SAML Authentication Against Multiple AD Forests

with Veeam Backup Enterprise Manager

Matthias Mehrtens

2024

Table of Contents

1 Abstract 5

1.1 Intended Audience 5

2 SAML Overview 6

2.1 SAML Process and Communication Flow 6

2.2 Identity Brokering 7

2.3 Certificates 8

3 Veeam Enterprise Manager Use Case 10

3.1 Initial Situation and Requirements 10

4 Preparation of VBEM and AD Forests 12

4.1 Roles, Groups and Users 12

4.1.1 Role Setup in Veeam Backup Enterprise Manager 12

4.1.2 Users and Groups in the Management Forest 14

4.1.3 Users and Groups in the Tenant Forests 15

5 Preparation and Deployment of Certificates 17

5.1 Create and Export Required Certificates 17

5.1.1 Export CA Certificate 19

5.1.2 Export Individual Server Certificates 20

5.2 Deploy CA Certificate to DCs and VBEM 21

5.3 Deploy Individual Certificates to DCs and VBEM 22

5.4 Enable LDAPS on AD Domain Controllers 23

5.5 Select Webserver Certificate on VBEM Server 24

6 Setting up Keycloak 26

6.1 Creating a Custom Realm 29

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

7 Building the Relationship between Keycloak and Veeam Backup Enterpise Manager 32
7.1 Download SAML Metadata Files from Veeam Backup Enterprise Manager 32
7.2 Import SAML Metadata into Keycloak 34
7.3 Download SAML IdP Metadata from Keycloak 35
7.4 Import SAML IdP Metadata into Veeam Backup Enterprise Manager 36
7.5 Add Mapper for the "NameID" Attribute 37
7.6 Test the first SAML Authentication 40
7.6.1 Add UPN Attribute to Test User Account 43
8 Keycloak Identity Broker Configuration 45
8.1 Prepare Keycloak for Groups: Create "Group List" Mapper 45
8.2 Preparing Active Directory Forests 47
8.3 Create User Federation for each Active Directory Forest 48
8.3.1 Add Mappers for Group Attribute 50
8.3.2 Add Mappers for UPN Attribute 51
9 Additional Configuration Options 52
9.1 Securing Mappers via LDAP Filters 52
9.2 Enforcing Secure LDAPS (Keycloak → AD) 53
9.2.1 Test LDAPS Connection 57
10 Conclusion 58
11 Addendum 59
11.1 Troubleshooting 59
11.1.1 General Advice Regarding Keycloak Configuration Changes 59
11.1.2 Keycloak container stops immediately after startup 59
11.1.3 Unsigned LDAP Connections are not blocked 59
11.2 References 60

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

12 About Veeam Software 61
12.1 About the Author 61

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

1 Abstract

Veeam introduced the ability to connect Veeam Backup Enterprise Manager (VBEM) to external authentication sources via the Security Assertion Markup Language (SAML) protocol some time ago. However, I have often heard from customers who struggle to fully utilize this functionality, as the additional configuration of all software products involved can be challenging when trying to build a single sign-on (SSO) solution for a single VBEM instance with multiple authentication sources via SAML.

This document describes an example of how to set up such an environment, in particular by using the open-source product Keycloak as a single "Identity Broker" (IB) between multiple, untrusted AD forests and VBEM.

In this example, I also describe how the users' group memberships in the respective AD sources are mapped to "External Groups" in VBEM by using attribute mapping and SAML assertions/claims. This is critical for efficient operations as it reduces the administrative overhead in VBEM to just one group per role and authentication source - there is **no** need to set up or maintain additional accounts in VBEM for each individual user authenticating through this solution - we are building a true single sign-on (SSO) solution based on unique identities!

1.1 Intended Audience

Readers should be familiar with both Microsoft Active Directory and Veeam Backup Enterprise Manager as part of a Veeam Backup & Replication infrastructure. A general understanding of a public key infrastructure (PKI) is also helpful, i.e. you should at least know the basic concepts of certificates, a certification authority, signing, public and private keys, etc. On the other hand, I do not expect the reader to be familiar with SAML or Keycloak, as the basics of these are explained in this document - at least to the extent required by the use case scenario it describes.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

2 SAML Overview

Let's start with a brief overview about SAML and its concepts.

Security Assertion Markup Language (SAML) is an XML-based framework that enables secure single sign-on (SSO) authentication and authorization across different applications and systems. It allows users to authenticate once with an identity provider (IdP) and gain access to multiple service providers (SPs) without the need for separate login credentials.

2.1 SAML Process and Communication Flow

The SAML communication flow involves three main actors: the user, the IdP and the SP. The user initiates the authentication process by accessing a service provided by the SP, which is presented as a web front-end via https. The SP requires authentication from the user and redirects the user's browser to the IdP's web-based authentication form so that the IdP can ask the user for login credentials. The user therefore authenticates with the IdP and not with the SP, as only the IdP has the relevant user account information. Once authentication is complete, the IdP creates a SAML assertion containing the user's identity information, signs the assertion and sends it back to the user's web browser. Upon receipt, the user's web browser forwards this SAML assertion back to the SP, which validates the IdP's signature and grants access based on the content of the assertion.

The SAML communication flow does not require direct communication between the SP and the IdP. This makes the process very flexible in the security context of unconnected services (e.g. SP is on site while the IdP is in the public cloud).

Fig. 1 illustrates the process and communication flow.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

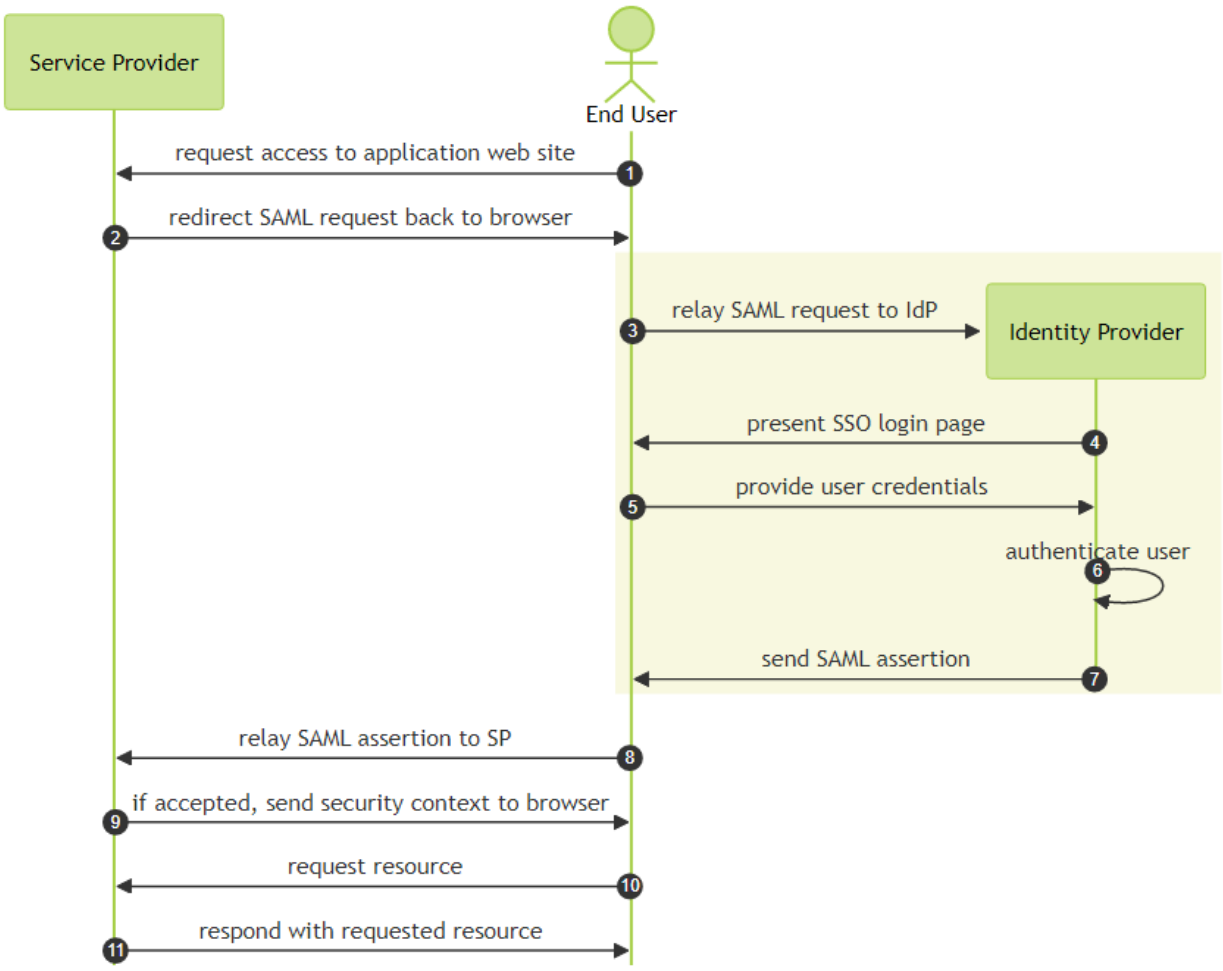


Figure 1: SAML Communication Flow Diagram

2.2 Identity Brokering

To enable SSO with multiple identity providers, the additional role of an "Identity Broker" (IB) can be used.

In the context of the SP, the IB role transparently replaces the IdP (i.e. in the configuration of the SP, the IB is configured as the IdP), so that the SP usually does not know that the authentication is being performed outside the IB. The IB then decides which IdP is to be used for authentication (either rule-based or by presenting the requesting user with a selection dialog). It forwards the request to the chosen IdP and creates the SAML assertion based on the response before sending it back to the user's browser.

Fig. 2 illustrates the extended communication flow including the Identity Broker role (added elements highlighted in the red box).

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

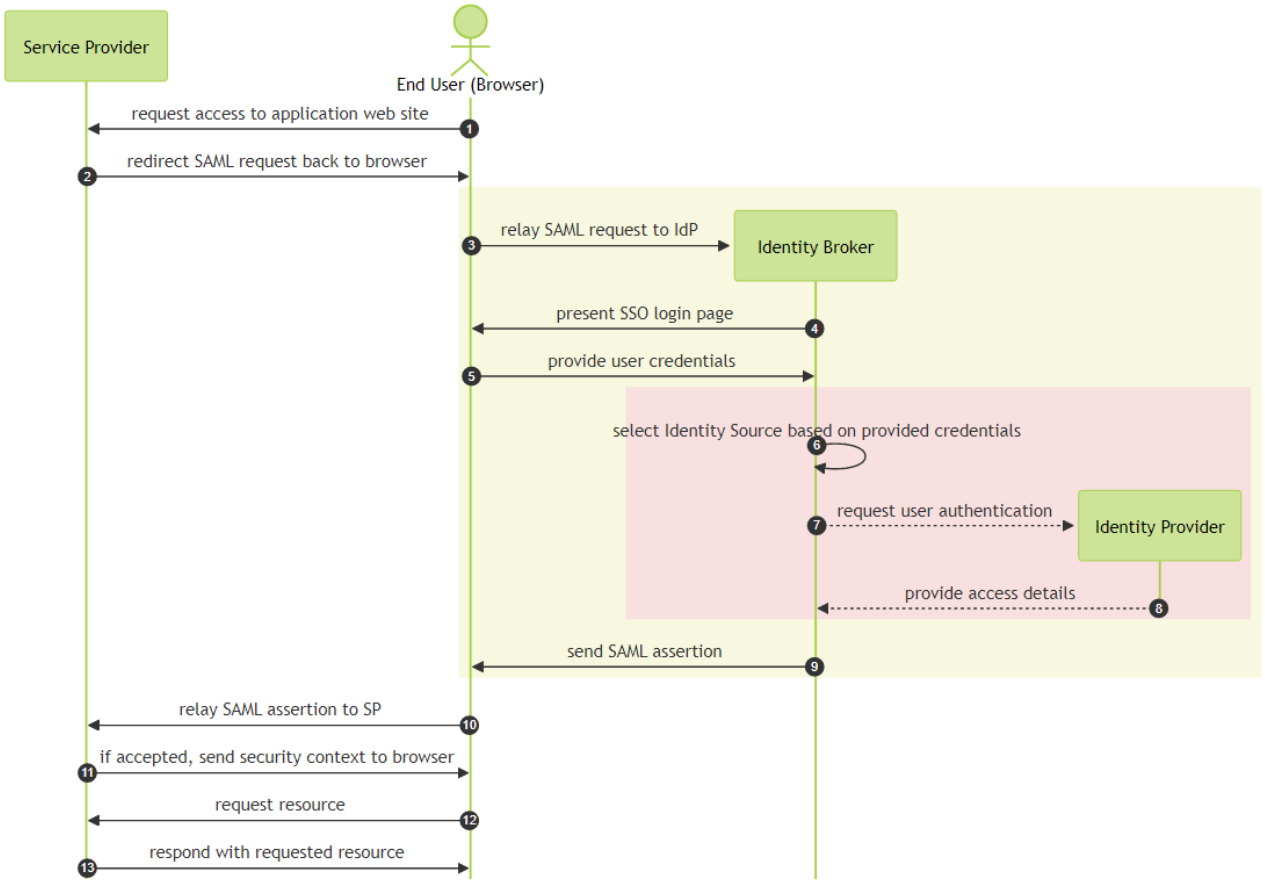


Figure 2: Extended SAML Communication Flow Diagram (including Identity Broker)

2.3 Certificates

To facilitate secure communication, protocols within the SAML framework are commonly based on https communication between the involved parties, secured by Transport Layer Security (TLS). The most important SAML protocols are:

- **SAML Authentication Request Protocol**: This protocol allows the SP to initiate the authentication process by redirecting a user's authentication request to the IdP. The request contains information such as the requested authentication method and the "assertion consumer service" (ACS) URL of the SP.
- **SAML Assertion Protocol**: Once the user is authenticated, the IdP generates a SAML assertion containing a pre-selected set of the user's identity attributes (e.g. username, group memberships, email address, etc.). The assertion is encrypted and digitally signed by the IdP to ensure its integrity and authenticity.

Because of the TLS encryption requirements of these protocols, certificates play a crucial role in SAML-based authentication to ensure secure communication and validate the integrity of the exchanged data.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

The IdP and SP each need a digital certificate that includes a public key for encryption and a private key for digital signing. The SP verifies the signature on the SAML assertion using the IdP's public key, while the IdP encrypts sensitive information using the SP's public key.

I've already mentioned in sec. 2.1 that there is **no requirement** for direct communication between the SP and the IdP/IB. However, it is required that the signatures created by the IdP/IB and SP are mutually trusted, i.e. the SP trusts the digital signatures created by the IdP/IB and vice versa. This is usually achieved by using certificates on both sides that have been created (i.e. requested from and signed by) a certification authority (CA) that is trusted by both sides. The simplest approach would be to use a public, well-known CA to create these certificates and ensure that the IdP/IB and SP have this CA in their list of "trusted root certificate authorities". But self-signed certificates can also be used as long as they are mutually trusted.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

3 Veeam Enterprise Manager Use Case

For the example to be described here, I have chosen a use case that customers have asked me for help with several times. I have tried to make the example applicable to both corporate environments and service provider scenarios.

3.1 Initial Situation and Requirements

Let's assume we have an in-house IT environment with a Veeam Backup & Replication (VBR) infrastructure protecting virtual machines running in VMware vSphere. The VBR setup includes a Veeam Backup Enterprise Manager (VBEM) instance running within an internal Active Directory (AD) forest, which we call the "Application Forest". For administrative and security reasons, we also operate an additional AD forest, the "Management Forest". For this example, it is irrelevant whether these two forests have a trust relationship or not (normally there is one, but it is not required for this example).

In addition, there are several external AD forests that run outside the security zone of our data centre or in separate locations over which our internal IT operations have no control. These forests have no trust relationships with each other or with any of our internal AD forests, so these could e.g. be AD forests run by several tenants (assuming the in-house IT environment belongs to a service provider). That's why we're calling them "tenant forests" from now on. For corporate environments running multiple forests, there are usually trust relationships between some or even all of these forests or domains. But as such a "trusted" environment would be capable of using regular AD authentication mechanisms across forest or domain boundaries without involving any SAML components or processes, let's just assume that no trust relationships exist at all.

Nevertheless, it is required that the users of the management forest as well as of each tenant forest have access to the central VBEM instance and that the roles assigned to each of these users are based on specific user group memberships within their respective source ADs.

Some additional requirements are:

- An internally managed Keycloak instance shall be used as SAML Identity Broker (IB) between all AD forests (except the application forest) and Veeam Backup Enterprise Manager.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

- Keycloak will be allowed to initiate outbound ldaps (636/tcp, TLS-secured) connections to a selected Domain Controller server of each AD forest containing VBEM users.
- Incomig traffic from external users to the VBEM web interface will be allowed via TLS-secured https protocol (443/tcp) only.
- Incoming traffic from external systems to the Keycloak instance will also be authorised for the TLS-secured https protocol (443/tcp).

Fig. 3 illustrates the described scenario, using the SAML components we discussed in the previous section:

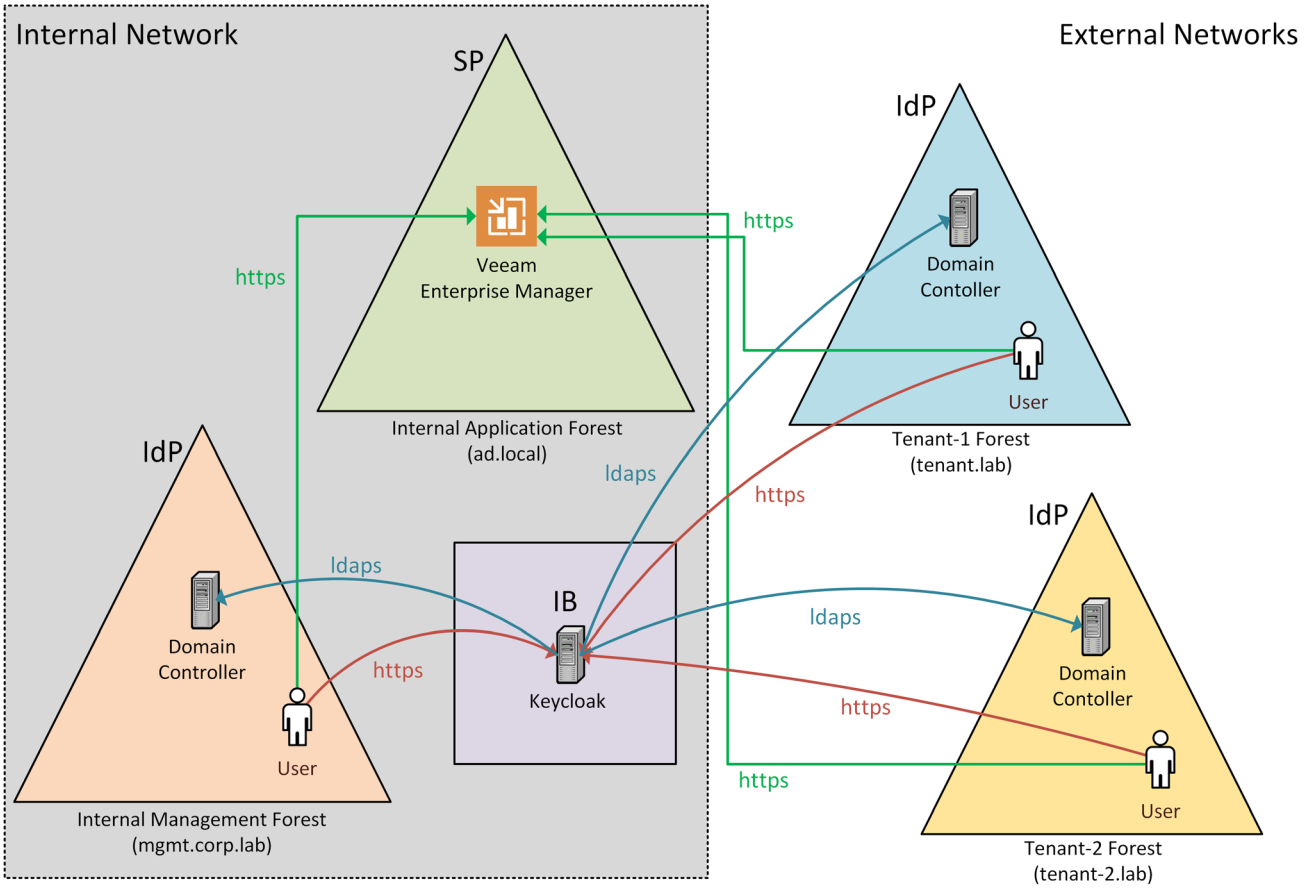


Figure 3: Scenario Components and Connections

In this example, the Keycloak instance is located within the boundaries of the "internal network" according to fig. 3. In principle, however, it can be placed anywhere as long as the connectivity requirements are met. You could therefore use Keycloak in a completely different security zone, for example by deploying it in a public cloud such as Microsoft Azure, Amazon Web Services (AWS) or Google Cloud Platform (GCP). This makes the whole approach very flexible in terms of your specific security requirements.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

4 Preparation of VBEM and AD Forests

4.1 Roles, Groups and Users

One of the main goals of this example is to assign VBEM roles to users based on their group memberships within their source Active Directories. To do this, we first need to set up the required roles, groups and users.

4.1.1 Role Setup in Veeam Backup Enterprise Manager

There are two different configuration areas in VBEM that enable us to use and configure user roles which belong to two independent services:

1. **Accounts and Roles** (for users accessing the regular VBEM website): VBEM offers 3 different roles a user can be assigned to (details can be found in the Online User Guide):
   - Portal Administrator
   - Restore Operator
   - Portal User
2. **vSphere Self-Service Backup Portal** (for tenants): Users can be allowed to create and manage backup jobs that process vSphere VMs and restore data from backups created with these jobs, while being restricted to a subset of VMs based on the configured "Delegation Mode" and the assigned "Tenant Account" (details in the User Guide).

For our example, we will use a VBEM Role for users located in the "Internal Management Forest". For users located in the external "Tenant Forests", we will use Self-Service Tenant Groups instead.

To create a "Portal Administrator" role assignement in VBEM using the account type "External Group", go to "Configuration/Roles" in the UI and click on "Add…".

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

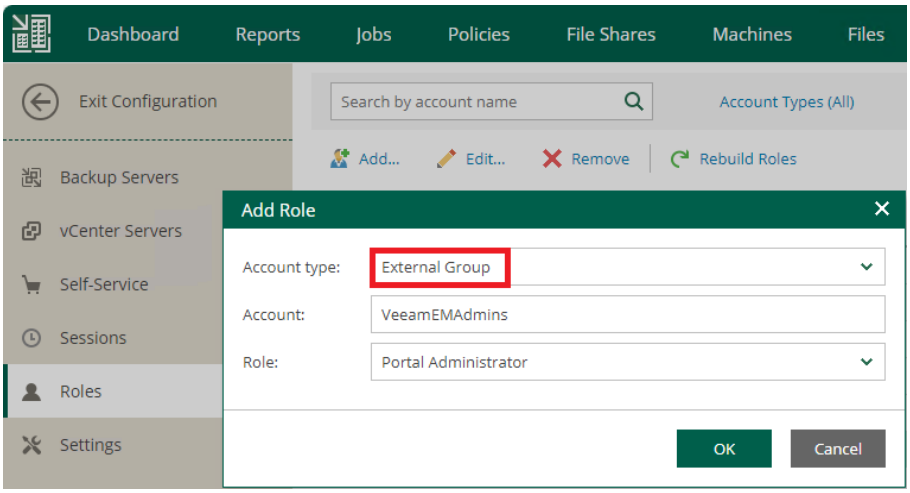


Figure 4: Assign VBEM Portal Administrator Role to External Group named `VeeamEMAdmins`

The external account name `VeeamEMAdmins` entered in this dialog will later be matched to an AD group in the `mgmt.corp.lab` AD in fig. 3.

For the tenant users located in the Tenant Forests of fig. 3, I chose the delegation mode "vSphere Tags" as shown in fig. 5:

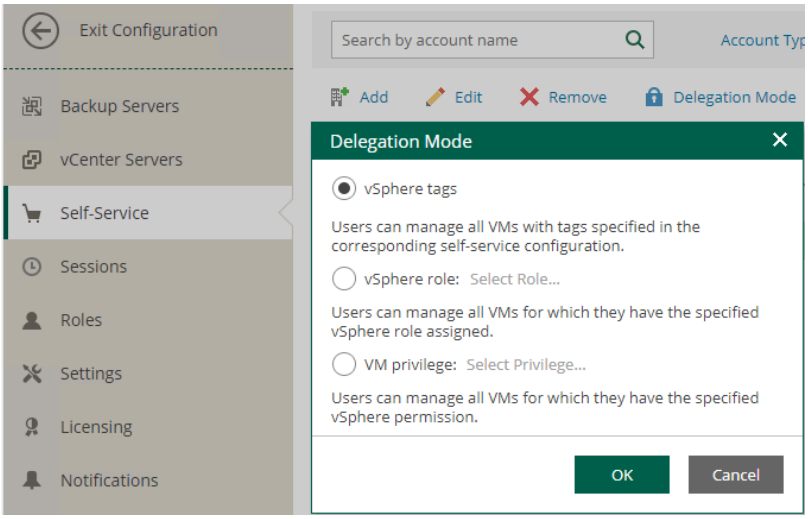


Figure 5: Self-Service Delegation Mode configuration

For the corresponding settings in vSphere, I created a tag category named "Tenant" and added tags for each of the tenants (named "Tenant-1" and "Tenant-2") according to fig. 3. I've then tagged a few VMs with "Tenant-1" and some others with "Tenant-2" to simulate the corresponding ownership of these VMs. Fig. 6 shows an example of the tags applied to one of my vSphere VMs:

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

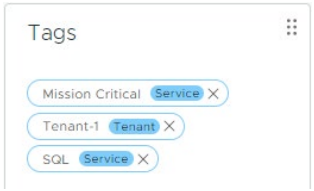


Figure 6: vSphere tags assigned to a VM

Now we can create Self-Service Tenant accounts in VBEM that will be restricted to the tagged VMs by clicking on "Add..." on the "Configuration/Self-Service" page:

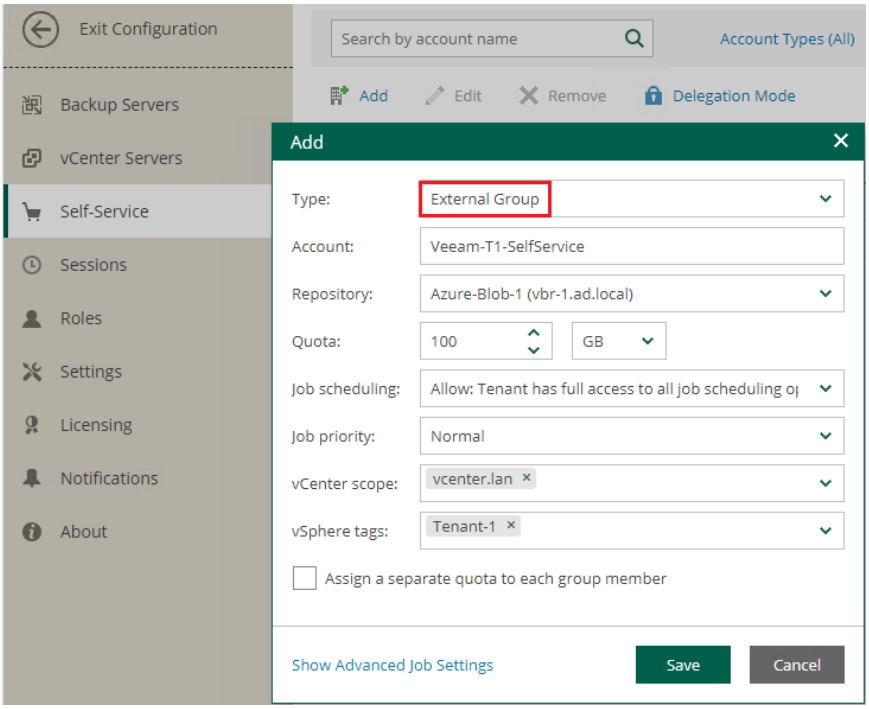


Figure 7: Adding External Group to VBEM Self-Service named `Veeam-T1-SelfService`

Again, we have to select "External Group" as the type in the dialog shown in fig. 7. It's obvious from the name `Veeam-T1-SelfService` that this account will later refer to users of our first tenant AD `tenant.lab`. I have also created a second Self-Service account named `Veeam-T2-SelfService` which I'm going to use for users of the second tenant AD `tenant-2.lab` shown in fig. 3.

4.1.2 Users and Groups in the Management Forest

Now we have to set up corresponding user groups in the ADs in order to be able to assign the group members to the VBEM roles. Let's start with our internal management forest from fig. 3:

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

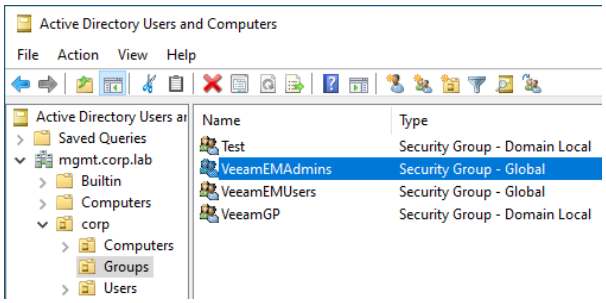


Figure 8: Active Directory User Group in the Management Forest

Please make sure that the group name `VeeamEMAdmins` in fig. 8 is the same as we chose for the "External Group" name when we created the VBEM Portal Administrator role shown in fig. 4.

It is also important to note that I have created a custom organizational unit (OU) for this group in fig. 8. As we will see later in this document (sec. 8.3.1 and sec. 9.1), Keycloak can be restricted to "see" only a subset of all existing AD groups by searching for them only in a specific OU. To do this, we need to take a note of the `distinguishedName` attribute of the OU. The same information will later be needed for the OU containing our users:

- Management users: `OU=Users,OU=corp,DC=mgmt,DC=corp,DC=lab`
- Management groups: `OU=Groups,OU=corp,DC=mgmt,DC=corp,DC=lab`

To finalise the work in the AD management forest, I also created two AD user accounts (`emuser1` and `emuser2`) and added them to the newly created group:

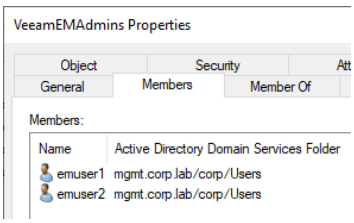


Figure 9: AD Users in the `VeeamEMAdmins` Group

4.1.3 Users and Groups in the Tenant Forests

The group setup in the two tenant AD forests of this example (fig. 3) has to be done in the same way, just making sure we use the same group names we already defined in VBEM:

- `Veeam-T1-SelfService` in `tenant.lab`

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

- `Veeam-T2-SelfService` in `tenant-2.lab`

The following screenshots show the created groups.

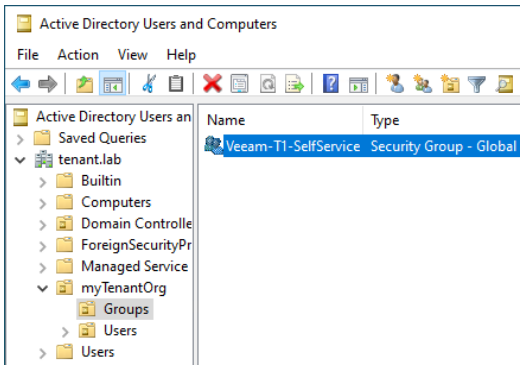


Figure 10: Adding User Group in Tenant AD Forest `tenant.lab`

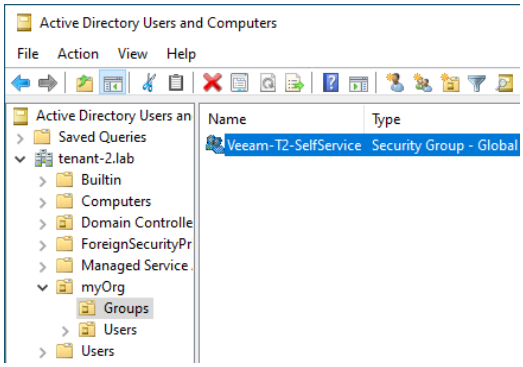


Figure 11: Adding User Group in Tenant AD Forest `tenant-2.lab`

Again, we have to make a note of the `distinguishedName` attributes of each OU containing the new groups as well as for each users OU:

- Tenant 1 users: `OU=Users,OU=myTenantOrg,DC=tenant,DC=lab`
- Tenant 1 groups: `OU=Groups,OU=myTenantOrg,DC=tenant,DC=lab`
- Tenant 2 users: `OU=Users,OU=myOrg,DC=tenant-2,DC=lab`
- Tenant 2 groups: `OU=Groups,OU=myOrg,DC=tenant-2,DC=lab`

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

5 Preparation and Deployment of Certificates

5.1 Create and Export Required Certificates

As all communications have to be secured via https or ldaps, we need to create certificates for all involved components. As mentioned in sec. 2.3, self-signed certificates would be sufficient, but for this example I decided to use a certification authority. I'm not going to describe all the details, as this would disctract us too far from our main topic, and there are a lot of resources on the web to research how this can be done. Here are some of the key steps:

- For the certification authority (CA), I've used the open-source software XCA. It can be installed on a Windows server, an MSI-package for installation is available on Github.
- Within XCA, I first created a CA certificate using a new key and the "`[default] CA`" template:

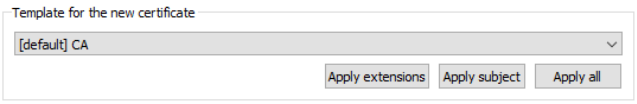


Figure 12: New Certificate "CA" Template

- The next step is to create server certificates for the following servers:
  - One for at least one Domain Controller (DC) per AD forest. In our example (fig. 3), there are 3 relevant forests: Two tenant forests and one management forest. We don't need a certificate for the application forest for this example.
  - One for the Veeam Backup Enterprise Manager (VBEM) server
  - One for the Keycloak server (which we will create later)

It's important that all server certificates have the following properties:

- Each certificate should have its own unique private key. In XCA, you can pre-create private keys before creating certificates to use them. If you forgot to pre-create a key, make sure that you click on *"Generate a new key"* at the bottom of the *"Subject"* page of the *"New Certificate"* wizard:

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

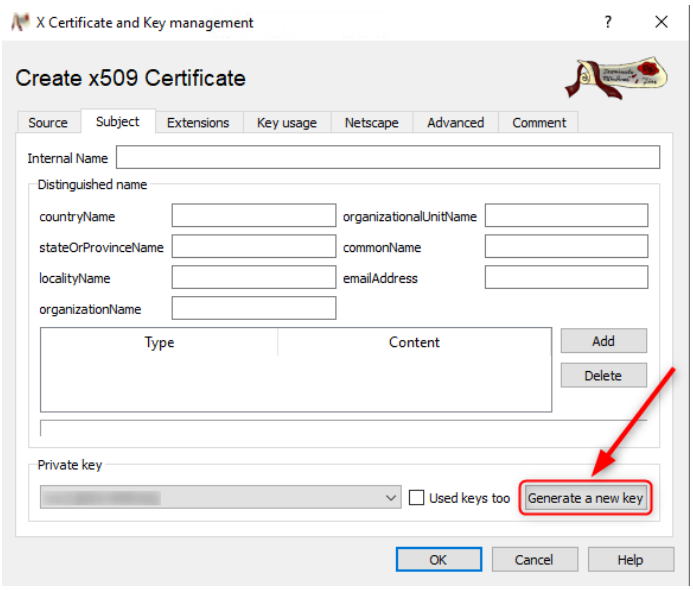


Figure 13: New Certificate, Generate New Key

- Enhanced Key usage properties must include *"Server Authentication"* (`1.3.6.1.5.5.7.3.1`). In XCA, just use the existing "`[default] TLS_Server`" template in the *"New Certificate"* dialog and click *"Apply all"*:

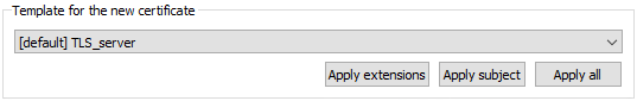


Figure 14: New Certificate "`TLS_Server`" Template

- *"Subject Alternate Name"* (`SAN`) property includes all DNS names of the server it will be installed on. Make sure to include the server's DNS FQDN(s). It's also recommended to include the server's single label DNS name (i.e. its hostname) as a SAN entry if you haven't used this name for the *"Common Name (CN)"* attribute already.

- Certificates must be signed by the previously created CA. Within XCA, the easiest way to do this is to enable option *"Use this Certificate for signing"* and select the CA certificate in the *"New Certificate"* dialog:

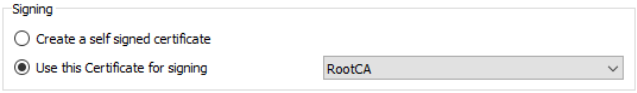


Figure 15: New Certificate needs to be signed by CA

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

5.1.1 Export CA Certificate

We need to export the certificate of the CA we've created because it needs to be distributed to all servers. In XCA, we can export the certificate via the context menu:

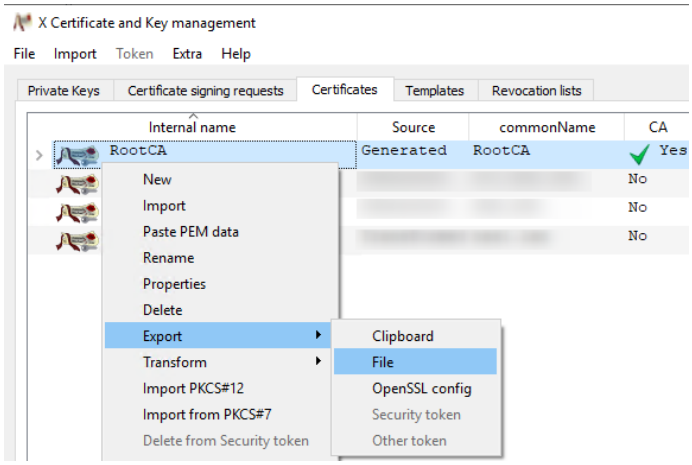


Figure 16: Export CA Certificate

For the export format we select "`PEM (*.crt)`" as this format can be imported on Windows as well as Linux computers. Please make sure that you do **not** select a format that contains the private key, as the private key of the CA certificate must **never** be passed on!

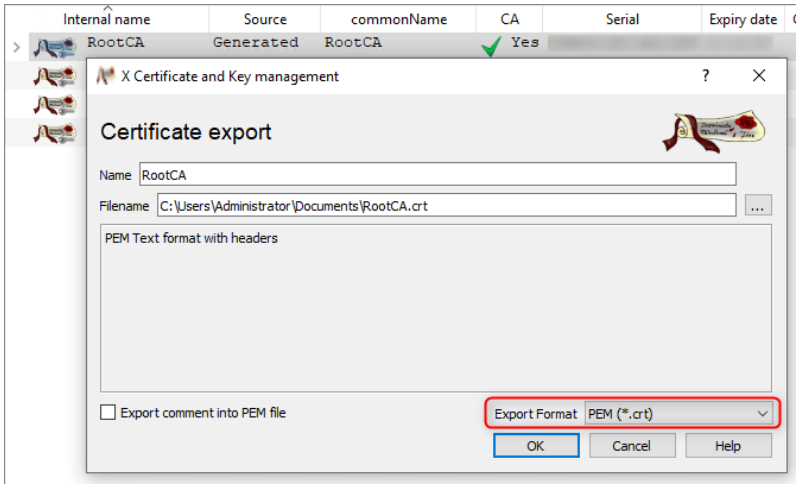


Figure 17: Certificate Export Format

We now need to make this file available on our "target" servers where we want to import it, e.g. by copying it to their local disks or to a network share they have access to.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

5.1.2 Export Individual Server Certificates

This process is almost the same as the CA certificate export described in sec. 5.1.1, but this time we need to make sure that each exported certificate file also contains the private key for the server we're going to deploy it to. For each of the Windows server certificates (for the Domain Controllers as well as for the VBEM server), I chose the following export format:

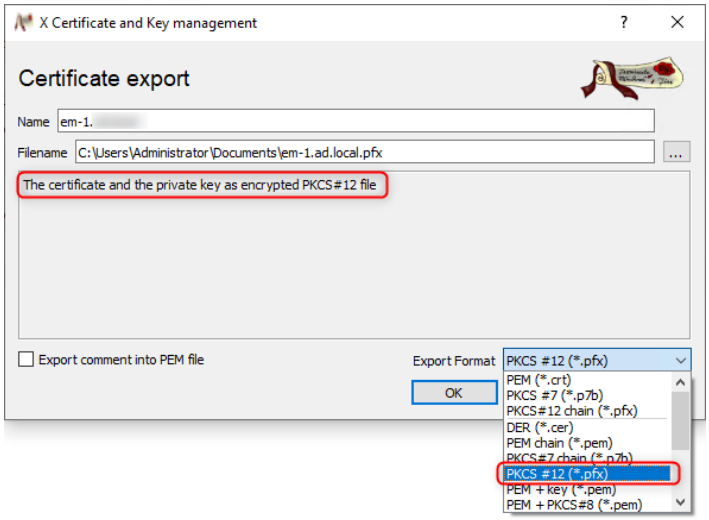


Figure 18: Export Certificate for Windows Server in PFX format

As the dialog says, the *"Personal Information Exchange (PFX)"* format allows storing the private key into the same file as the certificate itself. The next step of the export wizard presents a password dialog to enable encryption of the file because it will contain sensitive information.

For Linux servers (our Keycloak server we will setup later), I chose a different format requiring two files. The first one includes the certificate and the private key:

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

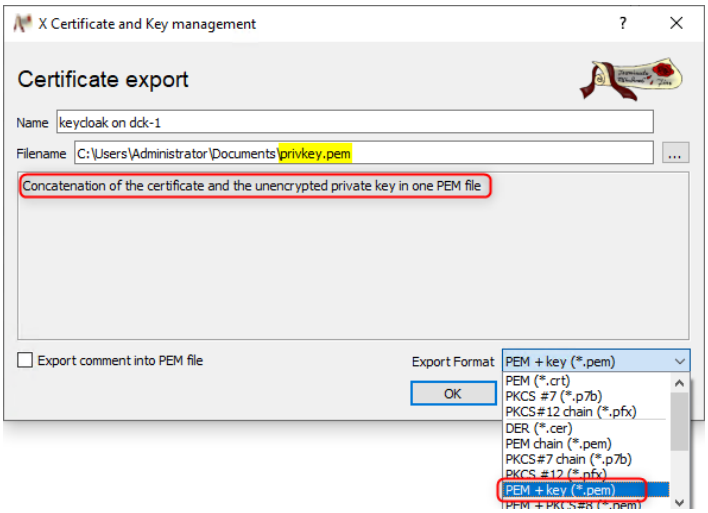


Figure 19: Export Private Key for Linux Server in PEM format (`privkey.pem`)

Additionally, we have to export a file containing the full certificate chain (i.e., including the CA certificate):

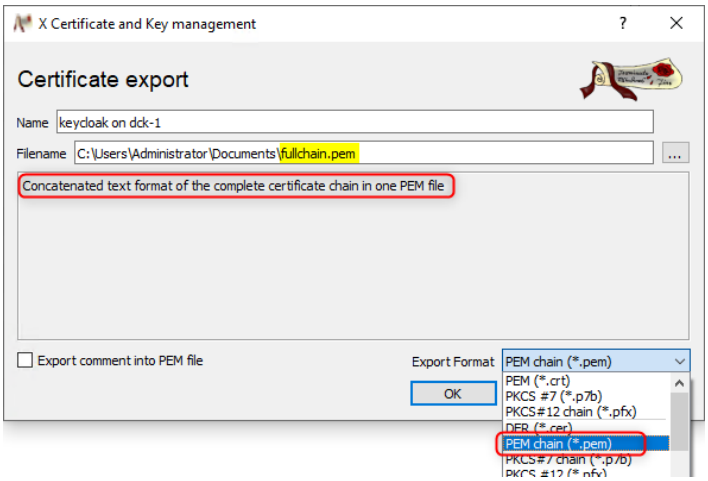


Figure 20: Export Certificate Chain for Linux Server in PEM format (`fullchain.pem`)

5.2 Deploy CA Certificate to DCs and VBEM

On each DC, we can import the CA certificate exported in sec. 5.1.1 by simply double-clicking the exported file. This will show the certificate's contents (to check if you clicked on the correct file!) and allow us to click on the *"Install Certificate..."* button to start the import process.

It is important to select *"Local Machine"* as the *"Store Location"*:

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

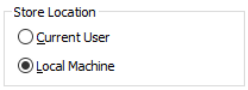


Figure 21: Import Certificate Store Location

In the next step, we have to select "`Trusted Root Certification Authorities`" as the target certificate store:

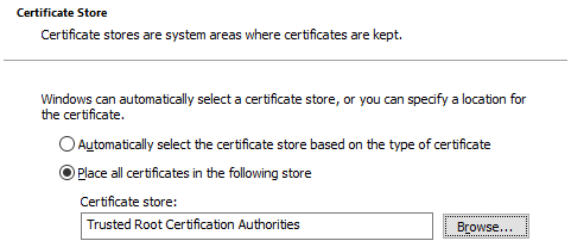


Figure 22: Import Certificate to Trusted Root Store

This ensures that the server will now trust all certificates that have been signed by our CA certificate.

The same process needs to be done on both the VBEM server and on your client computer (assuming it is running Windows) that you use to access VBEM and Keycloak through your browser. You may complain that this would be a lot of work in a large corporate environment where users on many client devices need this kind of access, and you are right. For this reason, in enterprise environments, you should use certificates issued by a "well-known" certification authority (as mentioned in sec. 2.3). This essentially means that all computers already trust these certificates, so you don't have to set anything up on the client computers. The disadvantage of using these certificate authorities is that obtaining certificates from them is usually not free (an exception would be Let's Encrypt).

Another option for corporate environments would be to use the certification authority capabilities of Microsoft Windows Server and use Active Directory to automatically distribute the CA certificates to all clients connected to the domain. However, this was too time-consuming for a test laboratory, so I chose the manual method described here.

5.3 Deploy Individual Certificates to DCs and VBEM

This is almost the same process as the deployment of the CA certificate described in sec. 5.2. But we now have to import into the local computer's "`Personal`" certificate store. After double-clicking the server's

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

individual certificate file (exported in sec. 5.1.2 in PFX format), we need to use the following settings in the import wizard:

- For the certificate store location, we again have to select the *"Local Machine"* option from fig. 21.
- You need to enter the password that has been configured during export:

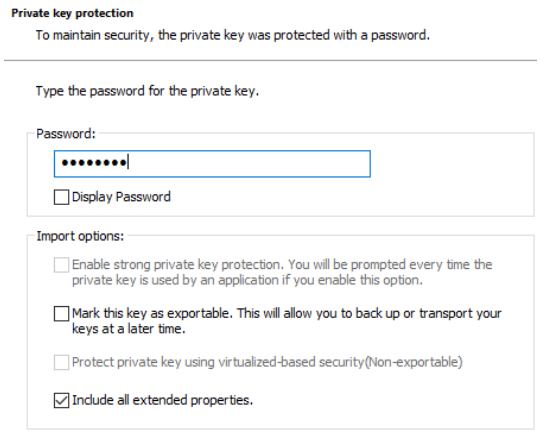


Figure 23: Import Certificate: Exportable Private Key

- Final step: Make sure you select the "`Personal`" certificate store by clicking on the *"Browse..."* button:

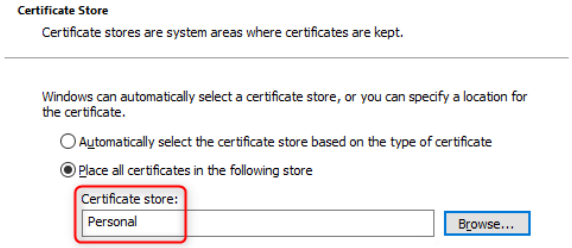


Figure 24: Import Certificate to `Personal` Certificate Store

5.4 Enable LDAPS on AD Domain Controllers

If the certificate import was successful on the Domain Controllers, everything should be set to enable secure ldaps on port 636/tcp. The server will select the certificate located in the "`Personal`" store automatically. To test if ldaps is working correctly, have a look at the section *"Verify an LDAPS connection"* in this Microsoft article. There is also an excellent blog article on "xdot509.blog" about Domain Controller certificates for ldaps that has many more details.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

Important Note: The unsigned/insecure ldap access is not automatically deactivated as soon as ldaps is activated. To enforce signed ldaps on Domain Controllers, you would have to follow this Microsoft article.

5.5 Select Webserver Certificate on VBEM Server

Veeam Backup Enterprise Manager uses Microsoft Internet Information Server (IIS) to provide its web-based GUI. This is why we have to reconfigure IIS in order to set the TLS certificate that IIS should use to present TLS secured webpages. IIS can use the local computers certificate stores, and we already imported the certificate into this store, so we have to start the IIS Manager for this last step of the configuration.

- In IIS Manager, select the *"VeeamBackup"* site in the tree view, right-click it and select *"Edit Bindings..."*.

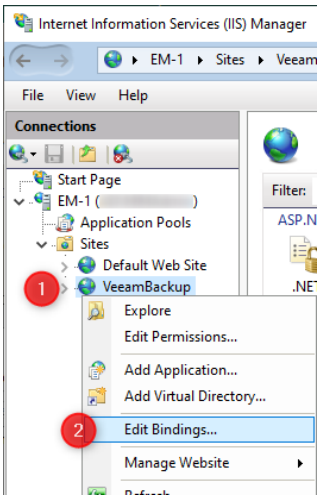


Figure 25: IIS Manager Site Context Menu

- This will open a dialog showing http and https IP address and port bindings, the default on VBEM should look like this:

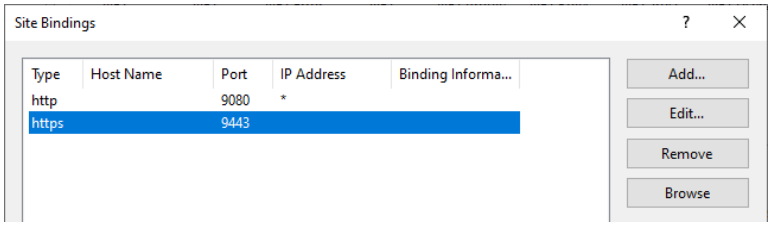


Figure 26: IIS Manager Site Bindings

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

- Selecting the line with type https and clicking on *"Edit…"* will get you to another dialog that includes the *"SSL certificate"* selection for this site:

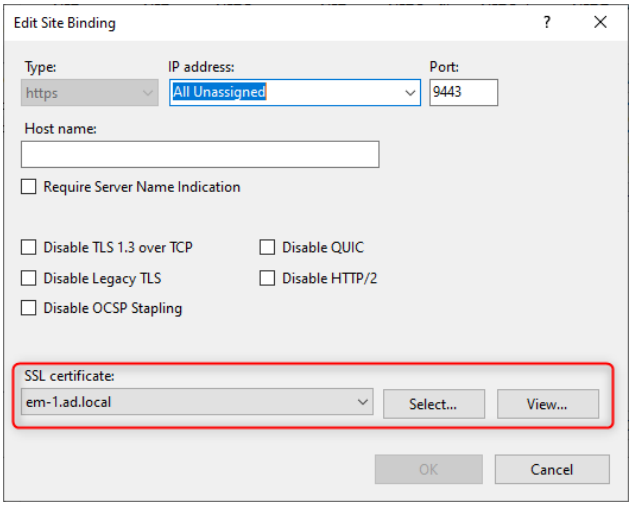


Figure 27: IIS Manager SSL Certificate Selection

- In this dropdown, select the certificate that you imported to this server following sec. 5.3 above. Clicking on the *"Select…"* button shows more details of the existing certificates and helps with finding the right one. To verify, navigate to your VBEM web UI using the https URL (default: `https://[FQDN of your VBEM server]:9443`). This will open the login page as usual, but you should see the lock icon in the address bar of your browser. By clicking on this icon, you can check the certificate that VBEM has presented to the browser. In the Microsoft Edge browser it looks like this:

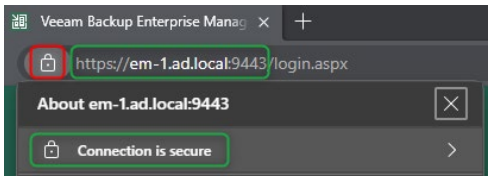


Figure 28: MS Edge Address Line of VBEM's https URL

- Click on the arrow to the right to display the details of the certificate, which should be the one you have just configured.

Please note that this will only work if you also installed the CA certificate on your local client (see sec. 5.2). This is required to ensure your client computer trusts the certificate that VBEM is presenting to your browser.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

6 Setting up Keycloak

There are many different ways to run Keycloak and that means there are also many ways to install it. For my lab setup, I went for the setup as a Docker container. I recommend following this guide to get yourself familiar with a very simple Keycloak container installation. One of the advantages of the container-based approach is that if something goes wrong, you can simply go back and start again without having to install a completely new server.

To make this even easier, I used "`docker compose`" to create a reproducible Keycloak installation. I also used *"bind-mount"* volumes to preserve Keycloak's configuration data on the Docker host, even after a container rebuild. In a production environment, you would use an external database (e.g. PostgreSQL) to store Keycloak's persistent data. However, since this approach would also require the creation of a customized container image (as described here), I didn't bother. Here is the "`docker-compose.yaml`" file that I started with in my lab:

Contents of file `docker-compose.yaml`

```
version: "3.9"


services:
  keycloak:
    image: "keycloak/keycloak:23.0"
    container_name: keycloak
    restart: unless-stopped
    ports:
      - 8080:8080
      - 443:8443
    volumes:
      - ./data/:/opt/keycloak/data/
      - ./certs/:/etc/x509/https/
    env_file:
      - ./keycloak.env
    command: start-dev
```

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

I used this file in the `/opt/keycloak` folder of a virtual server running Ubuntu 22.04.3 with 2 vCPUs and 4 GB RAM. The system requirements for my small lab environment are certainly lower, but starting a Keycloak instance can put quite a strain on the CPU due to the Java runtime overhead.

I have also created 2 subfolders and some additional files in the newly created folder, so the whole structure now looks like this:

Folder structure on the Docker host:

```
/opt/keycloak
      ├── data/                  <-- Keycloak will store its configuration here
      ├── certs/
      │   ├── fullchain.pem      <-- contains certificate chain
      │   └── privkey.pem        <-- contains certificate and private key
      ├── docker-compose.yaml
      └── keycloak.env
```

The text file `keycloak.env` configures some environment variables required by Keycloak:

```
KEYCLOAK_ADMIN=[your admin account name]
KEYCLOAK_ADMIN_PASSWORD=[your admin account password]
KC_HOSTNAME=[FQDN of your docker host]
KC_HTTPS_CERTIFICATE_FILE=/etc/x509/https/fullchain.pem
KC_HTTPS_CERTIFICATE_KEY_FILE=/etc/x509/https/privkey.pem
QUARKUS_TRANSACTION_MANAGER_ENABLE_RECOVERY=true
TZ=Europe/Berlin
```

You need to customize some of the settings according to your setup:

Table 1: Keycloak Environment Variables

Variable	Description
KEYCLOAK_ADMIN	Keycloak admin username
KEYCLOAK_ADMIN_PASSWORD	Keycloak admin password
KC_HOSTNAME	DNS FQDN of the docker host
KC_HTTPS_CERTIFICATE_FILE	Full path to the file containing the certificate. Note that in my example, this path resolves inside the

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

Variable	Description
	container to one of the mounted volumes in the `docker-compose.yaml` file above (where I placed the file `fullchain.pem` from fig. 20).
KC_HTTPS_CERTIFICATE_KEY_FILE	Full path to the file containing the private key of the certificate (`privkey.pem` from fig. 19 in my example).
QUARKUS_TRANSACTION_MANAGER_ENABLE_RECOVERY	Set this to `true` to make Keycloak's internal database more robust against unexpected shutdowns (not needed when using an external database).
TZ	Optional timezone setting (Keycloak is using UTC by default and ignores the docker host's timezone setting)

Don't forget to copy the two exported certificate files from sec. 5.1.2 (fig. 19 and fig. 20) into the `/opt/keyloak/certs/` folder as shown above.

Now start the Keycloak container using the `docker-compose.yaml` file:

```
cd /opt/keycloak
```

followed by

```
docker compose up
```

The first initialization phase can take some time, so be patient. After a minute or two, access should be possible via http on port 8080 or https on standard port 443 (as specified in the `docker-compose.yaml` file above):

- `http://[FQDN of your docker host]:8080`
- `https://[FQDN of your docker host]` (no port required as we configured the default https port 443)

If the https connection is not working immediately, you can still fallback using the http URL and troubleshoot later. As soon as https works fine, you should delete the line containing "`- 8080:8080`" from the `docker-compose.yaml` to disable unsecured http access to your Keycloak instance.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

The Keycloak welcome page should now be displayed in your browser:

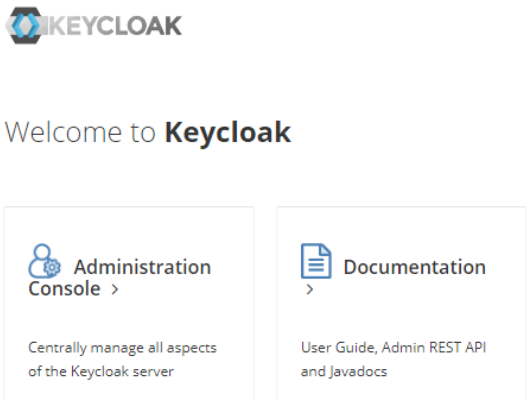


Figure 29: Keycloak Welcome Page

Congratulations if you've made it this far, you now have an operational Keycloak instance!

6.1 Creating a Custom Realm

To start Keycloak administration, click on *"Administration Console"* on the welcome page. This will prompt for login credentials, so use the credentials you've defined in your `keycloak.env` file above. On the top left of Keycloak's administration console page, you will see a dropdown box showing *"master"*. This is the main administrative "Realm" where all configurations are done that affect the Keycloak instance globally, for example regarding the administrative users. Fig. 30 shows our current user account which we've created via our `keycloak.env` file:

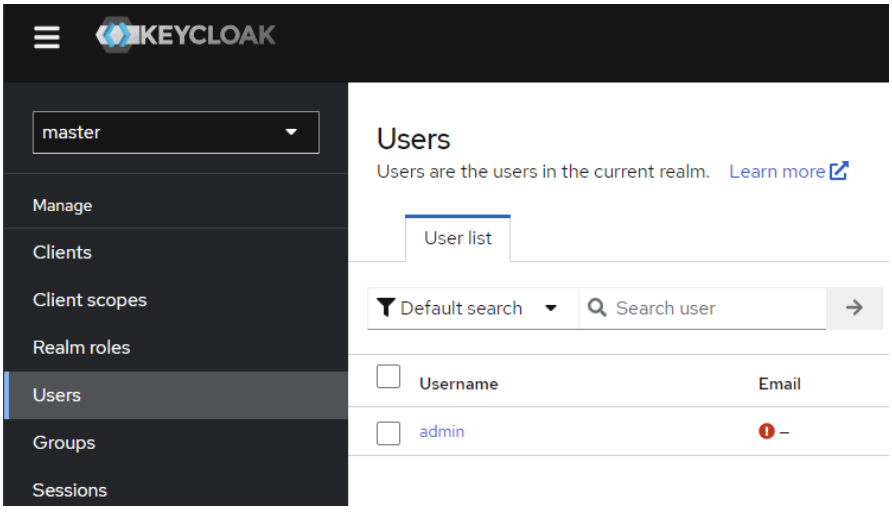


Figure 30: Keycloak Users in the "master" Realm

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

A realm in Keycloak is a container for a set of users, credentials, roles, and groups. It is used to logically isolate a specific space, somehwat like multi-tenancy, where different realms are isolated from each other and can only manage and authenticate the users that they control.

To start configuring Keycloak for our example scenario, we should therefore create a new realm by clicking on the mentioned dropdown and click on the *"Create realm"* button, which brings us to the "Create realm" page:

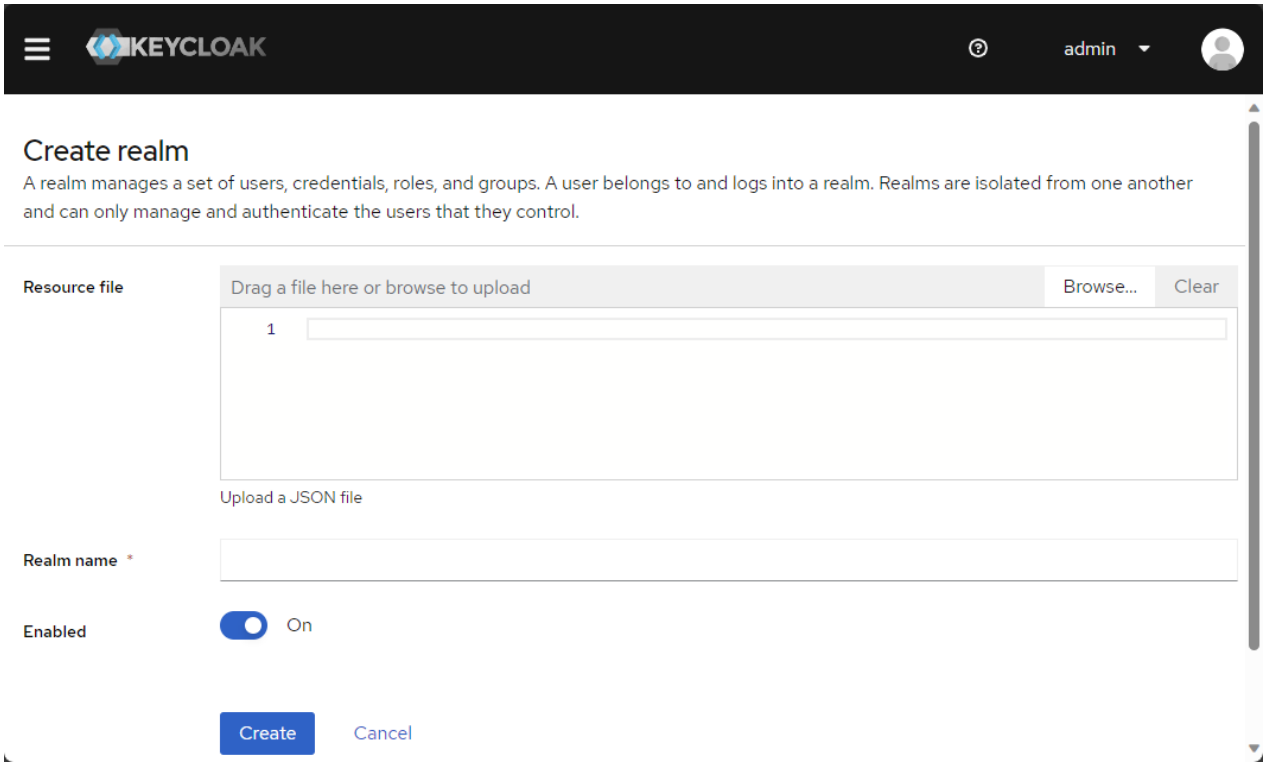


Figure 31: Create New Realm in Keycloak

This page allows you to upload a JSON file with an existing realm configuration. We do not have such a file (yet), but it is good to know that we can (almost) completely restore a realm using this method, we just need to make sure that we have exported the required file at some point.

We are starting from scratch, so the only information we can enter here is our desired realm name and whether we want to activate it immediately via the "Enabled" switch. Once you have entered a realm name and clicked *"Create"*, the new realm will appear in the dropdown box at the top left. I have used the realm name "vbem-saml" and selected it as the active realm for further work.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

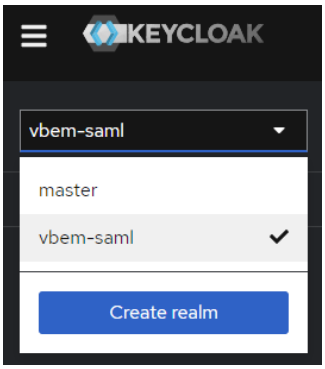


Figure 32: New Realm Selected

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

7 Building the Relationship between Keycloak and Veeam Backup Enterpise Manager

Now we are finally in a position to establish the relationship between Keycloak and Veeam Backup Enterprise Manager. First of all, we need to test whether VBEM accepts the certificate from Keycloak. The easiest way to do this is to verify a browser session from VBEM to Keycloak via secure https (this is not a requirement!). Open a browser on your VBEM server and enter the https URL of our Keycloak instance. As soon as you see Keycloak's welcome page without getting any warning (e.g. like "this connection is not secure") and clicking on the lock symbol in the address line of your browser shows "Connection is secure", you're good to move on:

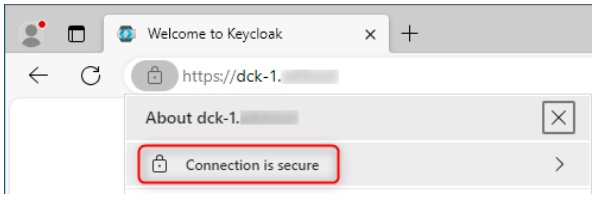


Figure 33: Browser Address Bar on VBEM Server accessing the Keycloak Welcome Page

There is no need to login to Keycloak in this browser session, we only wanted to validate that the secured https access is available. I would like to emphasize again that there are **no connectivity requirements between VBEM and Keycloak** to run the solution in a production environment (as mentioned in sec. 2.1). We only took this step to validate our certificate configuration.

To establish the relationship between Keycloak and VBEM, we need to configure both sides. Let's start with VBEM.

7.1 Download SAML Metadata Files from Veeam Backup Enterprise Manager

Login to VBEM's web portal using an account with "Portal Administrator" role and navigate to *"Configuration/Settings"* and then *"SAML Authentication"*:

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

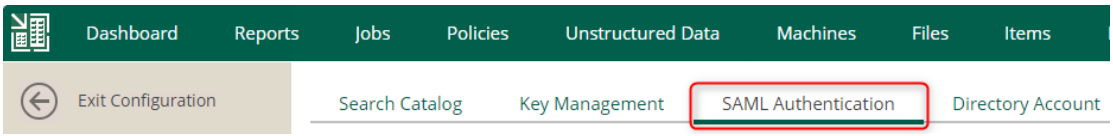


Figure 34: *SAML Authentication* Settings page in Veeam Backup Enterprise Manager

After enabling the checkbox *"Enable SAML 2.0"* at the top of the page, you will notice that there are a lot of options available for configuration. But don't worry, you don't need to enter all this information manually! On this page, there are two buttons labeled *"Download"* (see marker 1 on fig. 35):

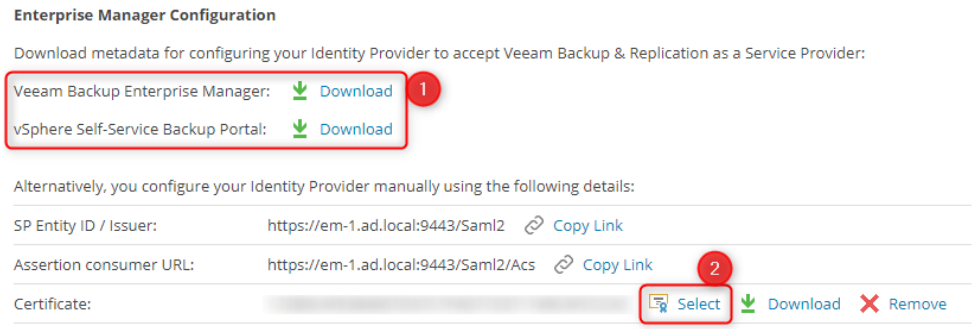


Figure 35: Metadata Download on *SAML Authentication* Settings page in Veeam Backup Enterprise Manager

These downloadable files contain all required SAML information in XML format to register VBEM as a Service Provider in Keycloak. Why two files? This is because VBEM provides two independent services, *"Veeam Backup Enterprise Manager"* and *"vSphere Self-Service Backup Portal"* (as mentioned in sec. 4.1.1) which for our example both need to be registered at the Identity Provider: Self-Service for tenants, VBEM website for internal/management users.

Important: Before we download these two metadata files, we need to make sure that they are signed by VBEM with the correct certificate. To do this, we click on the *"Select"* button at the bottom of the same page (marker 2 on fig. 35), which takes us to a selection dialog that shows all eligible "personal" certificates currently installed on your VBEM server. So before you download the two metadata files, make sure that you have selected the VBEM certificate imported in sec. 5.3 in this dialog.

Now we can use the downloaded XML files to configure the Keycloak side of the connection.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

7.2 Import SAML Metadata into Keycloak

Login to your Keycloak instance with your browser and your administrator account from sec. 6 and select the realm created in sec. 6.1. Navigate to the *"Clients"* menu item on the left and click on *"Import client"*:

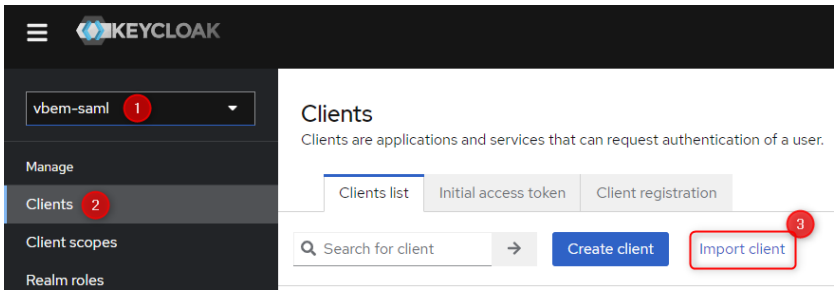


Figure 36: Import a New Client into Keycloak

This gets you to the *"Import client"* page where you can click on the *"Browse..."* button to import the first XML file you've downloaded from VBEM. As soon as you've selected the import file, most of the other configuration items on this page will update with the imported information. You only need to provide the name and an optional description into the respective input fields.

I've entered "Veeam Backup Enterprise Manager" as the name and "VBEM main portal" as the description for the first downloaded XML file from sec. 7.1, as shown on the left part of fig. 37:

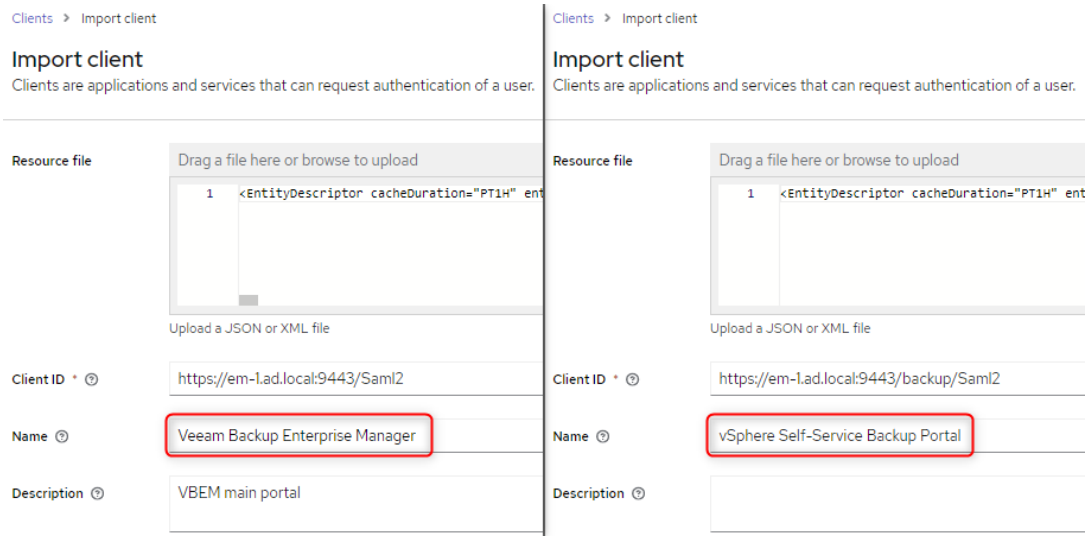


Figure 37: Client Import Details

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

Now click on the *"Save"* button and repeat the process for the second XML file. I have used the name "vSphere Self-Service Backup Portal" for this import (see right part of fig. 37, I didn't bother to enter a description here). After you have saved the second import, the two new clients with their client IDs (URLs), names and the type "SAML" should be displayed on the *"Clients"* page:

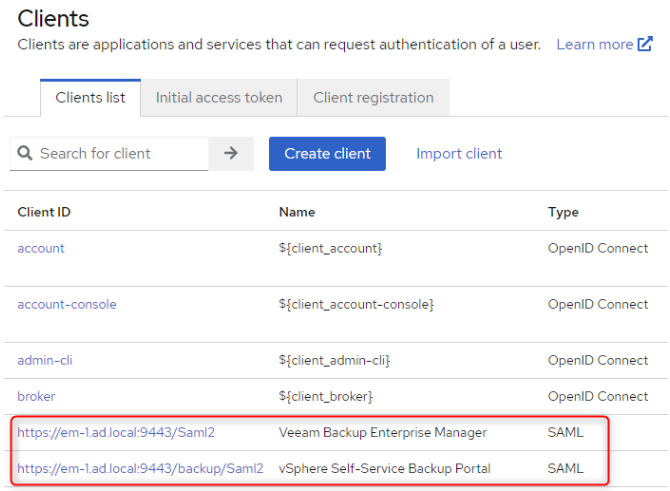


Figure 38: List of Clients in Keycloak

7.3 Download SAML IdP Metadata from Keycloak

In order to make VBEM accept logins from our IdP (Keycloak), we need to complete the configuration that we started in sec. 7.1. On the left-hand menu of our realm "vbem-saml" in the Keycloak administation UI, select *"Realm settings"* and click on the download link named *"SAML 2.0 Identity Provider Metadata"* near the bottom of the page:

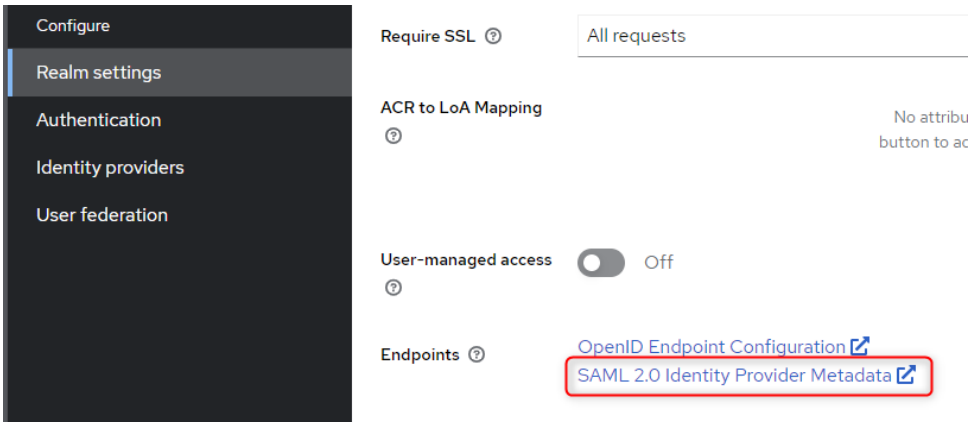


Figure 39: Download Keycloak IdP Metadata

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

Save and copy the file with extension ".xml" to a location that is accessible by your VBEM server (either locally on your VBEM server or in a shared folder accessible by your VBEM server).

7.4 Import SAML IdP Metadata into Veeam Backup Enterprise Manager

Now we can go back to Veeam Backup Enterprise Manager (logged in as "Portal Adminsitrator") and go to *"Configuration/Settings/SAML Authentication"* again (fig. 34). In the section *"Identity Provider Configuration"*, click on the *"Import from File"* button and upload the XML file we just downloaded from Keycloak in the previous step:

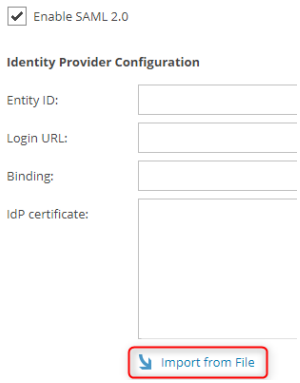


Figure 40: Import IdP Metadata into Veeam Backup Enterprise Manager

All the fields above the import button should now be populated. This is how it looks like in my lab setup:



Figure 41: Identity Provider Configuration in VBEM after Import

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

This completes the configuration of building the relationship between Keycloak and VBEM. Although we haven't added any of our external identity sources yet (the AD forests from fig. 3), we should be able to authenticate to VBEM with an internal Keycloak user, so let's set one up and test it.

7.5 Add Mapper for the "NameID" Attribute

Each SAML assertion created by an Identity Provider must include a `NameID` attribute in order to enable the Service Provider to uniquely identify the authenticated user account. So we need to tell Keycloak which user property it should use to fill in the NameID attribute. To select the correct user property, we need to know what Veeam Enterprise Manager is expecting within the `NameID` attribute. We can find this information in the VBEM User Guide (section "How It Works", step 5):

> The SAML assertion must meet the following requirements:
>
> - Contain a User Principal Name (UPN) of the user in the `<NameID>` element of the SAML response.
> - Specify that the UPN type is **Persistent**.

So we have to make sure that Keycloak forwards the content of a user's `upn` or `userPrincipalName` property in the `NameID` attribute of each SAML assertion. The user guide tells us also that the type of the attribute should be set to `persistent`, so let's add this configuration. In Keycloak, select *"Client scopes"* from your realm's main menu. You should see a list like this:

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

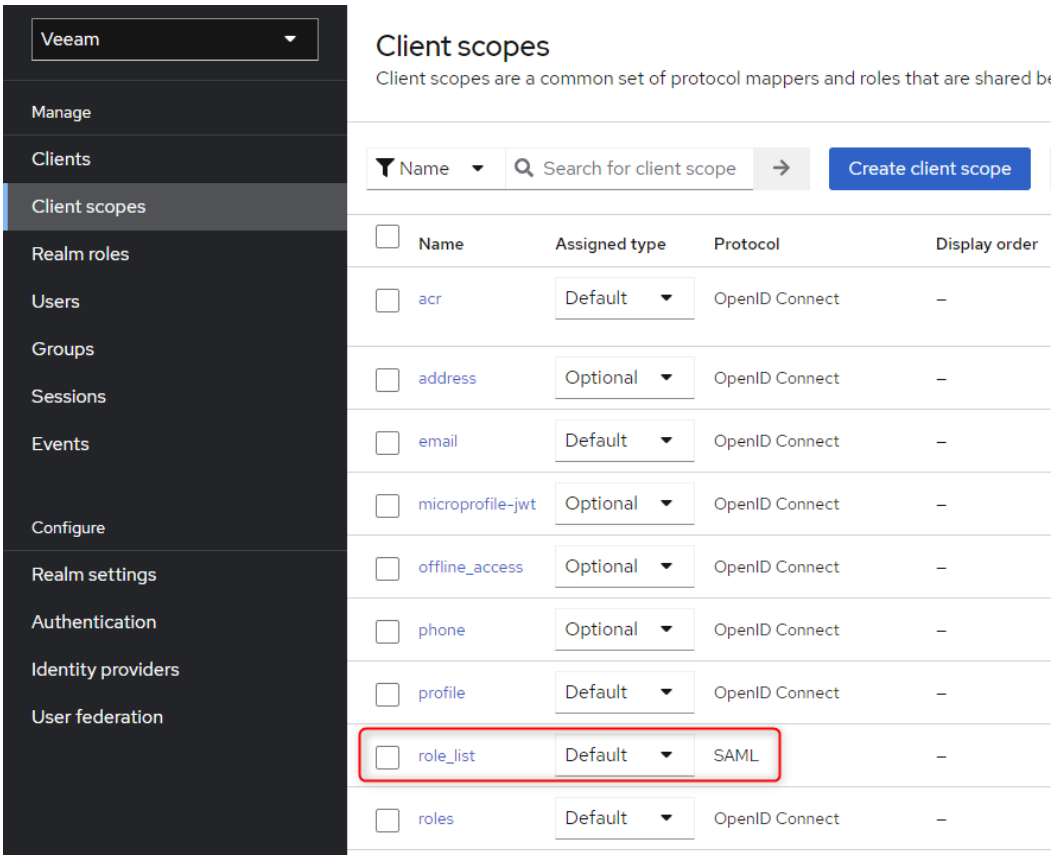


Figure 42: List of Client Scopes

We need to create a "mapper" within a client scope for the SAML protocol. As there is already a client scope of that type (named *"role_list"* as highlighted in fig. 42), we open this one by clicking on its name in the first column. Within the settings of the *"role_list"* scope we select the *"Mappers"* tab and click on the *"Add Mapper"* button and select *"By configuration"* in the drop down list.

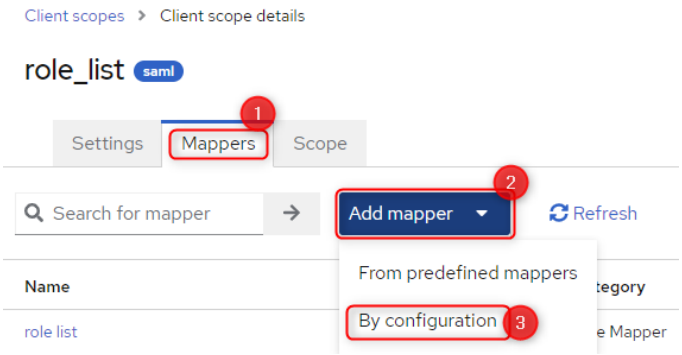


Figure 43: Add Mapper within Client Scope

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

On the next dialog, click on *"User Attribute Mapper for NameID"*:

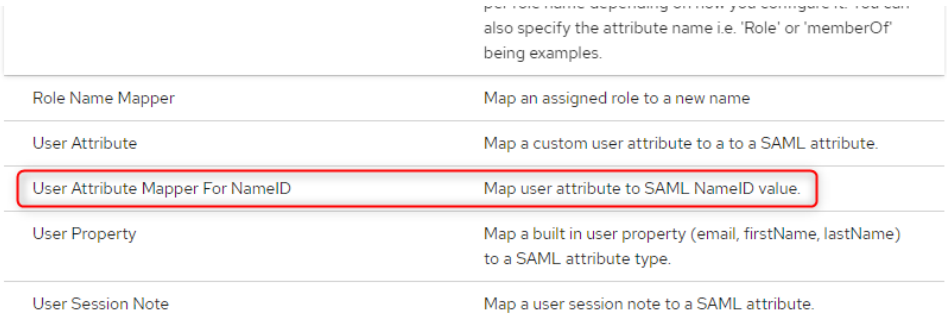


Figure 44: "Configure a new mapper" dialog

Now we have to enter the configuration of the new mapper:

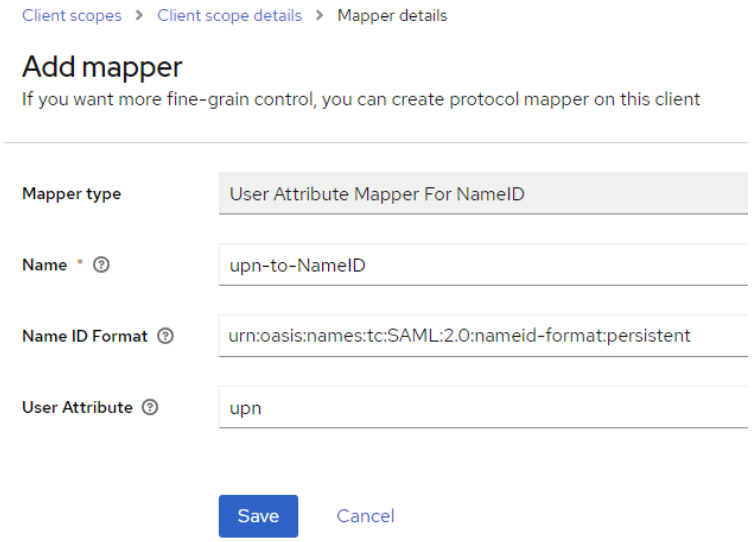


Figure 45: Mapper Configuration for `upn` to `NameID` Mapping

The following table shows the values entered in fig. 45, the important ones being *"Name ID Format"* (selectable via dropdown) and *"User Attribute"*:

Table 2: NameID Mapper Configuration

Setting	Value
Mapper type	User Attribute Mapper For NameID
Name	`upn-to-NameID`
Name ID Format	`urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

Setting	Value
User Attribute	`upn`

Don't forget to click the *"Save"* button to finish this configuration step.

7.6 Test the first SAML Authentication

For this test, we create a new "external user" with the "Portal User" role in VBEM (you could also use any other available role). Go to VBEM's configuration page, select *"Roles"* on the left hand menu and click on the *"Add..."* button at the top of the page. In the *"Add Role"* dialog, make sure you select account type *"External User"*. Then select the desired role and configure any restrictions for this test (I chose to permit only file level restores for this user):

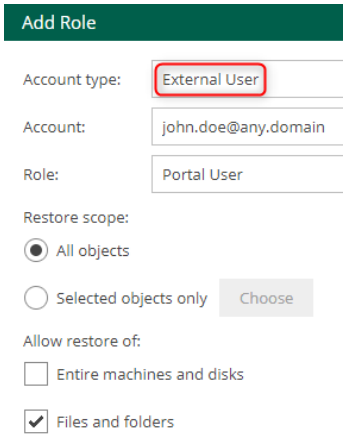


Figure 46: Create external Test User in VBEM

Please note that the user name must be entered in e-mail (or *UPN = User Principal Name*) format (`username@domain.tld`). It is only about the format, it does not have to be an existing e-mail address (in my example, `any.domain` does not even exist)!

Back to Keycloak, we create a "static" user within our realm "vbem-saml" by selecting *"Users"* on the left hand menu and clicking the *"Add user"* button. I've chosen the following configuration:

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

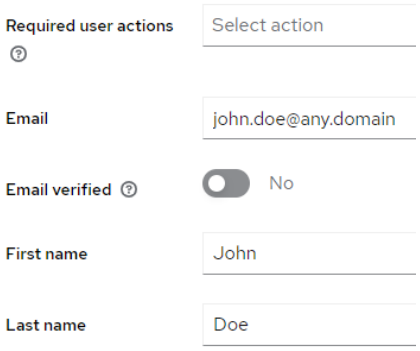


Figure 47: Create a Test User Account in Keycloak

It's important to leave the field *"Required user actions"* at its default value of *"Select action"*, otherwise Keycloak would try to perform some additional verification which we haven't configured! After clicking on the *"Create"* button at the bottom of the page the user will be created, indicated by a small alert at the top right of your browser window, and Keycloak presents the *"User details"* page:

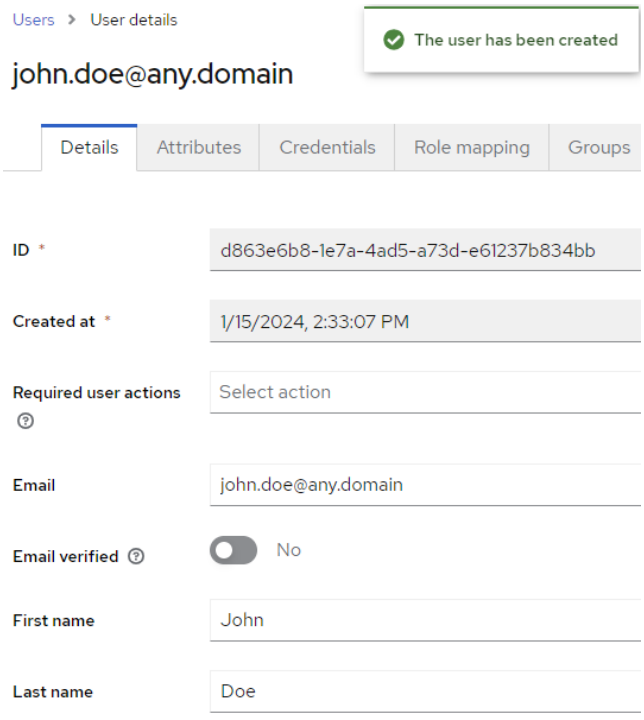


Figure 48: Keycloak Test User Details

To enable this user for login, we also have to add a password. Just select the *"Credentials"* tab in fig. 48 and click on the *"Set password"* button. If you leave the *"Temporary"* switch on this dialog enabled, Keycloak will ask the user to change the password on first logon. I disabled this setting for this test.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

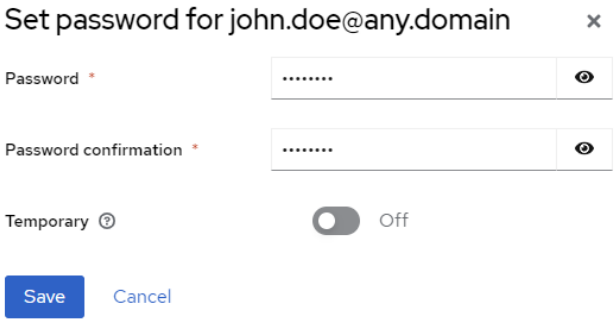


Figure 49: Set Password for Test User

Now it's time to test authenticating this user against VBEM. Please open a new "private" or "incognito" window of your browser to not interfere with any open sessions with Keycloak and/or VBEM. Enter the URL of your VBEM instance (`https://[your VBEM server]:9443/login.aspx`) and on the login screen, click on the *"Use Single Sign-On (SSO)"* button:



Figure 50: Single Sign-On Button on VBEM Login Page

This should forward you to Keycloak's login page (titled with our realm's name). Enter the test user credentials and click on *"Sign In"*:

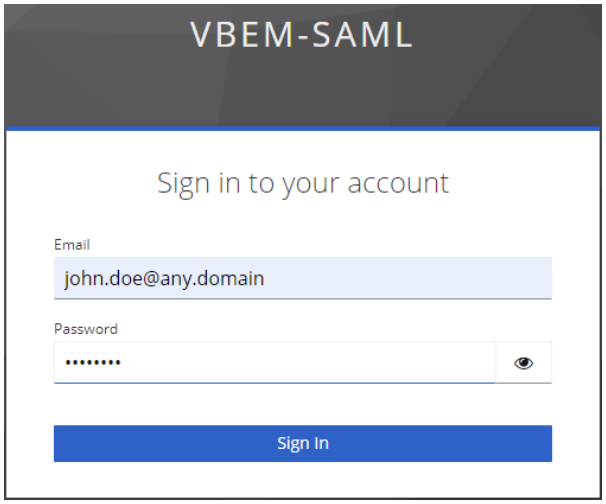


Figure 51: Single Sign-On Button on VBEM Login Page

However, this login attempt will not be successful, VBEM will respond with the following error message:

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

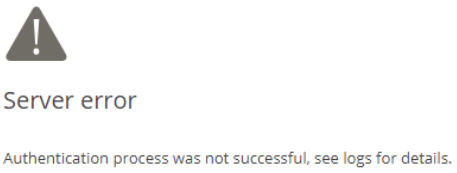


Figure 52: Login Error reported by VBEM

This is expected, because we haven't provided a upn property for the user, but this is required to be mapped to the SAML `NameID` attribute as mentioned in sec. 7.5. So let's add this property to our user account in Keycloak.

7.6.1 Add UPN Attribute to Test User Account

Go back to your Keycloak administration UI and select *"Users"* from the realms menu. To find the user, simply enter * into the search field and click on the arrow pointing to the right:

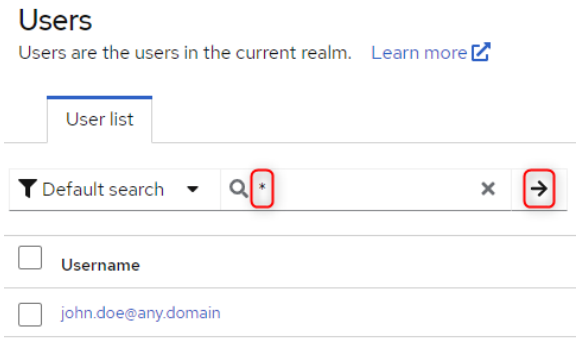


Figure 53: Get a List of Users in Keycloak

Select the *"Attributes" tab at the top and click on* "Add an attribute". Enter "upn" as the key name and the test user's email we used in fig. 47.

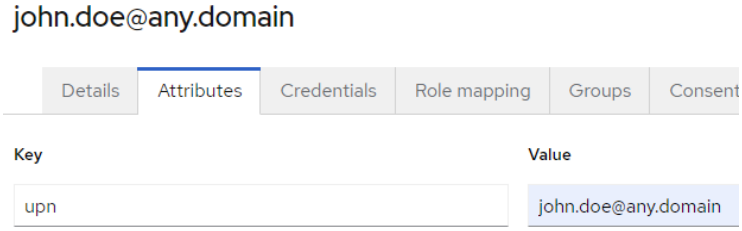


Figure 54: Add UPN attribute to Test User

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

Don't forget to click the *"Save"* button. Now try again to authenticate to VBEM (in a new private/incognito browser window) using the *"SSO"* login button and the test user credentials. This time, authentication should be successful and VBEM should show the logged-in user on the far right of its menu bar:

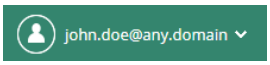


Figure 55: Successful Login to Veeam Backup Enterprise Manager

This finally confirms that we have established a valid SAML relationship between Keycloak and Veeam Backup Enterprise Manager.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

8 Keycloak Identity Broker Configuration

The configuration we have created so far is only able to authenticate users that are internal to Keycloak to Veeam Backup Enterprise Manager. However, a requirement from sec. 3.1 was that we need to enable authentication for users of some Active Directory forests via LDAP. This essentially means that we need to get Keycloak to connect to these authentication sources to select the correct source each time a user tries to log in and then perform authentication there.

The first idea would be to use the menu item *"Identity Providers"* on the administration page of your realm and add our AD forests there. However, this function only supports IdP types and protocols that Active Directory does not offer by default. Although we could add the required SAML capabilities to each AD forest by installing Active Directory Federation Service into each of them, I have chosen a simpler way called *"User Federation"*. So in a way, I'm cheating a bit by not using a proper SAML identity broker role, but a different keycloak capability that is very useful for the requirements we've set.

8.1 Prepare Keycloak for Groups: Create "Group List" Mapper

According to our requirements of sec. 3.1, we need to make sure that the users' group memberships are included in the SAML assertions created by Keycloak. This can be achieved by creating an additional "Client Scope" mapper in a similar way as we did for the `NameID` property in sec. 7.5 before:

Navigate to your realm's *"Client scopes"* and enter the scope named *"role_list"* from the list of scopes (see fig. 42). Again, we add a new mapper to this scope by clicking on the *"Add mapper"* button within the *"Mappers"* tab and select *"By configuration"* from the dropdown (see fig. 43). But this time we select *"Group list"* from the list in the *"Configure a new mapper"* dialog:

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

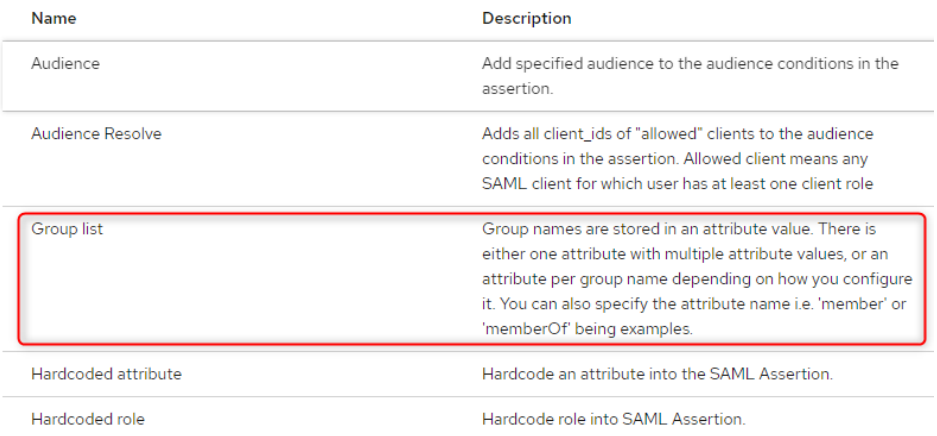


Figure 56: "Group List" Mapper Type Selection

The configuration of this new mapper has to be entered like this:

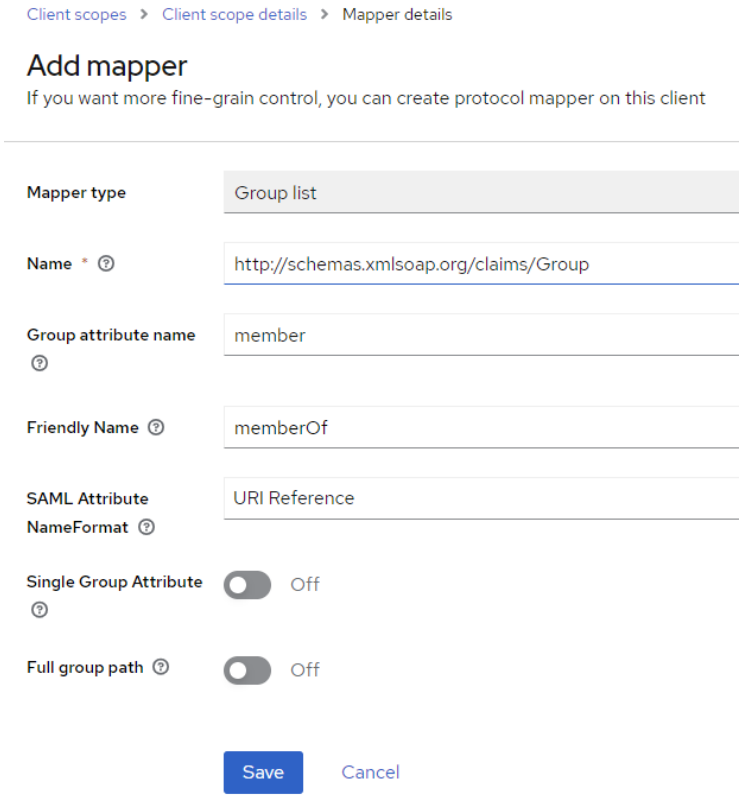


Figure 57: "Group List" Mapper Configuration

Make sure that you enter the values exactly as given in fig. 57 above (see tbl. 3 below for copy/paste). Hit the *"Save"* button to finsh this configuration.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

Table 3: Group List Mapper Configuration

Setting	Value
Mapper type	Group list
Name	`http://schemas.xmlsoap.org/claims/Group`
Group attribute name	`member`
Friendly name	`memberOf`
SAML Attribute NameFormat	`URI Refrence`
Single Group Attribute	`Off`
Full group path	`Off`

Note: The value entered into the *"Name"* field in fig. 57 needs to match the corresponding value of *"Group claim type"* in VBEM's Advanced Settings on the SAML configration page:

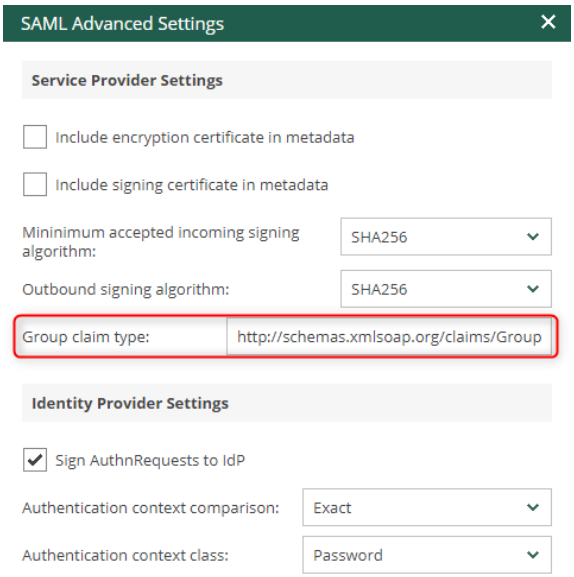


Figure 58: "Group Claim Type" in VBEM's Advanced SAML Settings

8.2 Preparing Active Directory Forests

First, we need to allow Keycloak to connect to each of the AD forests defined in our scenario (fig. 3). This requires authentication, so we need an account in each forest that Keycloak can use for these purposes. I created an account named "`svcKeycloak`" in each forest with read-only access to the AD. This account only needs to be a member of the "Domain Users" group (you can restrict it even further to prevent access outside the already defined organizational units where our users and groups are located).

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

8.3 Create User Federation for each Active Directory Forest

For a complete federation configuration, we need some more information from each AD forest:

Table 4: User Federation Parameters needed by Keycloak

Keycloak Federation Setting	Description
UI display name	friendly name within Keycloak's administrative UI
Connection URL	ldap connection string to connect to selected Domain Controller (`ldap://[FQDN of DC]` or `ldaps://[FQDN of DC]`)
Bind DN	"distinguished name" attribute (DN) of the user account Keycloak should use to connect to the Domain Controller
Bind credentials	password of this account
Users DN	"distinguished name" attribute (DN) of the organizational unit (OU) where Keycloak will look for user accounts eligible for authentication
Search scope	either "One Level" or "Subtee"; defines whether Keycloak will also look for users in OUs nested beneath the given "Users DN"

There are a lot more settings available, but tbl. 4 should be sufficient to setup a working federation. Please note that the *"Users DN"* setting defines Keycloak's "starting point" to find eligible user accounts.

For each AD forest, create a new user federation by selecting the *"User federation"* menu item and clicking on *"Add new provider" and then "LDAP"*:

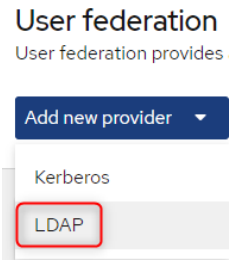


Figure 59: Add New LDAP User Federation to Keycloak

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

Enter the following information for each user federation. Based on your setup, some items might be different! This table shows my lab setup according to fig. 3 and the distinguished names of the OUs containing our users (we've taken a note of those in sec. 4.1.2 and sec. 4.1.3):

Table 5: Configuration Parameters for each User Federation in Keycloak

Keycloak Federation Setting	Value
	Management Forest:
UI display name	mgmt.corp.lab
Connection URL	`ldap://adm-dc-1.mgmt.corp.lab`
Bind DN	`CN=svcKeyCloak,OU=Users,OU=corp,DC=mgmt,DC=corp,DC=lab`
Bind credentials	[password of `svcKeycloak` account in this forest]
Users DN:	`OU=Users,OU=corp,DC=mgmt,DC=corp,DC=lab`
	Tenant 1:
UI display name	tenant.lab
Connection URL	`ldap://tenant-dc.tenant.lab`
Bind DN	`CN=svcKeycloak,OU=Users,OU=myTenantOrg,DC=tenant,DC=lab`
Bind credentials	[password of `svcKeycloak` account in this forest]
Users DN:	`OU=Users,OU=myTenantOrg,DC=tenant,DC=lab`
	Tenant 2:
UI display name	tenant-2.lab
Connection URL	`ldap://tenant-2-dc.tenant-2.lab`
Bind DN	`CN=svcKeycloak,OU=Users,OU=myOrg,DC=tenant-2,DC=lab`
Bind credentials	[password of `svcKeycloak` account in this forest]
Users DN:	`OU=Users,OU=myOrg,DC=tenant-2,DC=lab`

Note that I started with unsecure `ldap` instead of `ldaps` to ensure everything is working fine before forcing the secure method.

We are almost there! According to our requirements of sec. 3.1, we need to make sure that the users' group memberships are included in the SAML assertions created by Keycloak. This can - again - be achieved by creating a *Mapper* for each of the user federations, so let's do that now.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

8.3.1 Add Mappers for Group Attribute

Open the federation settings by selecting *"User federation"* on the realm's menu and click on the federation's name. Select the *"Mappers"* tab at the top of the page and click on *"Add mapper"*. Enter a name and select mapper type *"group-ldap-mapper"*. For each user federation, enter the appropriate settings from the following table (according to your setup, some items might be different!):

Table 6: Group Mapper Settings for each User Federation in Keycloak

Mapper Setting	Value
	for all federations:
Name	`group-mapper`
Mapper type:	`group-ldap-mapper`
Group Name LDAP Attribute	`DN`
Group Object Classes	`group`
Membership LDAP Attribute	`member`
Membership Attribute Type:	`DN`
Membership User LDAP Attribute Type:	`cn`
Mode	`READ_ONLY`
User Groups Retrieve Strategy	`LOAD_GROUPS_BY_MEMBER_ATTRIBUTE`
Member-Of LDAP Attribute:	`memberOf`
—	—
	for management forest federation:
LDAP Groups DN	`OU=Groups,OU=corp,DC=mgmt,DC=corp,DC=lab`
	for Tenant 1 federation:
LDAP Groups DN	`OU=Groups,OU=myTenantOrg,DC=tenant,DC=lab`
	for Tenant 2 federation:
LDAP Groups DN	`OU=Groups,OU=myOrg,DC=tenant-2,DC=lab`

It's important to notice that the *"LDAP Groups DN"* setting defines the starting point (organizational unit) for Keycloak to search for eligible user groups in each federated AD (we've mentioned that earlier in sec. 4.1.2 and sec. 4.1.3).

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

8.3.2 Add Mappers for UPN Attribute

We've learned in sec. 7.6 that we need Keycloak to use the `upn` property to fill in the SAML `NameID` attribute, so we needed to add this attribute explicitly (see sec. 7.6.1) to our test user account. The users of the federations we just created already have a `upn` property, as this is a default property of any Active Directory user account. Unfortunately, Active Directory refers to this property with the name `userPrincipalName` instead of just `upn` when queried via LDAP.

Therefore, we need to create another mapper within Keycloak for each user federation to ensure that the `userPrincipalName` property (as read by each AD federation) is correctly translated and forwarded as a property named `upn`, in order to be used as the `NameID` attribute.

Table 7: UPN Attribute Mapper Settings for each User Federation in Keycloak

Mapper Setting	Value
	for all federations
Name	`upn-mapper`
Mapper type:	`user-attribute-ldap-mapper`
User Model Attribute	`upn`
LDAP attribute	`userPrincipalName`

This concludes the configuration for each federation, and you should now be able to authenticate AD users from each of the forests. You can test authentications for all users of the different roles we've created in sec. 4.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

9 Additional Configuration Options

9.1 Securing Mappers via LDAP Filters

The example scenario in this document describes how users can gain access to Veeam Backup Enterprise Manager based solely on the fact that they are a member of a specific user group in their source Active Directory. This could be a potential security vulnerability: If a user group defined in one tenant AD is granted access to VBEM, how can we ensure that members of another user group defined in a completely different tenant AD but using the exact same naming convention (i.e. "spoofing" another tenant's group membership) are not granted the same access rights?

One might think that this is not a serious threat, as the groups are already restricted by the fact that the parameter `LDAP Groups DN` (see tbl. 6) refers to a specific organizational unit within the AD of the legitimate tenant. However, this does **not** prevent, for example, a malicious administrator from creating a "fake" group exactly within this permitted organizational unit. All members of such a fake group would be considered valid by Keycloak. Since the SAML assertion of these users would include these fake group memberships, they would be granted access to resources within VBEM that belong to a different tenant!

To mitigate this threat, it is recommended to configure an LDAP filter string as part of the group mapper settings that I described in sec. 8.3.1. Such a filter allows to restrict the "group memberships" attribute of the SAML assertions forwarded by Keycloak, either by defining a naming scheme dedicated to each client, or even by hard-coding (single or multiple) group names to be accepted. Other group attributes than a group's name could also be used to create such a filter. Here are a few examples:

Table 8: Examples of LDAP Filters in Group Mappers

Value of `LDAP Filter` Setting of Group Mapper	Effect
`(cn=CorpABC-VBEM-*)`	Only members of groups named with the prefix "`CorpABC-VBEM-`" will be included (`cn` is the "commonName" attribute, "`*`" character defines a wildcard).
`(|(cn=GroupA)(cn=GroupB)`	Only members of `GroupA` or `GroupB` are taken into account. The "`|`" character defines the logical "or"

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

Value of `LDAP Filter` Setting of Group Mapper	Effect
	operator, i.e. users must be members of either GroupA **or** GroupB **or both** to match this filter.
`(objectGUID=abcdef-ghi-jkl-mno-pqrst)`	Only members of the group with attribute `objectGUI` being exactly equal to the given GUID match this filter.

A very good overview of how to create your own customized LDAP filters can be found here.

9.2 Enforcing Secure LDAPS (Keycloak → AD)

In sec. 6, we included the certificates created earlier into the Keycloak configuration. However, the setting we used there only affects the https-based communication between Keycloak and its communication partners. To secure the ldap communication between Keycloak and the Domain Controllers as part of the user federation setup (sec. 8), we also need to make sure that Keycloak can use a valid certificate and private key to sign and encrypt outgoing ldap requests.

As Keycloak is based on Java technology, it is using a "keystore" file to manage keys and certificates. The tool needed to create and import certificates and keys into such a keystore file is called "`keytool`" and it is available within the docker container we've created. However, if you're not familiar with this tool it can be quite challenging to get to the desired results.

I've used a shell script within my Keycloak container to do most of the work. The script was originally found on this thread on stackoverflow.com but I modified it to work in our scenario.

Contents of shell script `create-truststore.sh`

```
#!/bin/bash

# 1) $KEY : Private key in PEM format
# 2) $LEAFCERT : Certificate for private key, also in PEM format
# 3) $CACERT : CA certificate of CA that issued/signed $LEAFCERT
#
# We want to create a fresh Java keystore $TARGET_KEYSTORE with the
# password $TARGET_STOREPW
#
# The keystore must contain: $KEY, $LEAFCERT, $CACERT

KC_FQDN=FQDN of your docker host needs to be added here!
```

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

```
TARGET_KEYSTORE=/etc/x509/https/kc-keystore.jks
TARGET_STOREPW=changeit

TLS=/etc/x509/https

KEY=$TLS/privkey.pem
LEAFCERT=$TLS/fullchain.pem
CACERT=$TLS/RootCA.crt

# Create PKCS#12 file to import using keytool later

TMPPW=$$ # Some random password

PKCS12FILE=`mktemp`

if [[ $? != 0 ]]; then
  echo "Creation of temporary PKCS12 file failed -- exiting" >&2; exit 1
fi

TRANSITFILE=`mktemp`

if [[ $? != 0 ]]; then
  echo "Creation of temporary transit file failed -- exiting" >&2; exit 1
fi

cat "$KEY" "$LEAFCERT" > "$TRANSITFILE"

openssl pkcs12 -export -passout "pass:$TMPPW" -in "$TRANSITFILE" -name etl-web > "$PKCS12FILE"

/bin/rm "$TRANSITFILE"

# Print out result for fun! Bug in doc (I think): "-pass " arg does not work, need "-passin"

openssl pkcs12 -passin "pass:$TMPPW" -passout "pass:$TMPPW" -in "$PKCS12FILE" -info

# Import contents of PKCS12FILE into a Java keystore.

if [[ -f "$TARGET_KEYSTORE" ]]; then
  /bin/rm "$TARGET_KEYSTORE"
fi

keytool -importkeystore \
   -deststorepass  "$TARGET_STOREPW" \
   -destkeypass    "$TARGET_STOREPW" \
   -destkeystore   "$TARGET_KEYSTORE" \
   -srckeystore    "$PKCS12FILE" \
   -srcstoretype  PKCS12 \
   -srcstorepass  "$TMPPW" \
   -alias $KC_FQDN

/bin/rm "$PKCS12FILE"

# Import the CA certificate.

echo "Importing CA certificate"
```

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

```
TT=-trustcacerts

keytool -import $TT -storepass "$TARGET_STOREPW" -file "$CACERT" -keystore "$TARGET_KEYSTORE" -alias RootCA

# Print contents

echo "Listing result"

keytool -list -storepass "$TARGET_STOREPW" -keystore "$TARGET_KEYSTORE"
```

Before you proceed with the following steps, I strongly recommend to take a backup (or at least a snapshot if running virtual) of your current docker host!

1. Copy the above script with filename `create-keystore.sh` to the `/opt/keycloak/certs` folder of your docker host. You also might have to mark it as "executable" by issuing the command

   ```
   chmod +x create-keystore.sh
   ```

2. Copy the CA certifcate (exported in sec. 5.1.1) to the same folder (filename `RootCA.crt` in the following example) and double-check if you need to change anything within the script (especially filenames, paths, the truststore password in variable `TARGET_STOREPW` and your docker host's DNS-FQDN in variable `KC_FQDN`).

3. On your docker host, start an interactive shell within the Keycloak container:

   - Retrieve the container ID of your running Keycloak container by typing

     ```
     docker ps
     ```

     on your docker host. The output should look similar to this:

     ```
     CONTAINER ID   IMAGE
     07fa346ec805   keycloak/keycloak:23.0
     ```

   - Here you find the container ID in the first column of the output, in my case it's `07fa346ec805`.

     Start a shell within the container

     ```
     docker exec -it 07fa346ec805 bash
     ```

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

and you should get to a shell prompt:

```
bash-5.1$
```

4. Now we're inside the container. Enter the folder where the certificates are stored and list its contents

```
cd /etc/x509/https
ls -la
```

(Note that this is one of the bind-mounted folders on your docker host.) The output of the `ls` command should show the files in this folder

```
drw-r-sr-x 2 root     ftp    4096 Jan 10 19:13 .
drwxr-xr-x 3 root     root   4096 Jan 12 21:01 ..
-rwxrwxrwx 1 keycloak users  2583 Jan 23 19:10 create-keystore.sh
-rw-r--r-- 1 root     ftp    4053 Jan 10 19:13 fullchain.pem
-rw-r--r-- 1 root     ftp    5312 Jan 10 19:13 privkey.pem
-rw-r--r-- 1 root     ftp    1988 Jan  2 13:24 RootCA.crt
```

5. Run the script

```
./create-keystore.sh
```

Confirm with the `ls` command that there is now a new truststore file in the current folder with a filename of `kc-keystore.jks`. Then exit the container shell by typing `exit`.

6. Finally, we need to make Keycloak use the new truststore. This needs an adjustment in our `docker-compose.yaml` file from sec. 6:

Change the last line of the file `docker-compose.yaml` from

```
    command: start-dev
```

to (this is still a single line!)

```
    command: start-dev --spi-truststore-file-file=/etc/x509/https/kc-keystore.jks --spi-truststore-file-password=password --spi-truststore-file-hostname-verification-policy=STRICT
```

and don't forget to change the `password` in this line to the one you've chosen to create the truststore file. Also make sure that you keep the original indentation of this line as shown in sec. 6 (4 spaces)!

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

7. Changes to the docker-compose file require a restart of the docker-compose stack, simply enter

   ```
   docker compose down
   ```

   followed by

   ```
   docker compose up
   ```

9.2.1 Test LDAPS Connection

You should now be able to test secure LDAPS from Keycloak against the Active Directory Domain Controllers. In Keycloak's web UI, go to the realm we created, select *"User federation"* from the menu and open one of the federations we created earlier. In the section *"Connection and authentication settings"*, simply change the connection string's protocol prefix. For example, change for the first tenant of our scenario

Connection URL `ldap://tenant-dc.tenant.lan` (unsecure LDAP)

to

Connection URL `ldaps://tenant-dc.tenant.lan` (TLS secured LDAPS)

Further down on the same page you'll find a button `Test authentication` for a quick test of the new settings. To ensure that unsecure LDAP will be blocked by the AD Domain Controllers, see this Microsoft article (which I already mentioned in sec. 5.4). If everything worked out as desired, you should see the following popup on the top right of your Keycloak UI:



Figure 60: Successfully connected to LDAP

Do not worry that it does not explicitly say "LDAPS", but only "LDAP". What is important is the fact that the authentication test worked with an LDAPS connection string.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

10 Conclusion

It takes a lot of preparation and configuration to establish a working relationship between Keycloak and Veeam Backup Enterprise Manager that can authenticate users from multiple, untrusted AD forests. But the effort is worth it if you have single sign-on requirements like those in sec. 3.1. This is because it allows you to delegate VBEM user management to the tenant or management AD administration. You only need to set up a single ("external group" based) role or tenant in VBEM and all users who are members of the referenced groups in their "home" ADs will automatically have access to VBEM without the need of additional user accounts, so it's a real Sngle-Sign-On solution.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

11 Addendum

11.1 Troubleshooting

11.1.1 General Advice Regarding Keycloak Configuration Changes

It can sometimes be very frustrating that after a small change to the configuration in Keycloak, nothing seems to work anymore. Even basic functions that you think are not affected by the change may no longer work.

Please note, however, that you must export the "SAML 2.0 Identity Provider Metadata" again and import it into Veeam Backup Enterprise Manager *every time* you make a change in Keycloak in the "Realm settings" or "Clients" area (except when adding or changing mappers). The same applies vice versa: If you change anything on the "SAML Authentication" settings page in Veeam Backup Enterprise Manager, it is likely that you need to download the two metadata files again and re-import them into Keycloak (see sec. 7).

11.1.2 Keycloak container stops immediately after startup

This can happen on the first container start if the permissions on the `data` subfolder don't allow keycloak to create its database structure. Check the output of "`docker logs <your keycloak container id>`" if there are any messages regarding "`initialization of folder h2`", or any other permission denied errors. To solve this issue, simply use the following command on your docker host (this makes sure that *everyone* has access to the folder, so it's of course **NOT** recommended to use in a production environment):

```
sudo chmod 777 /opt/keycloak/data/
```

11.1.3 Unsigned LDAP Connections are not blocked

Symptom: You've disabled unsigned ldap access to the Domain Controllers following this Microsoft article as mentioned in sec. 5.4. But following the verification steps given in that article, you are still connecting flawlessly using unsigned ldap.

Resolution: Make sure that you follow the article very precisely when using Group Policy Objects to enforce the ldap signing requirement. It explicitly says that you have to edit two Group Policy Objects (GPOs):

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

- `Default Domain Controllers Policy`
- `Default Domain Policy`

I was caught in this trap during my lab tests because I thought it would be sufficient to put the required settings into the `Default Domain Policy` only, as this would also apply to the Domain Controllers via GPO inheritance. But it only worked correctly (i.e., starting to block unsigned ldap access) as soon as I configured **both** Group Policy Objects (and no, I didn't accidentally "block inheritance" on the Domain Controllers OU 🧐).

11.2 References

Weblinks referenced in this document:

- Veeam
  - Veeam Backup Enterprise Manager Accounts and Roles
  - Veeam Backup Enterprise Manager vSphere Self-Service Portal
  - Veeam Backup Enterprise Manager - SAML Authentication Support
- Microsoft
  - Microsoft Active Directory Federation Service
  - Enable LDAP over SSL
  - How to configure the directory to require LDAP server signing for AD DS
- Keycloak
  - Keycloak Home Page
  - Get started with Keycloak on Docker
  - Running Keycloak in a container using a customized container image
- XCA
  - XCA: X-Certificate and Key Management
  - XCA source code
- Others
  - LDAPS / Domain Controller Certificates
  - Let's Encrypt homepage
  - LDAP Search Filter Cheatsheet
  - Script to create Java keystore

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.

12 About Veeam Software

Veeam provides organizations with resiliency through data security, data recovery and data freedom for their hybrid cloud. The company provides a single platform for Cloud, Virtual, Physical, SaaS and Kubernetes environments that give businesses the peace of mind their apps and data are protected and always available so that they can keep their businesses running. Headquartered in Columbus, Ohio, with offices in more than 30 countries, Veeam protects over 450,000 customers worldwide, including 82% of the Fortune 500 and 69% of the Global 2,000. Veeam's global ecosystem includes 35,000+ technology partners, resellers, service providers, and alliance partners. To learn more, visit www.veeam.com or follow Veeam on LinkedIn *@veeam-software* and on X (aka Twitter) *@veeam*.

Please leave your feedback at our dedicated Veeam Community Forum or the Veeam Community Resource Hub!

Try Veeam for free today!

12.1 About the Author

As a Senior Solutions Architect at Veeam, Matthias is helping customers and partners with planning, designing, and implementing strategies and solutions to protect mission-critical workloads. Matthias holds an advanced degree in physics and has been in the IT industry for more than 25 years in several operations, management, and consultancy positions.

© 2023 Veeam Software.. All rights reserved. All trademarks are the property of their respective owners.