Attribute-based access control

ONTAP 9

NetApp
December 21, 2024

Table of Contents

Attribute-based access control

Attribute-based access control with ONTAP

You can implement enhanced RBAC with attributes and attribute-based access control (ABAC) using ONTAP. ONTAP provides several approaches a customer might use to achieve file-level ABAC, including Labeled NFS 4.2 and XATTRS using NFS and SMB/CIFS.

Attribute-based access control (ABAC) is a sophisticated method for managing access rights that considers user attributes, resource attributes, and environmental conditions. The National Institute of Standards and Technology (NIST) has established a standard for ABAC, providing a framework for its secure and consistent implementation.

Beginning with ONTAP 9.12.1, you can configure ONTAP with NFSv4.2 Security Labels and Extended Attributes (XATTRS) so that it can be integrated with a role-based access control (RBAC) and attribute-based access control (ABAC) identity. This integration allows ONTAP to access control software that is categorized as a NIST ABAC-compliant data management solution, offering a robust and advanced approach to managing access rights in complex environments including Policy Enforcement Point (PEP), a Policy Decision Point (PDP), and policies that consider attributes associated with the user, the resource, and the environment.

The integration of NetApp ONTAP with extended attributes (XATTRS) and Attribute-Based Access Control (ABAC) software is in alignment with the guidelines set forth in NIST Special Publication 800-162, ensuring compliance with the NIST standards for ABAC implementation. The use of NFS 4.2 Security Labels and XATTRS allows for the association of user-defined attributes with files, meeting the NIST ABAC standard's requirement for considering resource attributes in access control decisions. The ABAC software's PEP and PDP align with the NIST ABAC standard's requirement for these components in the access control process. The ability to define complex policies that consider multiple attributes and conditions aligns with the NIST ABAC standard's requirement for policy-based access control.

Related information

- Approaches to ABAC with ONTAP
- NFS in NetApp ONTAP: Best practice and implementation guide
- Request for comments (RFC)
  - RFC 2203: RPCSEC_GSS Protocol Specification
  - RFC 3530: Network File System (NFS) Version 4 Protocol

Approaches to ABAC with ONTAP

ONTAP provides varied approaches a customer might use to achieve file-level ABAC, including Labeled NFSv4.2 and XATTRS using NFS and SMB/CIFS.

Labeled NFSv4.2

Beginning with ONTAP 9.9.1, the NFSv4.2 feature called Labeled NFS is supported.

Labeled NFS is a way to manage granular file and folder access by using SELinux labels and Mandatory Access Control (MAC). These MAC labels are stored with files and folders, and they work in conjunction with UNIX permissions and NFSv4.x ACLs.

Support for Labeled NFS means that ONTAP now recognizes and understands the NFS client's SELinux label settings. Labeled NFS is covered in RFC-7204.

Use Cases for Labeled NFSv4.2 include the following:

- MAC labeling of virtual machine (VM) images
- Data security classification for the public sector (secret, top secret, and other classifications)
- Security compliance
- Diskless Linux

Enable Labeled NFSv4.2

You can enable or disable Labeled NFS with the following advanced privilege option:

```
[-v4.2-seclabel {enabled|disabled}] - NFSV4.2 Security Label Support
(privilege: advanced)
```

This parameter is optional, and the default setting is `disabled`.

Enforcement modes for Labeled NFSv4.2

Beginning in ONTAP 9.9.1, ONTAP supports the following enforcement modes:

- **Limited Server Mode**: ONTAP cannot enforce the labels but can store and transmit them.

  The ability to change MAC labels is also up to the client to enforce.

- **Guest Mode**: If the client is not labeled NFS-aware (v4.1 or lower), MAC labels are not transmitted.

ONTAP does not currently support Full Mode (storing and enforcing MAC labels).

Example configuration of Labeled NFSv4.2

The following example configuration demonstrates concepts using Red Hat Enterprise Linux release 9.3 (Plow).

The user `jrsmith`, created based on John R. Smith's credentials, has the following account privileges:

- Username = `jrsmith`
- Privileges = `uid=1112(jrsmith) gid=1112(jrsmith) groups=1112(jrsmith) context=user_u:user_r:user_t:s0`

There are two roles: the admin account that is a privileged user and user `jrsmith` as described in the following MLS privileges table:

Users	Role	Type	Levels
`admins`	`sysadm_r`	`sysadm_t`	`t:s0`
`jrsmith`	`user_r`	`user_t`	`t:s1 - t:s4`

In this example environment, user `jrsmith` has access to files at the levels `s0` to `s3`. We can enhance the existing security classifications, as outlined below, to ensure that administrators do not have access to user-specific data.

- s0 = privilege admin user data
- s0 = unclassified data
- s1 = confidential
- s2 = secret data
- s3 = top secret data

Follow your security policies of your organization

Example of NFSv4.2 security label with MCS

In addition to Multi-Level Security (MLS), another capability called Multi-Category Security (MCS) allows you to define categories such as projects.

NFS security label	Value
`entitySecurityMark`	`t:s01 = UNCLASSIFIED`

Extended Attributes (XATTRS)

Beginning in ONTAP 9.12.1, ONTAP supports xattrs. xattrs allow metadata to be associated with files and directories beyond what is provided by the system, such as access control lists (ACLs) or user-defined attributes.

To implement xattrs, you can use `setfattr` and `getfattr` command-line utilities in Linux for managing xattrs of file system objects. These tools provide a powerful way to manage additional metadata for files and directories. They should be used with care though as improper use can lead to unexpected behavior or security issues. Always refer to the `setfattr` and `getfattr` man pages or other reliable documentation for detailed usage instructions.

When xattrs is enabled on an ONTAP filesystem, users can set, modify, and retrieve arbitrary attributes on files. These attributes can be used to store additional information about the file that is not captured by the standard set of file attributes, such as access control information.

Requirements for using xattrs in ONTAP

- Red Hat Enterprise Linux 8.4 or later
- Ubuntu 22.04 or later
- Each file can have up to 128 xattrs
- xattr keys are limited to 255 bytes
- The combined key or value size is 1,729 bytes per xattr
- Directories and files can have xattrs
- To set and retrieve xattrs, `w` or write mode bits must be enabled for the user and group

Use cases for xattrs

xattrs are utilized within the user namespace and do not carry any intrinsic significance to ONTAP itself. Instead, their practical applications are determined and managed exclusively by the client-side application that interacts with the file system.

xattr use case examples:

- Recording the name of the application responsible for creating a file.
- Maintaining a reference to the email message from which a file was obtained.
- Establishing a categorization framework for organizing file objects.
- Labeling files with the URL of their original download source.

Commands for managing xattrs

- `setfattr`: Sets an extended attribute of a file or directory:

  ```
  setfattr -n <attribute_name> -v <attribute_value> <file or directory name>
  ```

  Sample command:

  ```
  setfattr -n user.comment -v test example.txt
  ```

- `getfattr`: Retrieves the value of a specific extended attribute or lists all extended attributes of a file or directory:

  Specific attribute: `getfattr -n <attribute_name> <file or directory name>`

  All attributes: `getfattr <file or directory name>`

  Sample command:

  ```
  getfattr -n user.comment example.txt
  ```

Table 1. xattr key value pair examples

xattr	Value
`user.digitalIdentifier`	`CN=John Smith jrsmith, OU=Finance, OU=U.S.ACME, O=US, C=US`
`user.countryOfAffiliations`	`USA`

User Permissions with ACE for Extended Attributes

An Access Control Entry (ACE) is a component within an Access Control List (ACL) that defines the access rights or permissions granted to an individual user or a group of users for a specific resource, such as a file or directory. Each ACE specifies the type of access allowed or denied and is associated with a particular security principal (user or group identity).

Table 2. Access Control Entry (ACE) required for xattrs

File type	Retrieve xattr	Set xattrs
File	R	a,w,T
Directory	R	T

Explanation of the permissions required for xattrs:

Retrieve xattr: The permissions required for a user to read the extended attributes of a file or directory. The "R" signifies that read permission is necessary. **Set xattrs**: The permissions needed to modify or set the extended attributes. "a," "w," and "T" represent different examples of permissions, such append, write, and a specific permission related to xattrs. **Files**: Users need append, write, and potentially a special permission related to xattrs to set extended attributes. **Directories**: A specific permission "T" is required to set extended attributes.

SMB/CIFS protocol support for xattrs

ONTAP's support for the SMB/CIFS protocol extends to comprehensive handling of xattrs, which are an integral part of file metadata in Windows environments. Extended attributes allow users and applications to store additional information beyond the standard set of file attributes, such as author details, custom security descriptors, or application-specific data. ONTAP's SMB/CIFS implementation ensures that these xattrs are fully supported, allowing for seamless integration with Windows services and applications that depend on this metadata for functionality and policy enforcement.

When files are accessed or transferred over SMB/CIFS shares managed by ONTAP, the system preserves the integrity of xattrs, ensuring that all metadata is retained and remains consistent. This is particularly important for maintaining security settings and for applications that rely on xattrs for configuration or operation. ONTAP's robust handling of xattrs within the SMB/CIFS context ensures that file sharing across different platforms and environments is reliable and secure, providing users with a seamless experience and administrators with the assurance that data governance policies are upheld. Whether it's for collaboration, data archiving, or compliance, ONTAP's attention to xattrs within SMB/CIFS shares represents its commitment to data management excellence and interoperability in mixed-OS environments.

Policy enforcement point (PEP) and Policy Decision Point (PDP) in ABAC

In an attribute-based access control (ABAC) system, the Policy Enforcement Point (PEP) and Policy Decision Point (PDP) play crucial roles. The PEP is responsible for enforcing access control policies, while the PDP makes the decision on whether to grant or deny access based on the policies.

In the context of the Python code snippet provided, the script itself acts as a PEP. It enforces the access control decision by either granting access to the file by opening it and reading its contents or denying access by raising a `PermissionError`.

The PDP, on the other hand, would be part of the underlying SELinux system. When the script tries to open the file with a specific SELinux context, the SELinux system checks its policies to decide whether to grant or deny access. This decision is then enforced by the script.

Below is a step-by-step example breakdown of how this code works in an ABAC environment:

1. The script sets the SELinux context to `jrsmith` context using the `selinux.setcon()` function. This is equivalent to `jrsmith` trying to access the file.
2. The script tries to open the file. This is where the PEP comes into play.
3. The SELinux system checks its policies to see if `jrsmith` (or more specifically, a user with `jrsmith`

SELinux context) is allowed to access the file. This is the PDP's role.

4. If `jrsmith` is allowed to access the file, the SELinux system lets the script open the file, and the script reads and prints the file's contents.
5. If `jrsmith` is not allowed to access the file, the SELinux system prevents the script from opening the file, and the script raises a `PermissionError`.
6. The script restores the original SELinux context to ensure that the temporary context change does not affect other operations.

Using python, the code to get the context is shown below where the variable file path is the document that is to be checked:

```
#Get the current context

context = selinux.getfilecon(file_path)[1]
```

ONTAP cloning and SnapMirror

ONTAP's cloning and SnapMirror technologies are designed to provide efficient and reliable data replication and cloning capabilities, ensuring that all aspects of file data, including extended attributes (xattrs), are preserved and transferred along with the file. xattrs are critical as they store additional metadata associated with a file, such as security labels, access control information, and user-defined data, which are essential for maintaining the file's context and integrity.

When a volume is cloned using ONTAP's FlexClone technology, an exact writable replica of the volume is created. This cloning process is instantaneous and space-efficient, and it includes all file data and metadata, ensuring that xattrs are fully replicated. Similarly, SnapMirror ensures that data is mirrored to a secondary system with full fidelity. This includes xattrs, which are crucial for applications that rely on this metadata to function correctly.

By including xattrs in both cloning and replication operations, NetApp ONTAP ensures that the complete dataset, with all its characteristics, is available and consistent across primary and secondary storage systems. This comprehensive approach to data management is vital for organizations that require consistent data protection, quick recovery, and adherence to compliance and regulatory standards. It also simplifies the management of data across different environments, whether on-premises or in the cloud, providing users with the confidence that their data is complete and unaltered during these processes.

NFSv4.2 Security Labels have the caveats defined in Labeled NFSv4.2.

Examples of controlling access to data

The following example entry for data stored in John R Smith's PKI cert shows how NetApp's approach can be applied to a file and provide fine-grained access control.

These examples are for illustrative purposes, and it is the government's responsibility to define what metadata is NFSv4.2 security label and xattrs. Details on updating and label retention are omitted for simplicity.

Table 3. Example PKI cert values

Key	Value
entitySecurityMark	t:s01 = UNCLASSIFIED

Info	{
	"commonName": {
	"value": "Smith John R jrsmith"
	},
	"emailAddresses": [
	{
	"value": "jrsmith@dod.mil"
	}
	],
	"employeeId": {
	"value": "00000387835"
	},
	"firstName": {
	"value": "John"
	},
	"lastName": {
	"value": "Smith"
	},
	"telephoneNumber": {
	"value": "938/260-9537"
	},
	"uid": {
	"value": "jrsmith"
	}
	}

specification	"DoD"
uuid	b4111349-7875-4115-ad30-0928565f2e15

adminOrganization	{
	"value": "DoD"
	}

Key	Value
briefings	`[ { "value": "ABC1000" }, { "value": "DEF1001" }, { "value": "EFG2000" } ]`
citizenshipStatus	`{ "value": "US" }`
clearances	`[ { "value": "TS" }, { "value": "S" }, { "value": "C" }, { "value": "U" } ]`
countryOfAffiliations	`[ { "value": "USA" } ]`

Key	Value

digitalIdentifier	<pre>{
	  "classification": "UNCLASSIFIED",
	  "value": "cn=smith john r jrsmith, ou=dod, o=u.s.
	government, c=us"
	}</pre>

dissemTos	<pre>{
	  "value": "DoD"
	}</pre>

dutyOrganization	<pre>{
	  "value": "DoD"
	}</pre>

entityType	<pre>{
	  "value": "GOV"
	}</pre>

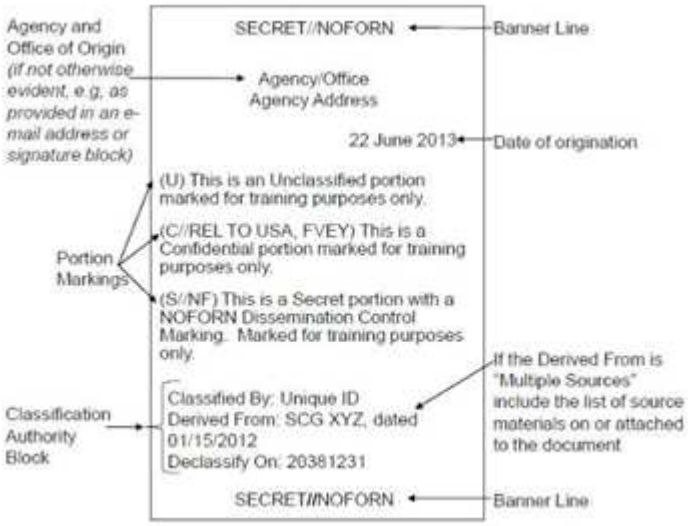
fineAccessControls	<pre>[
	  {
	  "value": "SI"
	  },
	  {
	  "value": "TK"
	  },
	  {
	  "value": "NSYS"
	  }
	]</pre>

These PKI entitlements show John R. Smith's access details, including access by data type and attribution.

If John R. Smith created and saved a document called *"sample_analysis.doc"*, according to the relevant policy guidance issuances the user would add the appropriate banner and portion markings, agency and office of origin, and appropriate classification authority block based on the classification of the document as shown in the following image. This rich metadata is only understandable after it has been scanned by Natural Language Processing (NLP) and had rules applied to make meaning from the markings. Tools such as NetApp BlueXP

Classification can do that but are less efficient for access control decisions because they require permission to look inside the document.

Unclassified CAPCO document portion marking



In scenarios where IC-TDF metadata is stored separately from the file, NetApp advocates for an additional layer of fine-grained access control. This involves storing access control information at both the directory level and in association with each file. As an example, consider the following tags linked to a file:

- NFSv4.2 Security Labels: Utilized for making security decisions
- xattrs: Provide supplementary information pertinent to the file and the organizational program requirements

The following key-value pairs are examples of metadata that could be stored as xattrs and offer detailed information about the file's creator and associated security classifications. This metadata can be leveraged by client applications to make informed access decisions and to organize files according to organizational standards and requirements.

Table 4. Example of xattr key-value pairs

Key	Value
`user.uuid`	`"761d2e3c-e778-4ee4-997b-3bb9a6a1d3fa"`
`user.entitySecurityMark`	`"UNCLASSIFIED"`
`user.specification`	`"INFO"`

Key	Value

user.Info	{
	"commonName": {
	"value": "Smith John R jrsmith"
	},
	"currentOrganization": {
	"value": "TUV33"
	},
	"displayName": {
	"value": "John Smith"
	},
	"emailAddresses": [
	"jrsmith@example.org"
	],
	"employeeId": {
	"value": "00000405732"
	},
	"firstName": {
	"value": "John"
	},
	"lastName": {
	"value": "Smith"
	},
	"managers": [
	{
	"value": ""
	}
	],
	"organizations": [
	{
	"value": "TUV33"
	},
	{
	"value": "WXY44"
	}
	],
	"personalTitle": {
	"value": ""
	},
	"secureTelephoneNumber": {
	"value": "506-7718"
	},
	"telephoneNumber": {
	"value": "264/160-7187"
	},
	"title": {
	"value": "Software Engineer"
	},

Key	Value
user.geo_point	[-78.7941, 35.7956]

```
    }
}
```

Auditing changes to labels

Auditing changes to xattrs or NFS security labels is a critical aspect of file system management and security. Standard file system auditing tools enable the monitoring and logging of all changes to a file system, including modifications to extended attributes and security labels.

In Linux environments, the `auditd` daemon is commonly used to establish auditing for file system events. It allows administrators to configure rules to watch for specific system calls related to xattr changes, such as `setxattr`, `lsetxattr`, and `fsetxattr` for setting attributes and `removexattr`, `lremovexattr`, and `fremovexattr` for removing attributes.

ONTAP FPolicy extends these capabilities by providing a robust framework for real-time monitoring and control of file operations. FPolicy can be configured to support various xattr events, offering granular control over file operations and the ability to enforce comprehensive data management policies.

For users utilizing xattrs, especially in NFSv3 and NFSv4 environments, only certain combinations of file operations and filters are supported for monitoring. The list of supported file operation and filter combinations for FPolicy monitoring of NFSv3 and NFSv4 file access events is detailed below:

Supported file operations	Supported filters
setattr	offline-bit, setattr_with_owner_change, setattr_with_group_change, setattr_with_mode_change, setattr_with_modify_time_change, setattr_with_access_time_change, setattr_with_size_change, exclude_directory

Example of an auditd log snippet for a setattr operation:

```
type=SYSCALL msg=audit(1713451401.168:106964): arch=c000003e syscall=188
success=yes exit=0 a0=7fac252f0590 a1=7fac251d4750 a2=7fac252e50a0 a3=25
items=1 ppid=247417 pid=247563 auid=1112 uid=1112 gid=1112 euid=1112
suid=1112 fsuid=1112 egid=1112 sgid=1112 fsgid=1112 tty=pts0 ses=141
comm="python3" exe="/usr/bin/python3.9"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key="*set-xattr*"ARCH=x86_64 SYSCALL=**setxattr** AUID="jrsmith"
UID="jrsmith" GID="jrsmith" EUID="jrsmith" SUID="jrsmith"
FSUID="jrsmith" EGID="jrsmith" SGID="jrsmith" FSGID="jrsmith"
```

Enabling ONTAP FPolicy for users working with xattrs provides a layer of visibility and control that is essential for maintaining the integrity and security of the file system. By leveraging FPolicy's advanced monitoring capabilities, organizations can ensure that all changes to xattrs are tracked, audited, and aligned with their security and compliance standards. This proactive approach to file system management is why enabling ONTAP FPolicy is highly recommended for any organization looking to enhance its data governance and

protection strategies.

Integration with ABAC identity and access control software

To fully harness the capabilities of attribute-based access control (ABAC), ONTAP can integrate with an ABAC-oriented identity and access management software.

> In parallel to this content, NetApp has a reference implementation using GreyBox. One assumption for this content is that the government's identity, authentication, and access services include at minimum a Policy Enforcement Point (PEP) and a Policy Decision Point (PDP) that act as intermediaries for access to the file system.

In a practical setting, an organization would employ a blend of NFS security labels and xattrs. These are used to represent a variety of metadata, including classification, security, application, and content, which are all instrumental in making ABAC decisions. XATTR, for instance, can be used to store the resource attributes that the PDP uses for its decision-making process. An attribute could be defined to represent the classification level of a file (for example, "Unclassified", "Confidential", "Secret", or "Top Secret"). The PDP could then utilize this attribute to enforce a policy that restricts users to access only files that have a classification level equal to or lower than their clearance level.

Example process flow for ABAC

1. User presents credentials (for example, PKI, Oauth, SAML) to system access to PEP and gets results from PDP.

   The PEP's role is to intercept the user's access request and forward it to the PDP.

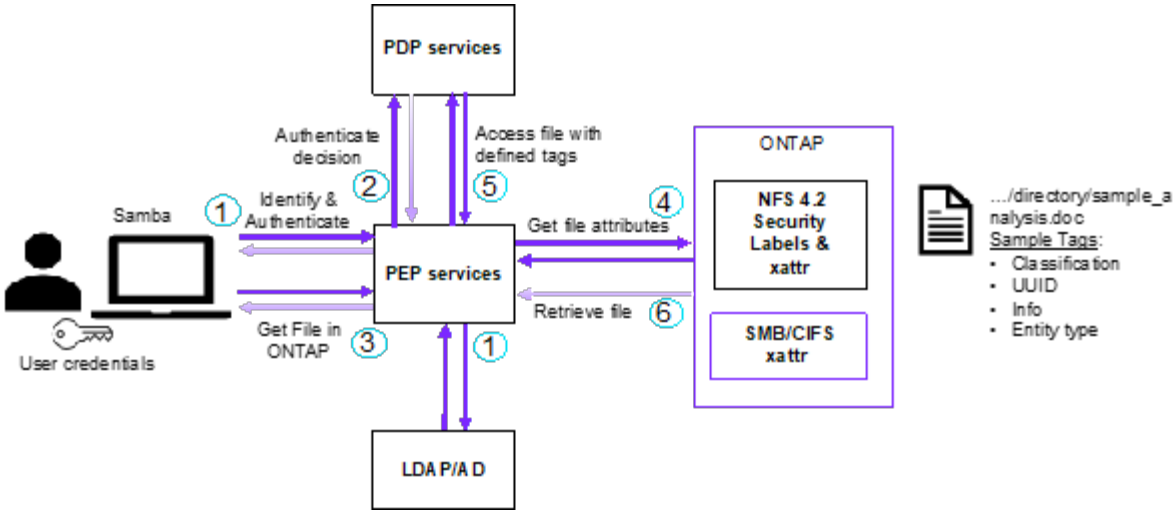
2. The PDP then evaluates this request against the established ABAC policies.

   These policies consider various attributes related to the user, the resource in question, and the surrounding environment. Based on these policies, the PDP makes an access decision to either allow or deny and then communicates this decision back to the PEP.

   PDP provides policy to PEP to enforce. The PEP then enforces this decision, either granting or denying the user's access request as per the PDP's decision.

3. After a successful request, the user requests a file stored in ONTAP (AFF, AFF-C, for example).
4. If the request is successful, PEP gets fine-grain access control tags from document.
5. PEP requests policy for user based on that user's certs.
6. PEP makes a decision based on policy and tags if the user has access to the file and lets the user retrieve the file.

> The actual access might be done using tokens not proxied through.



Related information

- NFS in NetApp ONTAP: Best practice and implementation guide
- Request for comments (RFC)
  - RFC 2203: RPCSEC_GSS Protocol Specification
  - RFC 3530: Network File System (NFS) Version 4 Protocol

Copyright information

Trademark information