Local Accounts and Privileges in Likewise Storage Server

Rafal Szczesniak
EMC Isilon
Introduction to LSASS

Likewise Security and Authentication Subsystem Service

- Identity mapping
  - Name
  - SID/UID/GID

- Authentication

- Security
  - Local Accounts
  - Local Privileges
Local Accounts
What is an authentication provider?
Authentication Provider API

- Resolving *LSA_SECURITY_OBJECTs*
- Checking group memberships
- Authentication
- Modifying the underlying database contents (add/query/update/delete accounts)
Authentication Providers’ teamwork

- Each provider has its chance until operation succeeds

- **ERROR_NOT_HANDLED**
  - “I don’t know what you’re asking for” or
  - “I don’t know how to do that”

- Operation fails if one of providers returns some specific error code or all of them say “not handled”
Local Provider

- Supports local user and group accounts
- Supports well-known provisioned accounts (BUILTIN domain)
  - BUILTIN\Administrators – a way of making someone a superuser in the local system
  - Potentially any other provisioned account as needed
Local Provider (contd.)

- The provider does not store objects itself, storage layer handles this task
- Validation layer (thin but expanding) – Sam API
- Storage and more extensive validation layer - SamDb API
- Default implementation utilises external database for local accounts storage
- Provisioned upon the first local provider start
SAM database
SAM database

- Three types of object types stored:
  - Domain
  - User
  - Local Group

- One internal object class – Foreign Security Principal (for representing foreign SIDs)

- Windows account information stored
- Unix account information is optional (if account is unix-enabled)
SAM database (contd.)

- **SamDb API** provides basic operations
  - Find/Enumerate
  - Add/Update/Delete
  - Get/Set Membership
  - Query/Set Security
- Default **SamDb** implementation utilises **sqlite** backend for database operations
  - thread-safe
  - transaction-protected
- **Sam** API called by local provider and rpc server
SAM database (backend details)

- Three SQL tables
  - *SamDb* Objects
  - Local group memberships
  - Database state (version and id counters)
- Each *SamDb* call turned into SQL statement
- Executed as root (security checks are up to the caller)
Local Interface

- Lsa client library (lsa/lsa.h) - common for accessing all authentication providers
- `LsaOpenServer` to get the connection handle
- `LsaFindObjects` to return the security object(s)
- `LsaOpenEnumObjects, LsaEnumObjects, LsaCloseEnum` to enumerate objects
- `LsaAddUser2, LsaModifyUser2, LsaDeleteUser2`
- `LsaCloseServer` to close the connection
RPC interface

- Interface described in [MS-SAMR] (\pipe\samr)
- `UserInfo` and `AliasInfo` types accessible via `ACCOUNT_HANDLE` from built-in and local domains
  - `SamrLookupName`, `SamrLookupRid`
  - `SamrEnumUsers`, `SamrEnumAliases`
  - `SamrCreateUser`, `SamrOpenUser`
  - `SamrQueryUserInfo`, `SamrSetUserInfo`
  - `SamrClose`
RPC interface (contd.)
RPC interface (contd.)

- Interface described in [MS-LSAT] (\pipe\lsarpc)
- Name/SID resolver accessible via POLICY_HANDLE
  - LsarLookupName[23]
  - LsarLookupSid[2]
RPC interface (contd.)
Privileges
What are the privileges for?

- Securing functionality that cannot be protected with Security Descriptors
- Security by-pass for certain (typically system-level) operations
- Apply to the local system only
- Can be granted to any security object identifiable by a SID (e.g. domain user accounts)
What are the privileges for? (contd.)

Examples:
- SeChangeTimeZonePrivilege
- SeRemoteShutdownPrivilege
- SeBackupPrivilege
- ISI_PRIV_HTTP
- ISI_PRIV_SMB
Privileges and Lsa Accounts

- No multiple-provider model
- All \textit{LsaPriv}* client calls are handled by the Privileges (and Lsa Accounts) server
Privileges

- Stored in in-memory hash table mapping privilege names to values
- The database is initialised from two sources:
  - Builtin (hardcoded) table
  - Registry key
- The privilege value is represented by _LUID_ (Locally Unique Identifier) – in fact a 64-bit number

Lsa Accounts Storage
Lsa Accounts

- Lsa Account attributes:
  - Account SID
  - Privileges granted (fixed-size array)
  - System Access Rights granted
  - Security Descriptor
- Lsa Account is **not** the same as Local Account
Lsa Accounts (contd.)

- Stored in in-memory hash table holding `LSA_ACCOUNT` objects indexed by Lsa Account SID
- Hash table initialised from the registry database after the privileges table is ready
- All `LSA_ACCOUNT` updates in the hash table end up in the registry database as well
System Access Rights

- Bitmask controlling how the local system can be accessed
- Allow or Deny:
  - Interactive access
  - Network access
  - Remote Interactive access
- Limited use right now since Likewise server is accessed primarily via SMB/SMB2 (network access)
Local Interface

- Employs Account Rights concept referring to Privileges and System Account Rights collectively by their unique names (see [MS-LSAD])
  - LsaPrivsAddAccountRights
  - LsaPrivsEnumAccountRights
  - LsaPrivsRemoveAccountRights
RPC Interface

- Interface described in [MS-LSAD] (\pipe\lsarpc)
- Functionality accessible via
  LSAR_ACCOUNT_HANDLE:
    - LsaCreateAccount, LsaOpenAccount
    - LsaAddPrivilegesToAccount, LsaRemovePrivilegesFromAccount
    - LsaGetSystemAccessAccount, LsaSetSystemAccessAccount
- ... as well as POLICY_HANDLE and Account Rights semantics
The Big Picture
What do other parts of the system need
- Map Names/SIDs/UIDs/GIDs
- Create ACCESS_TOKEN
- Query group memberships

What parts?
- LWIO – I/O service and foundation for the multi-protocol fileserver
- SRVSVCC – SMB server configuration service
- Potentially every new service
Doing security checks with Security Descriptors requires the **Access Token** representing a user requesting an access.

- Access Token includes (among others):
  - `TOKEN_GROUPS` (all group memberships)
  - `TOKEN_PRIVILEGES` (privileges granted)

- Can be created from user name, SID, UID, GID, login principal or security context

- `LwMapSecurityCreateAccessTokenFrom*`
Before using, privileges in Access Token have to be enabled

Only a few are enabled by default (in the database)

Checking required privileges is possible by calling `RtlPrivilegeCheck` (just like `RtlAccessCheck` for security descriptor checks)
Name/SID information

- Other components often need to identify a security object (e.g. user, group, machine)
- Security object can be found by its name, SID, GID, UID
- \texttt{LwMapSecurityGetSidFromName}, \texttt{LwMapSecurityGetNameFromSid}
- \texttt{LwMapSecurityGetSidFromId}, \texttt{LwMapSecurityGetIdFromSid}
Thank you!

Questions?

Rafal Szczesniak
rafal.szczesniak@isilon.com