DELL EMC

ONEFS MULTIPROTOCOL SECURITY UNTANGLED

ABSTRACT

This paper describes the role that identity management, authentication, and access control play in the security system of the Dell EMC Isilon OneFS operating system. This paper covers OneFS 7.0 or later.

December 2016

Table of Contents

OVERVIEW

The Dell EMC® Isilon® NAS platform combines modular hardware with unified software to harness unstructured data. Powered by the distributed OneFS® operating system, a Dell EMC Isilon cluster delivers a scalable pool of storage with a global namespace and multiprotocol data access.

Multiprotocol data access, however, usually coincides with networks that contain several directory services, such as Microsoft® Active Directory® for Server Message Block (SMB) and Lightweight Directory Access Protocol (LDAP) for Network File System (NFS).

The OneFS operating system presents an Isilon cluster's file system as a single shared volume—a single *namespace*—that supports common data-access protocols, such as SMB and NFS. Linux machines access the Isilon cluster's file system using NFS; Microsoft® Windows® computers access the file system using SMB. The default shared directory of OneFS, which is ifs, lets clients running Windows, UNIX, Linux, or Mac OS X access the same directories and files.

To securely support NFS and SMB clients, OneFS does three main things:

- Connects to directory services, such as Active Directory, NIS, and LDAP, which are also known as identity management systems and authentication providers. A directory service provides a security database of user and group accounts along with their passwords and other account information.
- Authenticates users and groups. Authentication verifies a user's identity and triggers the creation of an access token that contains information about a user's identity.
- Controls access to directories and files. OneFS compares the information in an access token with the permissions associated with a directory or a file to allow or deny access to it.

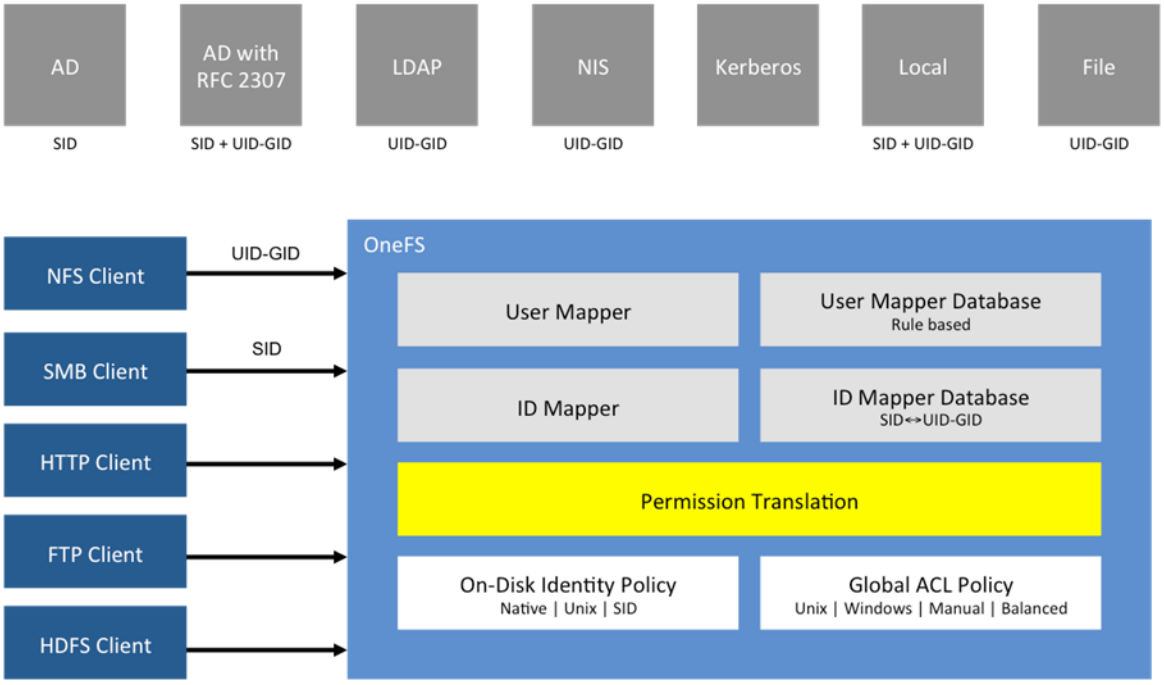
All three of these functions take place in an *access zone*—a virtual security context in which OneFS connects to directory services, authenticates users, and controls access to resources. By default, a cluster has a single access zone, which is known as the system access zone.

When a user connects to an Isilon cluster, OneFS checks the directory services to which the user's access zone is connected for an account for the user. If OneFS finds an account that matches the user's login name, OneFS verifies the user's identity—that is, it authenticates the user. During authentication, OneFS creates an access token for the user. The token contains the user's full identity, including group memberships, and OneFS uses the token later to check access to directories and files.

When OneFS authenticates users with different directory services, OneFS maps a user's account from one directory service to the user's accounts in other directory services within an access zone—a process known as *user mapping*. A Windows user account managed in Active Directory, for example, is mapped by default to a corresponding UNIX account with the same name in NIS or LDAP. As a result, with a single token, a user can access files that were stored by a Windows computer over SMB and files that were stored by a UNIX computer over NFS.

Similarly, because OneFS provides multiprotocol access to files, it must translate the permissions of Linux and Unix files to the access control lists of Windows files. As a result, a user who connects to the cluster with a Windows computer over SMB can access files that were stored on the cluster by a Linux user with NFS.

The following diagram summarizes how directory services (which are listed across the top of the diagram in the gray boxes), identity mapping, policies, and permissions play a role in the OneFS system of authentication and access control.



Identity management systems and directory services

OneFS works with the following directory services to authenticate users and control access to files:

- Microsoft Active Directory (AD), a popular directory service that uses domain controllers to authenticate users and authorize access to resources; to work with UNIX and Linux systems, Active Directory includes optional support for LDAP attributes with an implementation of RFC 2307.
- Lightweight Directory Access Protocol (LDAP), an application protocol for querying and modifying directory services.
- Network Information Service (NIS), a client/server directory service for distributing system information such as user names.
- Local users and local groups: unlike Active Directory, LDAP, and NIS, which are external systems, the local provider is an internal component of OneFS.
- File provider for working with accounts in /etc/spwd.db and /etc/group files; the file provider, which is also a OneFS component, lets you copy UNIX user and group account information from other systems.

You connect an Isilon cluster to a directory service by using either the OneFS web administration interface or the command-line interface. To view the directory services that a cluster running OneFS 7.0 or later is connected to, log in to a node and run the following command as root:

```
isi auth status

ID                                                Active Server                 Status
-------------------------------------------------------------------------------------
lsa-activedirectory-provider:YORK.HULL.EXAMPLE.COM yorkdns.york.hull.example.com online
lsa-local-provider:System                         -                             active
lsa-file-provider:System                          -                             active
lsa-ldap-provider:colorado                        -                             online
-------------------------------------------------------------------------------------
Total: 4
```

Joining an Active Directory Domain and Binding to an LDAP Server

In the OneFS user interfaces, directory services are often called authentication providers. The OneFS manual describes how to connect to, or join, an Active Directory domain, how to bind to an LDAP server, and how to set up other authentication providers. If you plan to join an Active Directory domain as well as bind to an LDAP server, see How to connect Isilon OneFS 7.x to LDAP and Active Directory at support.emc.com.
After you set up a directory service, OneFS authenticates all the connections from any protocol with the directory service. For example, after you connect the cluster to Active Directory, the users in Active Directory can access the cluster through any of the protocols that OneFS supports, including NFS, SMB, FTP, HTTP, and HDFS.

Active Directory with RFC 2307 and Windows Services for UNIX

In general, a best practice is to use Microsoft Active Directory with Windows Services for UNIX and RFC 2307 attributes to manage Linux, UNIX, and Windows systems. In some versions of Microsoft Windows, Windows Services for UNIX is also known as Identity Management for Unix (IDMU). Integrating UNIX and Linux systems with Active Directory centralizes identity management and eases interoperability. Make sure your domain controllers are running Windows Server 2003 R2 or later.

Authentication

OneFS uses the authentication providers that you have configured to verify a user's identity when the user logs on to the cluster. To authenticate a user, OneFS typically connects to a directory service, such as Microsoft Active Directory, with the Kerberos 5 security protocol to check the account and password—that is, the credentials—of the user with the directory service's security database.

Authentication involves the following subsystems and protocols. Some of the subsystems that OneFS uses vary by the operating system of the client computer:

THE LOCAL SECURITY AND AUTHENTICATION SUBSYSTEM, OR LSASS. It handles authentication and identity management as users connect to the cluster. If the lsass service fails, authentication falls back to /etc/passwd for SSH and console logins. To help troubleshoot authentication issues, you can view the daemon's log file at /var/log/lsassd.log.

THE KERBEROS 5 SECURITY PROTOCOL. You can set up Kerberos authentication by using either Microsoft Active Directory or a stand-alone MIT Kerberos 5 key distribution center. A best practice is to authenticate all users with Kerberos because it is a highly secure protocol. If you are authenticating users with Kerberos, make sure that both your Isilon cluster as well as your clients use either Active Directory or the same NTP server as their time source. With Kerberos, a clock skew among system components triggers authentication failures and other problems. Kerberos writes log messages to /var/log/lsassd.log and /var/log/lwiod.log. When Kerberos is used with NFS, Kerberos writes log messages to /var/log/nfs.log.

NT LAN MANAGER, OR NTLM. It is used only by SMB but can be backed up by the OneFS local provider, LDAP, and Active Directory. Both NTLM and Kerberos provide a user with a privilege attribute certificate, or PAC, during authentication. A PAC includes the group memberships of the user who requests access to a secure system object such as a file or a directory. NTLM writes log messages to /var/log/lsassd.log and /var/log/lwiod.log.

Access tokens

As OneFS authenticates the user, it generates an access token. The access token plays an instrumental role in the OneFS security system. The user mapper is one of several OneFS mechanisms that take part in building a token.

When you log on to an Isilon cluster, the user mapper expands your identity to include your other identities from all the directory services, including Active Directory, LDAP, and NIS. After the user mapper maps your identities across the directory services, OneFS generates an access token that includes the identity information associated with your accounts. A token includes the following *identifiers*:

- A UNIX user identifier, or UID
- A UNIX group identifier, or GID
- A security identifier, or SID, for a Windows user account
- A primary group SID for a Windows group account
- Supplemental identities listed by SID, UID, or GID

As a system administrator, you can view your identifiers by running the following command. This example shows the identifiers for the root account. Later sections of this paper dissect these identifiers for user accounts.

```
isi auth id

Ids
-------------------------------
ID Type          ID
-------------------------------
UID              UID:0
User SID         SID:S-1-22-1-0
GID              GID:0
Group SID        SID:S-1-22-2-0
On Disk User ID  UID:0
On Disk Group ID GID:0
Additional ID    GID:5
                 GID:20
                 GID:70
                 GID:10
-------------------------------
Total: 7
```

Privileges

An access token also contains privileges. A privilege is the right that you, as an administrator, assign to a role so that a user or group who is a member of the role can do something on the system, such as executing a command. On OneFS, an access token lists privileges that stem from administrative role-based access control, or RBAC. When you run the isi auth id command to view your identifiers, OneFS also displays your privileges. Here is an abridged example:

```
Privileges
------------------------------------------------------
ID                      Name               Access
------------------------------------------------------
ISI_PRIV_LOGIN_CONSOLE  Console            Read only
ISI_PRIV_LOGIN_PAPI     Platform API       Read only
ISI_PRIV_LOGIN_SSH      SSH                Read only
ISI_PRIV_SYS_SHUTDOWN   Shutdown           Read only
ISI_PRIV_SYS_SUPPORT    Support            Read only
ISI_PRIV_SYS_TIME       Time               Read only
ISI_PRIV_AUTH           Auth               Read/write
ISI_PRIV_ROLE           Privilege          Read/write
ISI_PRIV_ANTIVIRUS      Antivirus          Read/write
ISI_PRIV_AUDIT          Audit              Read/write
------------------------------------------------------
```

Generating an access token

On OneFS, four system processes play a role in generating a token:

1. Looking up a user in the directory services to which OneFS is connected
2. Associating the user's *identifiers*—that is, *ID* mapping—across the directory services
3. Combining access tokens from different directory services—that is, *user* mapping
4. Determining an on-disk identity

ID mapping

Although their names are similar, the ID mapping service differs from the user mapping service. The goal of the ID mapping service is to map Windows SIDs to UNIX UIDs and GIDs and vice versa in order to provide consistent access across protocols.

During authentication, the ID mapping service associates Windows identifiers with UNIX identifiers. When a user connects to an Isilon cluster over SMB, the ID mapping service maps the user's SIDs to UIDs and GIDs for access to files that were stored over NFS. By default, the ID mapping service matches accounts with the same name.

In contrast to SMB, when a user connects to the cluster over NFS, the ID mapping service does not map the UID and GIDs to SIDs by default, because the default identity stored on disk is already in the form of a UID and GIDs. Thus, with NFS, mapping the UID and GIDs to SIDs is unnecessary.

The following example illustrates the default ID mapping for a Windows user who authenticates with Active Directory, but has a corresponding account with the same name in LDAP. Because the user also has an account in LDAP with the same name, OneFS by default maps the user's Windows SID to the user's UID in LDAP and adds the UID to the token. (The formatting of an access token looks slightly different with OneFS 7.0 or later than with OneFS 6.5 or earlier; as this paper focuses exclusively on OneFS 7.0 or later, the following token is from OneFS 7.0. It has been abridged to remove some immaterial information.)

```
isi auth mapping token --user=york\\stand
         User
           Name : YORK\stand ❶
            UID : 4236 ❷
            SID : S-1-5-21-1195855716-1269722693-1240286574-591111 ❸
        On Disk : 4236
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
          Name : YORK\york_sh_udg
            GID : 1000000 ❹
            SID : S-1-5-21-1195855716-1269722693-1240286574-66133
Supplemental Identities
           Name : YORK\sd-york space group
            GID : 1000002
            SID : S-1-5-21-1195855716-1269722693-1240286574-579109
```

❶ The primary user name, which in this case comes from Active Directory

❷ The UID of the account for the user named stand in LDAP.

❸ The user's SID from Active Directory.

❹ Because the primary group account is from Active Directory, and no matching group name was found in LDAP, OneFS automatically allocates a GID to the group from the 1,000,000 to 2,000,000 range. OneFS also assigns a GID to the group in the list of supplemental identities.

You can manage some aspects of ID mapping by using the isi auth mapping command; see the OneFS Command Reference and the OneFS Administration Guide.

No overlapping ID ranges

In networks with multiple identity sources, such as LDAP and Active Directory with RFC 2307 attributes, you should ensure that UID and GID ranges do not overlap.

Without using RFC 2307 attributes, Active Directory contains no UIDs and GIDs. OneFS, however, relies on UIDs and GIDs as the primary form of identity. To store data in a multiprotocol environment with several directory services, OneFS maps an identity from Active Directory to a UID and a GID. If need be, OneFS automatically allocates a UID and a GID.

It is also important that the range from which OneFS automatically allocates UIDs
and GIDs does not overlap with any other ID range. The default range from which OneFS automatically allocates a UID and GID is 1,000,000 to 2,000,000.

If UIDs and GIDs overlap across two or more directory services, some users could gain access to other users' directories and files.

Avoid common UIDs and GIDs

In addition, you should not use well-known UIDs and GIDs in your ID ranges because they are reserved for system accounts. UIDs and GIDs below 1000 are reserved for system accounts; do not assign them to users or groups.

User mapping

While the ID mapper links account identifiers like UIDs and SIDs across directory services, the user mapping service combines access tokens from different directory services into a single token. When the names of an account in different directory services match exactly, OneFS automatically combines their access tokens into a single token. For example, the user mapping service maps, by default, a Windows account named YORK\jane from Active Directory to a UNIX account named jane from LDAP and generates an access token that combines the group membership information from the two accounts. OneFS also automatically maps two group accounts with exactly the same name.

The user mapper lets you combine and control a user's identities. After the user mapper collects a user's identities from the directory services, the user mapper can modify the token by applying a set of rules—rules that you, as a system administrator, can create. By manipulating tokens with rules, you can address use cases common to environments with several directory services:

- Merging several identities into a single token that works for access to files stored over both SMB and NFS. The token can include supplemental groups from both Active Directory and LDAP
- Selecting a primary group when there are competing choices from Windows and UNIX
- Managing identities when Active Directory and LDAP serve as authentication providers; for example, you can authenticate with Active Directory but use a UNIX identity for file access

You can address each of these use cases by creating user mapping rules through the web administration interface or the command-line interface to control how OneFS associates user and group accounts, or you can let OneFS manage the mappings with its default settings.

In an environment with two or more identity management systems, the simplest configurations name users consistently so that each UNIX user corresponds to a similarly named Windows user. Before assigning a UID and GID, OneFS searches its other authentication providers, such as LDAP, for other identities with the same name. If OneFS finds a match, the mapping service by default selects the associated UID and group memberships. Naming users consistently also allows user mapping rules with wildcards to match names and map them without explicitly specifying each pair of accounts.

For more information on the user mapping service and how it merges identities, see Identities, Access Tokens, and the OneFS User Mapping Service.

On-disk identity

After the user mapper combines tokens, OneFS determines an authoritative identifier for the primary user and group—the preferred on-disk identity. Your configuration options for the on-disk identity are UNIX, SID, and native. When you upgrade from a previous version of OneFS, the on-disk identity is by default set to UNIX, which was the default identity type on older versions of OneFS. The SID on-disk identity is for a homogeneous network of Windows systems managed only with Active Directory. On new installations, the on-disk identity is set to native.

Although you can change the type of on-disk identity, the native identity option is likely to be the best for a network with UNIX and Windows systems. In native mode, OneFS favors setting the UID as the on-disk identity because doing so improves NFS performance. OneFS stores only one type of identifier—either a UID and a GID or a SID—on disk at a time. As a best practice, if you change the on-disk identity, you should run the repair permissions job; see the OneFS Administration Guide.

You can view the on-disk identity by logging in to OneFS with SSH and running the following authentication mapping command. Here is an example in which the on-disk identity option is set to native. In this case, the user has no explicit mapping defined in an external directory service and no mapping stored persistently in the identity mapper database. As a result, OneFS sets the user's SID from Active Directory as the on-disk identity; the following output is truncated to focus on the on-disk identity:

```
isi auth mapping token --user=york\\janey
         User
           Name : YORK\janey
            UID : 1000000 ❶
            SID : S-1-5-21-1195855716-1269722693-1240286574-591111
        On Disk : S-1-5-21-1195855716-1269722693-1240286574-591111 ❷
```

❶ A UID that OneFS automatically generated because the user lacked it.

❷ The user's on-disk identity, which in this case is the SID from Active Directory. The SID, instead of the UID, is set as the on-disk identity because the on-disk identity type is set to native and because the UID was automatically generated by OneFS.

Note: All the examples, best practices, and use cases in this paper assume that the on-disk identity is set to native.

For more information on setting the on-disk identity, see the OneFS Administration Guide. The Dell EMC Isilon documentation portal includes additional best practices on working with several directory services. In particular, see the white paper titled "Dell EMC Isilon Multiprotocol Data Access with a Unified Security Model" on the Dell EMC website.

Access control

To control access, an Isilon cluster works with both the access control lists (ACLs) of Windows systems and the POSIX mode bits of UNIX systems. When OneFS must transform a file's permissions from ACLs to mode bits or from mode bits to ACLs, OneFS merges the permissions to maintain consistent security settings.

OneFS presents protocol-specific views of permissions so that NFS exports display mode bits and SMB shares show ACLs. You can, however, manage not only mode bits but also ACLs with standard UNIX tools, such as the chmod and chown commands, when you connect to an Isilon cluster with SSH. In addition, ACL policies that you can set through the web administration interface enable you to configure how OneFS manages permissions for networks that mix Windows and UNIX systems.

ACCESS ZONES. OneFS includes access zones that allow users from different authentication providers, such as two untrusted Active Directory domains, access different OneFS resources based on an incoming IP address. An access zone can contain multiple authentication providers and SMB namespaces.

RBAC FOR ADMINISTRATION. OneFS includes role-based access control (RBAC) for administration. Instead of a root or administrator account, RBAC lets you manage administrative access by role. For example, you can create separate administrator roles for security, auditing, storage, and backup.

Access zones

An access zone is a context that you can set up to control access based on an incoming IP address. The purpose of an access zone is to define a list of authentication providers that apply only in the context of a zone that contains SMB shares. As such, a key use case for an access zone is consolidating two or more Microsoft Windows file servers into a OneFS cluster.

But until you add an access zone, SMB shares and authentication providers operate in the default access zone, which is known as the *system access zone*. There is no requirement to create an access zone. NFS mounts cannot yet be segmented with zones.

An access zone can authenticate users with only one Active Directory domain. Although you can add more than one of the other directory services to a zone, a best practice is to limit each zone to no more than one of each of the directory services. User mapping rules apply only in the access zone in which you created them.

To view a list of access zones on your Isilon cluster, log in to a node with SSH and run the following command:

```
isi zone zones list
```

In the following output, only one access zone exists—the default zone named System:

```
Name
```

```
------
System
------
Total: 1
```

The following command shows the authentication providers for an access zone—for example, the default zone named System:

```
isi zone zones view --zone=System

                Name: System
          Cache Size: 4.77M
       Map Untrusted:
          SMB Shares: -
      Auth Providers: lsa-ldap-provider:ldapeast, lsa-activedirectory-
provider:EAST.EXAMPLE.COM, lsa-local-provider:System, lsa-nis-provider:nis1
      Local Provider: Yes
        NetBIOS Name:
      All SMB Shares: Yes
  All Auth Providers: Yes
  User Mapping Rules: -
Home Directory Umask: 0077
  Skeleton Directory: /usr/share/skel
             Zone ID: 1
```

Although you can create access zones to consolidate SMB shares, access zones are not NFS aware.

As a best practice, if you create access zones, make sure that the directory paths for each zone under ifs do not overlap. Instead, you should designate separate directory trees for each zone. Here is an example of separate directory trees:

Access Zone	Directory Tree
Zone 1	/ifs/data/zone1
Zone 2	/ifs/data/zone2

For more information on access zones and the how to manage them, see the OneFS Administration Guide and the OneFS Command Reference.

Administrative roles

With roles, you can assign privileges to users and groups to control administrative access. By default, only the root and admin users can log in to the command-line interface through SSH or the web administration interface through HTTP. The root or admin user can add other users to built-in or custom roles that contain the privileges that are required to log in and perform administrative functions. You should assign users to roles that contain the minimum set of necessary privileges. To create or assign roles, you must be logged on as a member of the security administrator role. For a list of roles and privileges, see the OneFS CLI Administration Guide.

Access checks with tokens and file permissions

A file contains permissions. On OneFS, the permissions appear as an access control list, or ACL, as well as POSIX mode bits. The ACL controls access to directories and files. When a user tries to access a file, OneFS compares the identities in the user's access token with the file's ACL. OneFS grants access when the file's ACL includes an access control entry, or ACE, that allows the identity in the token to access the file and does not include an ACE that denies the identity access.

You can visualize how OneFS compares the access token of a user with a file's ACE file by viewing a user's token and a file's ACL side by side. The following example shows a user's token on an Isilon cluster running OneFS 7. (All the commands and examples in this paper are from an Isilon cluster running OneFS 7.)

```
isi auth mapping token --user=MAINE-UNO\jsmith
         User
           Name : MAINE-UNO\jsmith ❶
```

```
            UID : 1000000
            SID : S-1-5-21-3542649673-1571749849-686233814-1117
         On Disk : S-1-5-21-3542649673-1571749849-686233814-1117
           ZID: 1
          Zone: System
   Privileges: -
Primary Group
          Name : MAINE-UNO\domain users
            GID : 1000000
            SID : SID:S-1-5-21-3542649673-1571749849-686233814-513
Supplemental Identities
           Name : MAINE-UNO\marketing ❷
            GID : 1000001
            SID : SID:S-1-5-21-3542649673-1571749849-686233814-1109
           Name : Users
            GID : 1545
            SID : S-1-5-32-545
            ------------------
```

And here is the ACL for a file on the cluster:

```
-rwxr--r-- + 1 MAINE-UNO\jsmith MAINE-UNO\marketing 2056 Feb 2 10:18
adocs.txt
OWNER: user:MAINE-UNO\jsmith
GROUP: group:MAINE-UNO\marketing
0: user:MAINE-UNO\jsmith allow ❸
file_gen_read,file_gen_write,file_gen_execute,std_write_dac
1: group:MAINE-UNO\marketing allow file_gen_read ❹
2: everyone allow file_gen_read
```

❶ In the token, the user's identity is MAINE-UNO\jsmith, which is an Active Directory account.

❷ The token also shows that the user is a member of the marketing group.

❸ In the ACL, an access control entry shows that MAINE-UNO\jsmith is allowed to access the file. The ACE also lists the user's permissions, such as file_gen_write, which is permission to write to the file.

❹ This ACE shows that members of the marketing group are allowed to read the file.

For general information about ACLs and ACEs, see MSDN. For information about what each OneFS ACE means, see Dell EMC Isilon Multiprotocol Data Access with a Unified Security Model.

Multiprotocol data access and permissions

To foster such multiprotocol data access, OneFS maps the POSIX mode bits of a file from a Linux or Unix system to the permissions model of the Windows system, and vice versa. To do so, Isilon uses several constructs:

- An on-disk identity that transparently maps to the right identifier for the requesting protocol.
- An on-disk authoritative representation of an object's permissions. Only one set of permissions—either ACLs or mode bits—is authoritative. The other set is a virtual approximation based on its internal representation.
- An internal representation of a file system object's permissions, such as a directory or a file, that contains information from either the POSIX mode bits or the ACL. The internal representation, which is partly based on RFC 3530, mediates between the access rights of the Windows security model and the mode bits of the POSIX security model. The internal representation helps enforce the same permissions across protocols.
- A synthetic ACL that approximates the mode bits of a UNIX file for an SMB client.
  A synthetic ACL illustrates the difference between a file's authoritative representation and its internal representation. The authoritative representation of a UNIX file stored over NFS version 3 is its mode bits; the file's internal representation is an estimation of the mode bits that, based on RFC 3530, is used to generate a synthetic ACL for SMB clients.
- A policy engine that transforms the authoritative representation of an object's permissions into a form that the requesting access protocol can process.

- ACL policies that optionally enable administrators to configure how OneFS manages permissions for different environments.
- A default ACL policy that, when a chmod command is run on a file with an ACL, optimally merges the new permissions with the old ones.

For a discussion of how OneFS maps permissions between the security models of Unix and Windows systems, see the paper titled “Dell EMC Isilon Multiprotocol Data Access with a Unified Security Model” on the Dell EMC website.

The Anatomy of a Cross-Platform Access Token

If you do not set up user mapping rules, a user authenticating with one directory service receives full access to the identity information in other directory services when the account names are the same. For example, a user who authenticates with an Active Directory domain as york\jane automatically receives identities for the corresponding UNIX user account for jane from LDAP or NIS.

In the most common scenario, OneFS is connected to two directory services, Active Directory and LDAP. In such a case, the default mapping provides a user with a UID from LDAP and a SID from the default group in Active Directory. The user’s groups come from Active Directory and LDAP. The user’s home directory, gecos, and shell come from Active Directory.

The following examples demonstrate how OneFS builds an access token for a Windows user who authenticates with Active Directory, but has a corresponding account with the same name in LDAP. No user mapping rules are in place.

First, you can view a user’s token from only Active Directory by running the following command and targeting the user’s Active Directory domain account. The output is abridged to remove some immaterial information.

```
isi auth users view --user=york\\stand --show-groups
                     Name: YORK\stand
                       DN: CN=stand,CN=Users,DC=york,DC=hull,DC=example,DC=com
               DNS Domain: york.hull.example.com
                   Domain: YORK
                 Provider: lsa-activedirectory-provider:YORK.HULL.EXAMPLE.COM
          Sam Account Name: stand
                      UID: 4326
                      SID: S-1-5-21-1195855716-1269722693-1240286574-591111
             Primary Group
                        ID : GID:1000000
                      Name : YORK\york_sh_udg
        Additional Groups: YORK\sd-york space group
                           YORK\york_sh_udg
                           YORK\sd-york-group
                           YORK\sd-group
                           YORK\domain users
```

Next, you can view a user’s token from only LDAP by running the following command and targeting the user’s LDAP account. The output is abridged.

```
isi auth user view --user=stand --show-groups
                     Name: stand
                       DN: uid=stand,ou=People,dc=colorado4,dc=hull,dc=example,dc=com
               DNS Domain: -
                   Domain: LDAP_USERS
                 Provider: lsa-ldap-provider:Unix LDAP
          Sam Account Name: stand
                      UID: 4326
                      SID: S-1-22-1-4326
             Primary Group
                        ID : GID:7222
                      Name : stand
        Additional Groups: stand
                           sd-group
                           sd-group2
```

When there are no mapping rules and when the user logs in to the cluster over SMB, OneFS authenticates the user with Active Directory and builds an access token that prioritizes the account information from Active Directory, but appends the supplemental groups from the UNIX LDAP token to the end of the final token:

```
isi auth mapping token --user=york\\stand
         User
           Name : YORK\stand ❶
            UID : 4326 ❷
            SID : S-1-5-21-1195855716-1269722693-1240286574-591111 ❸
        On Disk : 4326
          ZID: 1
         Zone: System
   Privileges: -
Primary Group
          Name : YORK\york_sh_udg ❹
            GID : 1000000
            SID : S-1-5-21-1195855716-1269722693-1240286574-66133
Supplemental Identities
           Name : YORK\sd-york space group ❺
            GID : 1000002
            SID : S-1-5-21-1195855716-1269722693-1240286574-579109
           Name : YORK\sd-york-group
            GID : 1000004
            SID : S-1-5-21-1195855716-1269722693-1240286574-475739
           Name : YORK\sd-workers
            GID : 1000003
            SID : S-1-5-21-1195855716-1269722693-1240286574-169779
           Name : YORK\domain users
            GID : 1000001
            SID : S-1-5-21-1195855716-1269722693-1240286574-513
           Name : Users ❻
            GID : 1545
            SID : S-1-5-32-545
           Name : sd-group ❼
            GID : 100001
            SID : S-1-22-2-100001
           Name : sd-group2
            GID : 100002
            SID : S-1-22-2-100002
```

❶ The primary username is from Active Directory

❷ The user's UID is from LDAP

❸ The user's SID is from Active Directory

❹ The primary group is from Active Directory

❺ These supplemental identities are from Active Directory, as indicated by the name of the domain before the name of the group

❻ The group named Users and its GID of 1545 is a built-in OneFS local group that comes from the OneFS local provider; it appears in the token by default because the OneFS operating system adopts the standard Microsoft Windows practice of adding the Domain Users group to the local users group when the system is joined to an Active Directory domain

❼ These last two groups are appended to the token from the UNIX LDAP token

The mapping service leaves out the user's LDAP primary group. You can add the primary group from LDAP to the final token by creating a user mapping rule.

By default, when you run the isi auth mapping command with a UNIX username, OneFS looks up the UNIX user's information from LDAP without mapping it to the UNIX user's Active Directory account information. Why? Because OneFS gives preference to using a UID to maximize NFS performance. If OneFS instead showed the information from Active Directory as well, the results of the command would have visual symmetry with the result of an isi auth mapping request for an AD user—which includes the information from LDAP. But the visual symmetry would come at the expense of NFS performance.

The Anatomy of a Cross-Platform ACL

On an Isilon cluster, each ACE in an ACL is presented as a single line prefaced by an index number, which starts at 0, and is followed by these parts:

- Identity: the identity to which the ACE applies
- Allow or deny: whether the ACE allows or denies the permissions listed in the ACE
- Permissions: a list of one or more permissions that are allowed or denied by the ACE
- Permission flags: flags that reflect the types of inheritance

The identity can be one of three types: user (listed as "user:"), group (listed as "group:"), or the special identity, everyone. For directories, it can also be one of two special template identities: creator_owner or creator_group. When present in the ACL of a containing directory, these template identities are replaced in the ACL of a newly created file system object with the specific user and group of the respective creator.

An ACE can optionally contain flags that specify whether it is inherited by child folders and files. Inheritance takes place when files and subdirectories are created; modifying an inherited rule affects only new files and subdirectories, not existing ones. The following flags specify the types of inheritance for permissions in the ACE:

- object_inherit: Only files in this directory and its descendants inherit the ACE.
- container_inherit: Only directories in this directory and its descendants inherit the ACE.
- no_prop_inherit: This ACE will not propagate to descendants (applies to object_inherit and container_inherit ACEs).
- inherit_only: The ACE does not apply to this object, but will apply to descendants when inherited. For example, when this flag is set on a directory, the ACE for the directory will not apply to the directory but will apply to its subdirectories.
- inherited_ace: The ACE was inherited.

Here is an ACL listing from OneFS that shows some of these components. The listing was obtained by running the ls command with an option (le) that Isilon added to show the ACL. The option is available only on the Isilon cluster, not on a UNIX client that has mounted an export. See the OneFS man page for the ls command. The plus sign that follows the POSIX mode bits indicates that the file contains an actual ACL, not a synthetic ACL.

```
ls -le bar.txt
-rw-r--r-- + 1 root wheel 0 Apr 22 17:23 bar.txt
  OWNER: user:root
  GROUP: group:wheel
  0: group:Administrators allow
 std_read_dac,std_synchronize,file_read_ext_attr,file_read_attr
  1: user:root allow file_gen_read,file_gen_write,std_write_dac
  2: group:wheel allow file_gen_read
  3: everyone allow file_gen_read
```

Modifying an ACL with chmod

OneFS provides methods to modify an ACL. Isilon added an option to the chmod command, for example, to manipulate ACL entries. You must connect to the cluster with SSH to run the following commands:

```
ls -lze
-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1
  0: user:guest deny file_read
  1: user:admin allow file_write

chmod +a# 1 group "domain users" deny file_read file1

ls -lze
-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1
  0: user:guest deny file_read
  1: group:"domain users" deny file_read
  2: user:admin allow file_write
```

As the examples illustrate, Isilon has added options to several commands that allow you to view and modify ACL permissions when you connect to OneFS by ssh:

Command and Option	Description
ls -le	The long format of the command that lists the contents of a directory, including the ACLs, owner and group information, and other security information.
ls –lze	The long format of the command that lists the contents of a directory plus the ACLs but not the owner and group information.
chmod +a	The command that changes a file's mode bits with the OneFS option to add a new ACE from the next argument of the command and insert the ACE into ACL.

For more information about the options, see the OneFS man pages for ls and chmod.

How OneFS handles chmod permissions changes

When a Windows client changes the permissions of a file with an ACL, no information is lost because the client reads the ACL, modifies it, and then writes it back to OneFS. The same holds true when a Windows client changes the permissions of a file with mode bits. Because OneFS maps the mode bits to a synthetic ACL and because the ACL model can capture the full range of POSIX permissions, no security information is lost. In such cases, OneFS replaces the file's synthetic ACL with an actual ACL that is equivalent to the mode bits.

The situation is different when a chmod or chown command modifies the permissions of a file protected by an ACL. OneFS must map the permission changes between two noncorresponding security models. To do so, OneFS, in its default setting, merges the ACL with the change in mode bits.

As an example, say that you want to run the chmod command to change the permissions of a file with an ACL from 764 (rwxrw-r--) to 744 (rwxr--r--). If you are connected to the Isilon cluster by ssh, you can check the file's on-disk permissions beforehand by running the ls -le command with the root account.

In the following example, the actual permissions are listed with the generic access rights for files and directories because the file was stored by a Windows client over the SMB protocol. OneFS approximates the mode bits as 764:

```
ls -le
-rwxrw-r-- + 1 MAINE-UNO\jsmith MAINE-UNO\market 2056 Feb 2 10:18
 adocs.txt
 OWNER: user:MAINE-UNO\jsmith
 GROUP: group:MAINE-UNO\marketing
 0: user:MAINE-UNO\jsmith allow
 file_gen_read,file_gen_write,file_gen_execute,std_write_dac
 1: group:MAINE-UNO\marketing allow file_gen_read,file_gen_write
```

```
 2: everyone allow file_gen_read
```

When you run the chmod command to change the permissions from 764 to 744, OneFS applies the difference between 764 and 744 to the ACL, removing the write permission of the primary group (in this case, marketing) while preserving the other permissions. The actual permissions, again listed as generic access rights, change as follows:

```
chmod 744 adocs.txt

ls -le
-rwxr--r-- + 1 MAINE-UNO\jsmith MAINE-UNO\market 2056 Feb 2 10:18
 adocs.txt
 OWNER: user:MAINE-UNO\jsmith
 GROUP: group:MAINE-UNO\marketing
 0: user:MAINE-UNO\jsmith allow
 file_gen_read,file_gen_write,file_gen_execute,std_write_dac
 1: group:MAINE-UNO\marketing allow file_gen_read
 2: everyone allow file_gen_read
```

In most cases, a result like this preserves the ACL information and minimizes conflicts between actual and expected behavior.

How actual access rights can differ from mode bits

The following examples show how the actual permissions, as set on OneFS, can differ from the mode bits displayed on a Linux client. The examples assume that OneFS is running with its default ACL policies.

On a Linux client connected to an Isilon cluster with NFS, a file with full control for the owner approximates the ACL permissions as follows (the file was originally stored by a Windows system over SMB):

```
[root@rhel5d adir]# ls -l print.css
-rwxr-x--- 1 1000000 1000001 807 Apr 2 2010 print.css
```

By connecting to OneFS with the root account over SSH, you can see the same permissions by running the ls -l command:

```
ls -l print.css
-rwxr-x--- + 1 MAINE-UNO\jsmith MAINE-UNO\market 807 Apr 2 2010
 print.css
```

You can, however, display the actual permissions stored on disk by using the ls -le command. On OneFS, adding the e option to -l prints the ACL from the security descriptor:

```
ls -le print.css
-rwxr-x--- + 1 MAINE-UNO\jsmith MAINE-UNO\market 807 Apr 2 2010
 print.css
 OWNER: user:MAINE-UNO\jsmith
 GROUP: group:MAINE-UNO\marketing
 0: user:MAINE-UNO\jsmith allow file_gen_all
 1: group:MAINE-UNO\marketing allow file_gen_read,file_gen_execute
 2: everyone allow std_read_dac,std_synchronize,file_read_attr
```

Because the file originated from a Windows computer over SMB and because ACL permissions map indirectly to POSIX mode bits, the actual permissions preserve the following Windows access controls for everyone else, even though the mode bits for everyone are unset:

```
allow std_read_dac, std_synchronize, file_read_attr
```

There is another difference that the mode bits cannot capture. In addition to std_write_owner, the owner has std_write_dac—the standard access right to modify the permissions in the object's security descriptor. It is included as part of file_gen_all.

Synthetic ACLs and authoritative mode bits

The following table summarizes how OneFS checks access by default for NFS and SMB clients:

Client accessing cluster	Type of permissions	File access evaluated against
NFS	No ACL	POSIX mode bits, which are authoritative.
SMB	ACL	ACL, which is authoritative.
NFS	ACL	ACL, which is authoritative. But running the ls command from the client shows the approximated POSIX mode bits.
SMB	No ACL	Synthetic ACL. The POSIX mode bits are authoritative.

Here are what the differences can look like in practice when you are logged on to a node in the cluster as root. The first instance shows a file's permissions when the file was stored by an NFS client: The example contains authoritative POSIX mode bits, which is signified by the absence ❶ of a plus sign, followed by a synthetic ACL ❷.

The second instance shows what the permissions look like when the same file was stored by an SMB client: The example contains approximated mode bits, which is signified by the plus sign ❸, followed by an authoritative ACL ❹.

```
wak1-1# ls -led isi_ps.log
-rw-r--r-- ❶   1 root  wheel  2205 Mar 21 14:57 isi_ps.log
OWNER: user:root
GROUP: group:wheel
SYNTHETIC ACL ❷
0: user:root allow file_gen_read,file_gen_write,std_write_dac
1: group:wheel allow file_gen_read
2: everyone allow file_gen_read

wak1-1# ls -led isi_ps.log
-rwxrwxr-- + ❸ 1 root  wheel  2205 Mar 21 14:57 isi_ps.log
OWNER: user:root
GROUP: group:wheel
CONTROL:dacl_auto_inherited ❹
0: group:YORK\domain admins allow file_gen_all
1: everyone allow file_gen_read
2: user:root allow file_gen_read,file_gen_write,std_write_dac
3: group:wheel allow file_gen_read
```

ACL policies for mixed environments

An Isilon cluster includes ACL policies that control how permissions are processed and managed. By default, the cluster is set to merge the new permissions from a chmod command with the file's ACL, as several previous examples have shown. Merging permissions is a powerful method of preserving intended security settings while meeting the expectations of users. In addition, managing ACL policies manually gives you a range of options to respond to the kind of special cases that can surface in mixed environments.

An alternative is to use balanced mode, which is the default global permissions policy. Balanced mode automates file sharing management for a network that mixes UNIX and Windows systems. In contrast to manually configuring ACL policies, balanced mode forces you to use a predetermined set of policies. If any of the balanced-mode policies are unsuitable for your mixed environment, it is recommended that you configure the policies manually.

Managing policies manually lets you set the following options to best serve your environment:

- ACL creation over SMB
- Chmod on files with ACLs
- Inheritance of ACLs created on directories by the chmod command from a Unix client
- Chown and chgrp on files with ACLs
- Access checks with chmod and chown
- Treatment of rwx permissions
- Group owner inheritance
- Chmod with 007 on files with ACLs

- Owner permissions
- Group permissions
- Deny ACEs
- Changing utimes
- Read-only DOS attribute
- The display of mode bits

For a description of the policies and a discussion of their usage, see Dell EMC Isilon Multiprotocol Data Access with a Unified Security Model.

Migrating permissions and mode bits

When you migrate SMB files from another vendor's storage system to an Isilon cluster, you must migrate not only the shares and their file data but also the security metadata, including security identifiers, ACLs, ACEs, and inheritance attributes. For more information, see SMB File Migration to Dell EMC Isilon at Dell EMC.com.

For information about migrating NFS files with POSIX mode bits from another vendor's storage system, see NFS File Migration to a Dell EMC Isilon Cluster at support.emc.com.

Required fields for Windows Services for Unix

If you use Microsoft Active Directory with Windows Services for UNIX and RFC 2307 attributes to manage Linux, UNIX, and Windows systems, the following fields are required in Active Directory:

- uid
- uidNumber
- gidNumber
- loginShell
- UNIXHomeDirectory

Required fields for LDAP

If you are using an LDAP server, such as OpenLDAP, the following fields are required:

- ldap-uid
- ldap-user-filter
- ldap-group-filter
- ldap-loginshell
- ldap-homedirectory

Conclusion

OneFS provides secure access to SMB and NFS clients by connecting to directories services, authenticating users, and controlling access to files. To manage security in a context of multiprotocol data access, OneFS relies on access tokens, file permissions, user identifiers, user mapping rules, ACL policies, and extensions to common Unix commands such as chmod and ls. By manipulating these mechanisms, you can manage the access of client computers and end users to make sure that it is at once seamless and secure.

References

Identities, Access Tokens, and the OneFS User Mapping Service at Dell EMC.com.

Dell EMC Isilon Multiprotocol Data Access with a Unified Security Model at Dell EMC.com.

How to connect Isilon OneFS 7.x to LDAP and Active Directory at support.emc.com, which requires a support account to access the document.

SMB File Migration to Dell EMC Isilon at Dell EMC.com.

NFS File Migration to a Dell EMC Isilon Cluster at support.emc.com, which requires a support account to access the document.