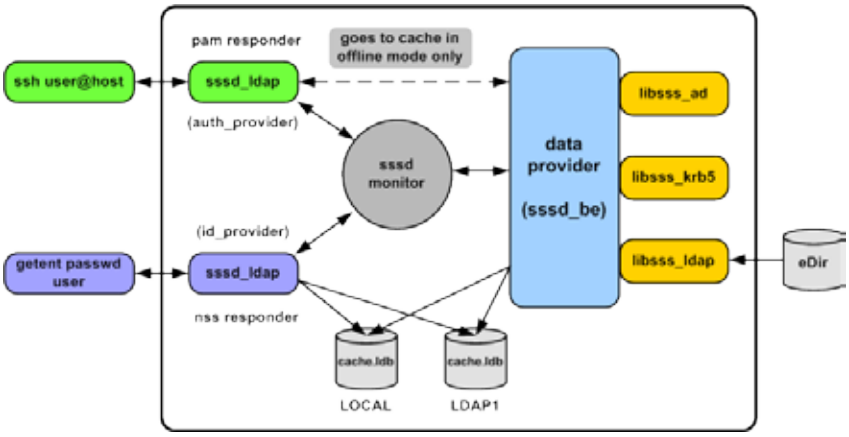

Figure 1: Using the SSSD LDAP providers

```
[sssd]
config_file_version = 2
services = nss, pam
domains = LOCAL,LDAP1

[nss]
filter_users = root
filter_groups = root

[pam]
reconnection_retries = 3

[domain/LOCAL]
cache_credentials = True
id_provider = local
min_id = 4000
max_id = 5000

[domain/LDAP1]
enumerate = False
cache_credentials = True
case_sensitive = False

id_provider = ldap
auth_provider = ldap
access_provider = ldap

ldap_schema = rfc2307bis
ldap_user_name = cn
ldap_user_member_of = groupMembership
ldap_group_name = cn
ldap_uri = ldap://darkvixen160.darkvixen.com
ldap_search_base = o=DVC
ldap_access_filter = groupMembership=cn=DARKVIXEN250_G,
                                        ou=LDAP,ou=SVS,o=DVC
ldap_tls_cacert = /etc/openldap/cacerts/DARKVIXEN160.crt
ldap_tls_cipher_suite = HIGH:MEDIUM:+SSLv2
ldap_tls_reqcert = Demand
ldap_id_use_start_tls = True
```

The /etc/sssd/sssd.conf file

- Allows users in the DARKVIXEN250_G group to authenticate to the server
- Enforces a high TLS/SSL encryption cypher

Microsoft Active Directory via LDAP and Kerberos

In this example access to Active Directory is implemented using LDAP and kerberos. Improved security is provided using the Simple Authentication Service Layer/Generic Security Service API (SASL-GSSAPI) method, which kerberos supports fully (instead of anonymous credentials).

Using SASL-GSSAPI provides a way for the Linux server to communicate securely with Active Directory using its native authentication framework. First, the kerberos client is used to request the initial kerberos tickets from a domain controller, and then the Linux server is joined to the target domain using samba utilities.

Implement a basic samba and kerberos client configuration to facilitate connectivity to a domain controller. Be sure to include the *security = ads* directive in the /etc/samba/smb.conf file and to use all uppercase characters for the *realm* in /etc/krb5.conf file.

Properly configured time synchronization and Domain Name Service (DNS) services are key to successful and reliable Active Directory authentication configurations.

First, we request a kerberos ticket for our joining operation using a privileged account

```
darkvixen250:~ # kinit DomainJoin
```

Enter password when prompted

Verify the domain controller is visible and responds correctly

```
darkvixen250 ~# net ads info
```

Join the domain and generate a keytab credential file (for SASL-GSSAPI connectivity)

```
darkvixen250 ~# net ads join -k
```

```
Using short domain name -- DVC
Joined 'DARKVIXEN250' to dns domain 'dvc.darkvixen.com'
```

Verify the following attribute values for the new Linux server computer object

distinguishedName:	CN=darkvixen250,CN=Computers, DC=dvc,DC=darkvixen,DC=com
dNSHostName:	darkvixen250.dvc.darkvixen.com
sAMAccountName:	darkvixen250$
servicePrincipleName:	HOST/DARKVIXEN250, HOST/darkvixen250.dvc.darkvixen.com

Verify the *sAMAccountName* attribute value is listed in the keytab file credentials

```
darkvixen250:~ # net ads keytab list
```

Destroy the current kerberos ticket cache used for joining

```
darkvixen250:~ # kdestroy
```

Obtain a new kerberos ticket using the server credentials

```
darkvixen250:~ # kinit -k DARKVIXEN250$
```

(Notice no password is required)

Test a secure LDAP search using the SASL-GSSAPI method

```
darkvixen250:~ # ldapsearch -H ldap://darkvixen160win.dvc.darkvixen.com
-Y GSSAPI -N -b "dc=dvc,dc=darkvixen,dc=com" -s sub cn="<Some_User>"
```

Noting the header of the results of a successful search operation

```
SASL/GSSAPI authentication started
SASL username: DARKVIXEN250$@DVC.DARKVIXEN.COM
SASL SSF: 56
SASL data security layer installed.
```

The /etc/sssd/sssd.conf file

```
[sssd]
config_file_version = 2
services = nss, pam
domains = LOCAL,dvc.darkvixen.com

[nss]
filter_users = root
```

```
filter_groups = root

[pam]
reconnection_retries = 3

[domain/LOCAL]
cache_credentials = True
id_provider = local
min_id = 4000
max_id = 5000

[domain/dvc.darkvixen.com]
cache_credentials = true
enumerate = false
case_sensitive = false
dns_discovery_domain = dvc.darkvixen.com

id_provider = ldap
auth_provider = krb5
access_provider = ldap

krb5_realm = DVC.DARKVIXEN.COM
krb5_server = darkvixen160win.dvc.darkvixen.com
krb5_kpasswd = darkvixen160win.dvc.darkvixen.com
krb5_validate = true
krb5_renewable_lifetime = 1d
krb5_lifetime = 1d

ldap_schema = ad
ldap_id_mapping = false
ldap_uri = ldap://darkvixen160win.dvc.darkvixen.com
ldap_user_search_base = dc=dvc,dc=darkvixen,dc=com
ldap_group_search_base = dc=dvc,dc=darkvixen,dc=com
ldap_disable_referrals = true
ldap_access_order = filter, expire
ldap_account_expire_policy = ad
ldap_access_filter = memberOf=cn=DARKVIXEN250_G,ou=LDAP,
                     ou=SVS,dc=dvc,dc=darkvixen,dc=com
ldap_sasl_mech = GSSAPI
ldap_sasl_authid = DARKVIXEN250$@DVC.DARKVIXEN.COM
```

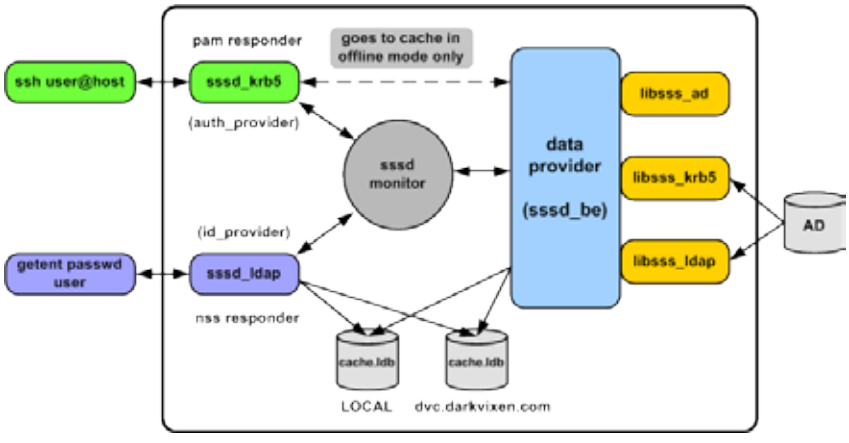

Figure 2: Using the SSSD LDAP and kerberos providers

The configuration (as shown in figure 2) explained:

- The LOCAL domain, then the dvc.darkvixen.com domain is searched for users
- The NSS and PAM responders allow three user login attempts per session (but never return "root")
- Implements a UID/GID range for local accounts and remote UID/GID values are not mapped
- All users are cached on query and user ID case is ignored
- Uses SASL-GSSAPI security for Active Directory access
- Does not allow expired accounts to authenticate
- Allows users in the DARKVIXEN250_G group to authenticate to the server

Microsoft Active Directory via AD identity provider

The SSSD Active Directory identity provider demonstrates significant improvements over its LDAP/kerberos predecessor in terms of functionality, performance and simplicity. Currently the Active Directory identity provider is not available on the SLES 11 SP2/SP3 platforms. However it is being included in SLES 12 and the author is an animated supporter of it being included in the SLES 11 SP4 distribution. That said the following configuration applies to RHEL 6.5 only.

Be sure to implement the same samba and kerberos client configurations, and perform the same domain joining operations as detailed in the LDAP/kerberos section.

The configuration (as shown in figure 3) explained:

- The LOCAL domain, then the dvc.darkvixen.com domain is searched for users
- The NSS and PAM responders allow three user login attempts per session (but never return "root")

The /etc/sssd/sssd.conf file

```
[sssd]
config_file_version = 2
services = nss, pam
domains = LOCAL,dvc.darkvixen.com

[nss]
filter_users = root
filter_groups = root

[pam]
reconnection_retries = 3

[domain/LOCAL]
cache_credentials = True
id_provider = local
min_id = 4000
max_id = 5000

[domain/dvc.darkvixen.com]
cache_credentials = True
enumerate = False

ad_server=darkvixen160win.dvc.darkvixen.com

id_provider = ad
auth_provider = ad
access_provider = ldap

ldap_schema = ad
ldap_id_mapping = False

ldap_sasl_mech = GSSAPI
ldap_sasl_authid = DARKVIXEN250$@DVC.DARKVIXEN.COM

ldap_access_order = filter, expire
ldap_account_expire_policy = ad
ldap_access_filter = memberOf=cn=DARKVIXEN250_G,ou=LDAP,
                     ou=SVS,dc=dvc,dc=darkvixen,dc=com
```

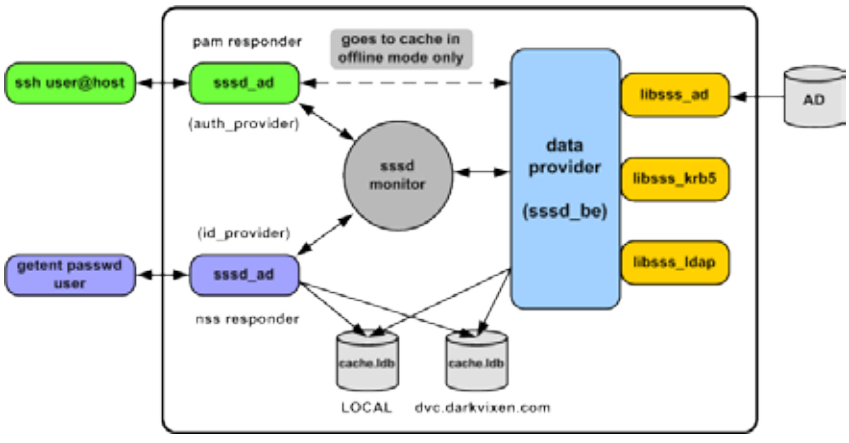

Figure 3: Using the SSSD Active Directory and LDAP providers

- Implements a UID/GID range for local accounts and remote UID/GID values are not mapped
- All users are cached on query and user ID case is ignored
- Uses SASL-GSSAPI security for Active Directory access
- Does not allow expired accounts to authenticate
- Allows users in the DARKVIXEN250_G group to authenticate to the server

Notice the numerous LDAP directives in the file are replaced with a few select directives set to a value of "ad". This is due to the development of the SSSD to natively understand Active Directory and its features. The "access_provider" however is still set to "ldap".

This is due to limitations in the v1.9 Active Directory access provider. It currently can only assess whether Active Directory accounts are expired. However, part of the flexibility of the SSSD configuration is that identity, authentication and access providers can be mixed to accommodate nearly any desired configuration.

So if the Active Directory access provider can't do it yet, the LDAP access provider can if configured properly.

User naming collisions

One of the most remarkable features of the SSSD is its ability to gracefully handle user naming collisions. Suppose all three of the SSSD domains in these examples existed in the same sssd.conf file (and they can).

Demonstrating this, the "domains=" directive in the [sssd] section would look like the example below.

```
[sssd]
config_file_version = 2
services = nss, pam
domains = LOCAL,LDAP1,dvc.
darkvixen.com
```

Now, suppose as well that a user ID of "MSTEELE" existed in each of those domains. If you only specified the user ID as "MSTEELE", logging in using SSH for example, the SSSD would locate the user in the "LOCAL" domain first and require those credentials to authenticate the user. The SSSD uses python regular expressions to construct the user ID formats it chooses to understand (which is configurable). The default results in it deciding that anything following the user ID and an "@" character, is the SSSD domain.

This means the user could be specified using any of the following formats (noticing the "case play") to log into any of the configured domains:

msteele@LOCAL
MSTEELE@ldap1
msteele@dvc.darkvixen.com

Not many services grant methods for managing collisions at all, never mind one this easy to configure and use.

What's next for the SSSD?

With all of the features the SSSD brings to the table it is hard to believe there is more in an already stuffed pipe. Except that there is. The RHEL7 and SLES12 platforms are incorporating more recent versions of the SSSD (v1.11) that include features that make it one of the best, and simplest to deploy, integration tools in the box. For example, when joining a server to a domain using the realmd application, the SSSD Active Directory identity provider is configured in the sssd.conf file automatically.

Additionally the Active Directory access provider supports LDAP formatted filters and basic group policies. Version 1.12, which is in the wild and sure to be included in server updates, also touts initial CIFS integration support. There may be valid use cases for windbindd out there for now, but SSSD is well on its way to becoming the one stop Linux integration shop we have all been hoping for.

Lawrence Kearney has over 20 years of experience supporting and designing enterprise academic and corporate computing environments in the USA. He is a member of the Novell TTP Advisory board and has experience with many Novell and SUSE Linux solutions. He coauthored the Novell GroupWise 8 Best Practice Wiki.