Devolutions

How to Configure Secure LDAP (LDAPS) in Active Directory with Let's Encrypt

AN ESSENTIAL PART OF HARDENING AN ACTIVE DIRECTORY ENVIRONMENT IS CONFIGURING SECURE LDAP

An essential part of hardening an Active Directory environment is configuring Secure LDAP (LDAPS). When LDAPS is enabled, LDAP traffic from domain members and the domain controller is protected from prying eyes and meddling thanks to Transport Layer Security (TLS). While the insecure LDAP protocol can provide integrity (prevents tampering) and confidentiality (prevents snooping), it is no match for TLS, which is the industry standard for security.

Just like websites secured with HTTPS, LDAPS requires X.509 certificates signed by a trusted root certificate authority to function properly. Chances are, you have heard about Let's Encrypt, which is a popular certificate authority trusted by default in all browsers. Well, here is some good news: Let's Encrypt is completely free, and it also works for LDAPS!

Private Certificate Authority

Active Directory Certificate Services (AD CS) is the most common way to create a private certificate authority inside a Windows network, but only domain-joined machines are automatically configured for trust. Any other device on your network (macOS, Linux, or even a smartphone!) will not validate the LDAPS certificate, unless the private certificate authority is installed in the system's trusted root certificates. Even then, all devices need to use the internal DNS servers. Otherwise, it may not be possible to connect to the LDAPS server using the same name found inside the server certificate, thus causing a validation failure.

The Importance of DNS

All certificates contain a common name that must match the expected name, in order to be considered valid. For example, when loading "google.com" in a browser, the hostname is resolved using public DNS servers, and the certificate contains a matching name. The certificate authority that signed the certificate for google.com (GlobalSign) had to validate ownership of the domain before doing so. This process can be manual or automated, but it always relies on a form of challenge that the domain owner must complete to prove that it has control over the DNS domain name. Since private DNS servers are excluded for obvious reasons, the only way to use a public certificate authority like Let's Encrypt for LDAPS is to ensure we can request a certificate for a public DNS domain name that will match the name of the domain controller.

Naming Your Domain Wisely

If you have ever tried to follow a "Getting Started Guide to Promoting Windows Server to a Domain Controller," then this is usually the part where you might feel a bit lost. There are many names to choose from, and all of them look very important and impossible to change afterwards if you get it wrong. Trust me: I have been there, done that! If you do not have a domain name available, then you can simply buy one from Namecheap or your favorite registrar. For this example, let us be creative and use a fictional company called "IT Help Ninja" using the "it-help.ninja" internet domain:

Company Name: IT Help Ninja

DNS Domain Name: ad.it-help.ninja

NETBIOS Domain Name: IT-HELP

Domain Controller Name: IT-HELP-DC

Domain Controller FQDN: IT-HELP-DC.ad.it-help.ninja

Domain Administrator (UPN): Administrator@ad.it-help.ninja

Domain Administrator (NETBIOS): IT-HELP\Administrator

Following Active Directory naming best practices, the best approach is to use a short subdomain of an internet domain, such as "ad," "corp," or "internal." Using unassigned public domain names like ".local" or ".loc" is not recommended, because there is no protection against future registration of the domain name. The NETBIOS name is there for backward compatibility, and is limited to 15 characters, so keep it short. The name of the domain controller is most often just the NETBIOS name with "-DC" as a suffix.

Promoting Windows Server to Domain Controller

Note: If you already have a properly configured domain controller, then you can skip this step. If you have no prior experience creating a domain controller, or could gladly use a refresher, then this section is for you.

The recommended environment is a Windows Server 2019 Core VM with a public IP address. The Windows Server GUI is not required, and so installing it is a personal choice.

Start by reviewing the machine configuration:

- The machine is configured with the intended final name (IT-HELP-DC).
- The machine is configured with a static IP address on the local network.

Once promoted to a domain controller, the machine name cannot be changed. Since the domain controller becomes a DNS server, it needs to be reachable using a static IP by other machines inside the domain.

Install the Active Directory Domain Services feature, including the management tools:

```
Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
```

Create the new Active Directory forest with the Install-ADDSForest command. **Do not merely copy/paste this**

line! Change the parameter values to fit your intended domain name. If you use the wrong forest name, then the easiest way to fix the problem is to create a new VM and start over from scratch.

```
Install-ADDSForest -DomainName «ad.it-help.ninja» -DomainNetbiosName «IT-HELP» -InstallDNS
```

You will be prompted for the Directory Services Restore Mode (DSRM) password during the installation ("SafeModeAdministratorPassword"). Use a password manager to generate a strong password, and then save it for later. The machine will reboot to complete the installation process. If the next remote login appears stuck on "Please wait for Group Policy Client," just wait a little longer – it is normal.

Congratulations, you now have a domain controller! If this is your first time doing it, welcome to the club.

Requesting Certificate from Let's Encrypt

Obtaining a certificate from Let's Encrypt is always done using the ACME protocol. This automated process can use two different ways of validating domain ownership:

- DNS challenge
- HTTP challenge

The DNS challenge involves setting DNS records to a special value, while the HTTP challenge requires hosting a file with a special value at a known path. The downside of the DNS challenge is that the application needs write access to the DNS records, and the downside of the HTTP challenge is that the application needs access to the HTTP standard port 80. Since domain controllers generally run a restricted amount of services, we can afford the luxury of using TCP port 80 for the ACME challenge.

Add a Windows firewall exception for inbound traffic on TCP port 80:

```
netsh advfirewall firewall add rule name=»HTTP inbound» dir=in action=allow protocol=TCP localport=80
```

Do not forget to add a firewall exception in other places that require it, such as the Azure Network Security Group (NSG), if this is an Azure VM. Verify that no other program is currently listening on TCP port 80:

```
netstat -an | findstr :80
```

If another program such as Internet Information Services (IIS) is running and using the port, the ACME HTTP challenge can still be completed using a special reverse proxy rule that is not covered in this guide.

Install the Posh-ACME PowerShell module:

```
Install-Module -Name Posh-ACME -Scope AllUsers
```

The certificate common name has to match the domain controller FQDN. Since Let's Encrypt will need to resolve the same FQDN, do not forget to update your external DNS configuration accordingly. This means adding a DNS A record for "IT-HELP-DC" under "ad.it-help.ninja" that points to the domain controller public IP address. Make sure that the domain controller FQDN can be resolved before continuing:

```
New-PACertificate «IT-HELP-DC.ad.it-help.ninja» -Plugin WebSelfHost -AcceptTOS
```

Wait a few minutes for the challenge to complete. If it succeeds, then you should be able to get the last certificate object using the Get-PACertificate command, and find the path on disk for the files:

```
$Certificate = Get-PACertificate
$CertPath = Split-Path -Path $Certificate.PfxFullChain -Parent
```

If you can find the certificate files, congratulations! You now have a valid certificate usable for LDAPS.

Configuring LDAP Server Certificate

Did you think it was over already? Not so fast! We have a certificate, but the LDAP server needs to be configured to use it. This part is, unfortunately, a bit complicated unless you have the right code snippets available to do it, so keep a link to this blog post as a reference.

Let us begin by importing the server certificate into the local machine certificate store:

```
$CertificatePassword = $(ConvertTo-SecureString -AsPlainText 'poshacme' -Force)
$ImportedCertificate = Import-PfxCertificate -FilePath $Certificate.PfxFullChain `
-CertStoreLocation 'cert:\LocalMachine\My' -Password $CertificatePassword
$CertificateThumbprint = $ImportedCertificate.Thumbprint
```

The default password on certificates obtained using Posh-ACME is 'poshacme', but it can be changed. Certificate stores on Windows have a physical location inside the Windows registry. PowerShell offers a nice interface over some certificate stores, but not the one used by the LDAP server (NTDS). The trick is to import the certificate into a temporary store, then copy it to the destination store using the Windows registry paths:

```
$LocalCertStore = 'HKLM:/Software/Microsoft/SystemCertificates/My/Certificates'
   $NtdsCertStore = 'HKLM:/Software/Microsoft/Cryptography/Services/NTDS/
   SystemCertificates/My/Certificates'


if (-Not (Test-Path $NtdsCertStore)) {
 New-Item $NtdsCertStore -Force
}

Copy-Item -Path «$LocalCertStore/$CertificateThumbprint» -Destination $NtdsCertStore
```

The certificate thumbprint is the signature or hash of the certificate used as the name inside the registry store key structure. You do not need to know the details, other than that you need to find the right thumbprint to copy the right certificate. Run the following commands to tell the LDAP server to renew its server certificate configuration and apply the changes:

```
$dse = [adsi]'LDAP://localhost/rootDSE'
[void]$dse.Properties['renewServerCertificate'].Add(1)
$dse.CommitChanges()
```

At this point, the LDAP server should now properly respond to a TLS handshake over TCP port 636 (standard LDAPS port). Make sure that the firewall is properly configured, then test the TLS handshake using OpenSSL:

```
openssl s_client -connect IT-HELP-DC.ad.it-help.ninja:636 -showcerts
```

If it works, then OpenSSL should validate the certificate automatically, and show Let's Encrypt as the certificate authority. The chain should contain the Let's Encrypt intermediate CA, and the server certificate with the FQDN of your domain controller. If validation fails, take a closer look at the errors returned by OpenSSL.

Closing Thoughts

Considering the importance of Secure LDAP for the future of Active Directory, it is surprising to find out how difficult it is to properly configure the LDAP server to use a certificate. A lot of online guides use ldp.exe to test LDAP connectivity, but it has the unfortunate limitation of returning the same error when certificate validation fails as it does when the server is unreachable.

As if that was not enough, there is no way to select which certificate will be used by the LDAP server inside the NTDS certificate store. And so, the most foolproof way to do it is to delete all other certificates except the one that you wish to use. Let's Encrypt certificates are only valid for three months, and so it is recommended to create an automated script run to handle the renewal.

Please share this blog if you found it helpful, but more importantly, we would like to know more about your experience configuring Secure LDAP in your Active Directory environment. Please comment below!