|  |  |
| --- | --- |
|  | Managing Access Sentinel policies using LDAP and PowerShell |

## Introduction

Access Sentinel stores XACML policy information in several different directory entries in the ViewDS directory, and you can access and manage the data using the LDAP protocol. This document explains the different entries and attributes and provides some examples.

NOTE: The Access Sentinel management UI (VMA) contains logic to maintain certain kinds of logical consistency in the policy store, for instance ensuring that version numbers are sequential. The PowerShell commands do not at this time enforce these constraints, so take care and keep snapshots of your environment

NOTE: The Access Sentinel policy store can store more complex policies than the VMA can produce. For instance, the VMA only generates policies with a single XACML Rule. You can create policies with multiple rules in a single policy entry, and the PDP can handle it just fine. However, the VMA can only deal with policy entries containing a single rule and is unable to display or manage policies with multiple rules.

## Policy Elements

All of the directory entries used to maintain XACML policy information are subentries of the XACML access control domain. This means that they don't appear in normal LDAP searches unless the search explicitly specifies objectClass=subentry in the search filter.

The components of XACML policies stored in the directory include:

* XACML attribute definitions
* Named expressions (only used by the Windows admin interface – the VMA)
* ABAC policies
* Role enablement policies
* Role definitions

Some of the attributes contain complex data structures and are presented through the LDAP interface as structured strings using the SDUA syntax.

For example:

{ displayName "Username", category "urn:oasis:names:tc:xacml:1.0:subject-category:access-subject", attribute identifier:"urn:oasis:names:tc:xacml:1.0:subject:subject-id", dataType "http://www.w3.org/2001/XMLSchema#string",type viewDSUserName }

## PowerShell Commands

I've provided a set of PowerShell commands that provide a limited ability to manage policies in Access Sentinel.

### Installation

The PSAccessSentinel module contains all of the PowerShell commands to manage AccessSentinel policies. It has a dependency on the PSLDAP PowerShell module (currently only provided as binaries), which also needs to me installed.

After you pull the files from Github, add the top-level ~\PSAccessSentinel directory and the ~\PSAccessSentine\PSLDAP to your PSModulePath environment variable, and restart your PowerShell session. The run the following PowerShell command to load the modules:

PS> Import-Module PSAccessSentinel -verbose

PowerShell will display a list of commands that will now be available.

### Connection Commands

The PSAccessSentinel commands all use the concept of an Access Sentinel connection. It is the first parameter for each command. A connection includes the hostname, LDAP port, username and password, as well as the distinguished name (DN) of the access control domain the connection is to be used with.

A connection is just another PowerShell object and you can save and restore a connection object from file and store them in PowerShell variables. The Access Sentinel commands all support a "default" connection that is stored in a session-scope variable. If you don't provide a connection for a command, it will try to use the default connection.

|  |  |
| --- | --- |
| New-ASConnection | Returns a new Access Sentinel connection object |
| Write-ASConnection | Saves an Access Sentinel connection object to file |
| Read-ASConnection | Reads a previously saved Access Sentinel connection object from file |
| Set-DefaultASConnection | Sets the session-scope default connection, either from a variable, from the command parameters, or from file |
| Get-DefaultASConnection | Retrieves the current default connection object |

### Attribute Commands

The accessControlSubentry named "cn=XACML Attribute Mappings" subordinate to the access control domain entry contains the XACML attribute mappings for the XACML access control domain in the viewDSXACMLAttributePresentation attribute. The attribute is a multivalued string, and each value represents a single attribute mapping.

Each mapping associates a user-friendly attribute name with an XACML triple including the category URI, attribute URI, and data type URI. It also optionally associates the attribute with an LDAP attribute type (either by name or OID).

Each value is formatted as a SDUA string and looks similar to this:

{ displayName "Username", category "urn:oasis:names:tc:xacml:1.0:subject-category:access-subject", attribute identifier:"urn:oasis:names:tc:xacml:1.0:subject:subject-id", dataType "http://www.w3.org/2001/XMLSchema#string", type viewDSUserName }

There are commands to retrieve, add, remove, import, and export attribute definitions.

|  |  |
| --- | --- |
| Get-ASAttributeDefinitions | Retrieves the entire set of attribute definitions |
| Add-ASAttributeDefinition | Adds a new attribute definition |
| Export-ASAttributeDefinitions | Exports all of the attribute definitions to a file |
| Import-ASAttributeDefinitions | Imports all of the attribute definitions from a previous export |
| Remove-ASAttributeDefinition | Removes a single attribute definition |

### Policy Version Commands

Access Sentinel maintains multiple versions of policies that can be edited and evaluated. At any point in time, a single policy version can be *active*, indicating that it is the version to use when an application requests an authorization decision. All other policy versions are *inactive*.

A policy version can be based on another policy version, effectively inheriting its rules. If a policy version is used as the base of another version, it must be *locked*, which makes the policy version unchangeable. A policy that is not locked is consider *open*.

The versions form a tree. A policy is in every version in the subtree rooted at the version specified in the policy's version field excluding any contained subtrees rooted at a version specified in the viewDSXACMLVersionExclusion attribute of the subentry holding the policy.

Access Sentinel maintains policy version information in a subentry with a randomly generated UUID as the common name, for instance cn=041dc5cd-bb65-43fb-8b17-86a72e548604, o=Deltawing. It will be the only subentry containing viewDSXACMLActivePolicy and viewDSXACMLPolicyVersion attributes, so you can search for it with the LDAP filter (viewDSXACMLActivePolicy=\*).

The viewDSXACMLActivePolicy attribute contains the identifier of the currently active policy, and the viewDSXACMLPolicyVersion attribute contains the set of all policies and their current state (locked/unlocked) and the parent version, if any.

#### viewDSXACMLActivePolicy Attribute

The viewDSXACMLActivePolicy operational attribute is only permitted in the access control specific administrative point. It identifies the version of policy that should be used for each issuer. Only one version may be specified for each issuer. A value without an issuer identifies the version for the system policy. The attribute appears as an SDUA string, for example:

{version "1.0"}

#### viewDSXACMLPolicyVersion Attribute

A viewDSXACMLPolicyVersion attribute value describes the state of all versions of an access control policy for an access control administrative area. When a new version of a policy is created, a value of this attribute should be added to the access control administrative point identifying the new version and an optional version match string identifying the version the new policy is based on. Once the policy is completed, the locked field should be set to TRUE and not further editing of this version of the policy should be permitted. For example:

{ identifier "1.0", locked TRUE }

{ identifier "2.0", base "1.0" }

The current iteration of the Access Sentinel PowerShell commands only supports inspecting versions; you need to use the VMA to modify existing versions or create new versions. There are two commands for inspecting version information:

|  |  |
| --- | --- |
| Get-ASVersions | Retrieve the current set of versions |
| Get-ASActiveVersion | Retrieve the id of the currently active version |

### ABAC Policy Commands

ViewDS maintains policy rules using two types of directory entries. The viewDSXACMLSubtreePolicySubentry object class holds a single XACML policy with a single rule that applies to the subtree of its immediately superior entry. The VMA displays each policy as an ABAC Rule on the Policy Versions tab, and it has a Scope value of "subtree".

The viewDSXACMLEntryPolicySubentry object class holds a single XACML policy with a single rule that applies only to the its immediately superior entry. The VMA displays these rules with a scope of "entry". Beyond the name of the attribute that contains the actual XACML rule, the two object classes are essentially identical.

|  |  |  |
| --- | --- | --- |
| Attribute name | Data type | Description |
| viewDSXACMLSubtreePolicy/ viewDSXACMLEntryPolicy | String | Holds an uncomposed XACML policy that applies to the subtree/entry of the administrative point superior of the subentry. The policy is assumed to have an implicit target that restricts its applicability to that subtree/entry, including any subentries. |
| viewDSXACMLRoleCondition | String | *Only used for role enablement policies. TBD* |
| viewDSXACMLVersionExclusion | String | Contains a set of policy version identifiers within which this policy entry doesn't apply. Normally a policy will apply when its policy version is active, or the active policy version is derived from the policy's version. |
| viewDSXACMLPrecedence | Integer | Contains an integer representing the precedence of the policy. |

#### viewDSXACMLSubtreePolicy/viewDSXACMLEntryPolicy Attribute

The viewDSXACMLSubtreePolicy attribute contains an XACML policy element with certain characteristics and constraints. It corresponds to a single rule as presented in the VMA.

* The XML namespace abbreviation is "n0".
* The XML namespace URI is "urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
* The subtree policy and entry policy directory entries are subentries of the XACML access control area in the directory
* The PolicyId attribute of the policy must be the same as the common name of the entry containing the policy and in the form "urn:uuid:cf4c7fd6-2c85-44fd-8f80-b9752b3a43bc"
* The Description element of the Policy contains the name of the policy as shown in the "Label" column in the VMA.
* The combining algorithm must be "urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides"[[1]](#footnote-1)
* The Target element of the policy must be empty.
* The Policy contains a single Rule element[[2]](#footnote-2).
* Each Rule element has a unique RuleId attribute that is a UUID in the form "d3107fd6-2c85-44fd-8f80-b9752b3a771a"
* The Description element of the Rule contains the description of the rule as shown in the "Description" column of the VMA.

#### viewDSXACMLSubtreePolicy/viewDSXACMLEntryPolicy Boilerplate

* {0} is replaced by the PolicyId (UUID) of the policy
* {1} is replaced by the name of the policy
* {2} is replaced by the RuleId (UUID) of the rule
* {3} is replaced by "Permit" or "Deny" depending on the effect of the rule
* {3} is replaced by the name of the policy
* {4} is replaced by the XACML Condition element that makes up the rule

<n0:Policy xmlns:n0="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicyId="urn:uuid:{0}" Version="1.0" RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides">

<n0:Description>{1}</n0:Description>

<n0:PolicyDefaults>

<n0:XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</n0:XPathVersion>

</n0:PolicyDefaults>

<n0:Target/>

<n0:Rule RuleId="{2}" Effect="{3}">

<n0:Description>{4}</n0:Description>

<n0:Condition>

{5}

</n0:Condition>

</n0:Rule>

</n0:Policy>

There are three PowerShell commands for managing ABAC policies:

|  |  |
| --- | --- |
| Add-ASABACPolicy | Adds a new ABAC policy with the specified Policy element, or with a Condition element and the other parameters specified with the command |
| Get-ASABACPolicy | Retrieves all the ABAC policies |
| Remove-ASABACPolicy | Removes the specified ABAC policy |

### Examples

# Set the default Access Sentinel connection

PS> Set-DefaultASConnection -Hostname '192.168.0.150' -Port 3006 -UserDN 'cn=Margaret Hunter,O=deltawing' -Password testpass -DomainDN 'o=deltawing'

# Retrieve the default Access Sentinel connection

PS> Get-DefaultASConnection

Port : 3006

UserDN : cn=Margaret Hunter,O=deltawing

DomainDN : o=deltawing

Hostname : 192.168.0.150

Password : testpass

LDAPCon : System.DirectoryServices.Protocols.LdapConnection

# Save the current default connection in a file

PS> Write-ASConnection -Connection (Get-DefaultASConnection) -FilePath 'pdp.con'

# Restore the default Access Sentinel connection

PS> Set-DefaultASConnection -FilePath '.\pdp.con'

# List the XACML attributes

PS> Get-ASAttributeDefinitions

idOrSel : identifier

dataType : http://www.w3.org/2001/XMLSchema#string

category : urn:oasis:names:tc:xacml:1.0:subject-category:access-subject

ldapType : viewDSUserName

attribute : urn:oasis:names:tc:xacml:1.0:subject:subject-id

displayName : Username

# Add an XACML attribute

PS> Add-ASAttributeDefinition -DisplayName 'Phone' -Category 'urn:oasis:names:tc:xacml:1.0:subject-category:access-subject' -attribute 'http://viewds.com/xacml/attributes/telephone' -Datatype 'http://www.w3.org/2001/XMLSchema#string'

# List the XACML attributes

PS> Get-ASAttributeDefinitions

idOrSel : identifier

dataType : http://www.w3.org/2001/XMLSchema#string

category : urn:oasis:names:tc:xacml:1.0:subject-category:access-subject

ldapType : viewDSUserName

attribute : urn:oasis:names:tc:xacml:1.0:subject:subject-id

displayName : Username

attribute : http://viewds.com/xacml/attributes/telephone

category : urn:oasis:names:tc:xacml:1.0:subject-category:access-subject

idOrSel : identifier

dataType : http://www.w3.org/2001/XMLSchema#string

displayName : Phone

### Policy Testing Commands

There are two commands you can use to test policy evaluation with different sets of attributes.

|  |  |
| --- | --- |
| New-XACMLRequestAttr | Creates a new attribute to use in an XACML authorization request |
| Send-XACMLRequest | Sends an authorization request to the designated PDP endpoint and returns the response. |

# Send a XACML authorization request

PS>

PS> Send-XACMLRequest -Uri 'http://192.168.0.16:3009' `

>> -SubjectAttrs @(New-XACMLRequestAttr 'urn:oasis:names:tc:xacml:1.0:subject:subject-id' 'icampbell') `

>> -ResourceAttrs @(New-XACMLRequestAttr 'urn:oasis:names:tc:xacml:1.0:resource:resource-id' 'cn=The second application,cn=Applications,o=Deltawing' x500Name) `

>> -Verbose

Response

--------

@{Decision=Permit; Status=}

1. You can use any of the standard combining algorithms for XACML, but the VMA will not manage them properly. [↑](#footnote-ref-1)
2. You can specify more than one rule using any combining algorithm, but the VMA will not be able to manage the policy properly. [↑](#footnote-ref-2)