Active Directory Hardening Series - Part 3 – Enforcing LDAP Signing

techcommunity.microsoft.com/blog/coreinfrastructureandsecurityblog/active-directory-hardening-series---part-3—-enforcing-ldap-signing/4066233

Blog Post

Hi all! Jerry Devore back again to continue talking about hardening Active Directory. If you have been following this series, I hope you have been able to enforce NTLMv2, remove SMBv1 from your domain controllers, and you are ready to tackle the next important topic which is enforcing LDAP signing. Preventing unsecure LDAP communication by enforcing signing is an issue that the security community feels strongly about, and much has already been written on the topic. However, there seems to be a

considerable amount of confusion and misunderstanding about the impact of enforcing LDAP signing. I hope to clear things up today and give you the information you need to move forward with confidence.

LDAP Signing vs Sealing

LDAP is used to read, write and modify Active Directory objects. If security settings have not been enabled on the LDAP client and LDAP server, that information will cross the network as clear text. As a result, Active Directory attributes and the credentials used to authenticate could be easily readable to an Adversary-in-the-Middle (AiTM). Additionally, the AiTM could modify packets, hijack existing sessions and perform relay activities. Those risks can be mitigated by using signing and sealing but what does that actually mean?

Signing and sealing is not unique to LDAP. In fact, it is not unique to computing. Its history can be traced back thousands of years to the practice of applying a wax seal to a folded or rolled letter using a stamping device. If the seal was not broken, then the recipient would know the letter had not been read or modified during the delivery process. Additionally, since the stamping device was unique, the recipient was certain of the sender's identity.

Given wax seals would wreak havoc on routers the process of signing had to be updated for network communication. Rather than distribute stamping devices, signing keys are exchanged in a secure manner. The method used to exchange keys and the strength of the keys varies based on the protocol and authentication method. As messages are sent, a hash of the message is created then that hash is signed with the signing key. The recipient can then use its copy of the signing key to verify the hash and confirm the message was not modified while in transit.

In network communication, sealing is achieved via encryption. As with signing keys, encryption keys are exchanged over an authenticated session. While the exchange of keys may rely on asymmetrical encryption, the process of message encryption\ decryption generally relies on symmetric keys which is much faster than asymmetrical encryption.

When it comes to LDAP, Microsoft has policies to enforce signing but you will not find a setting to require sealing because the LDAP RFCs do not provide a standard for enforcing it. Encryption of LDAP traffic is dependent on what is supported by the client application. When NTLM is used for a SASL bind, encryption is always enabled but with Kerberos sealing is dependent on the client using the session option **LDAP_OPT_ENCRYPT** (can change during the session). If you want to confirm a particular application is requesting sealing, you could use ETW tracing (preferred) or a network capture.

To be clear, when signing is used but sealing is not, the communication is protected from tampering (relay, modification) even though the messages are readable by anyone in the middle.

Binding types explained

Domain controllers support two types of authenticated binds for LDAP which are **Simple Binds** and **SASL** (Simple Authentication and Security Layer). Using a Simple Bind for LDAP is similar to using Basic Authentication with a web server. In both cases the client provides a username and password in order prove its identity and without some form of session security those passwords traverse the network in the clear along with the data. Additionally, there is no way to detect if those messages have been modified during transit. To address those issues TLS (Transport Layer Security) is used to create a secure channel for the session. Technically TLS does not perform signing but instead achieves integrity through encryption. As a result, domain controllers requiring LDAP signing will accept simple binds if used with TLS.

Similar to how Simple Binds parallel Basic Authentication, SASL can be compared to Integrated Windows Authentication. Rather than use a clear password, SASL leverages derived versions of the password (NTLM hash, MD5 digest), Kerberos tickets (RC4 or AES hashes), or client certificates to both authenticate the client and to exchange signing and encryption keys. Of those options, certificate authentication differs the most since the authentication happens outside of the LDAP header. That is why it is also referred to as External.

Configuring signing LDAP

Like other protocols, the negotiation of LDAP signing is accomplished by the client and server stating what they are willing to support. The session will then be established with the strongest security that can be mutually agreed upon. If the server requires signing but the client does not support signing, the session is disconnected by the server.

On a Windows LDAP client the signing is managed by the policy setting Network security: LDAP client signing requirements (for the Microsoft LDAP client runtime)

When the policy is set to **None**, the LDAP client will not offer to perform signing with SASL binds. If **Negotiate signing** is configured the LDAP client will offer to do signing with SASL binds but will proceed if the server does not agree. Configuring the policy to **Require signing** will cause the client to terminate the binding request if the LDAP server is not willing to perform signing. Modifying this policy does not require a reboot to be effective so you can toggle it off and on as you perform testing. When you change this policy setting the behavior is ultimately enforced by updating the client's registry (HKLM\System\CurrentControlSet\Services\LDAP\LdapClientIntegrity – 0=None, 1=Request Signing, 2=Require Signing).

The good news is that Windows has supported LDAP signing for SASL binds for a very long time (Windows XP/2003 and later) and the default client setting is Negotiate Signing (LdapClientIntegrity=1). As a result, enforcing LDAP signing on domain controllers will not break Windows clients when they use SASL authentication for LDAP binds (e.g.

Searching AD, applying GPO, etc.) unless the default setting has been overwritten. However, you may have non-windows devices integrated with your domain that are making SASL binds without requesting signing.

On a domain controller LDAP signing is managed using the policy setting Domain controller: LDAP signing requirements.

As you can see the policy has two possible settings. The **None** setting will configure the domain controller to negotiate signing but not require it if the client does not agree to signing. The **Require Signing** setting as the name implies configures the server to reject the bind if signing is not supported by the client. Managing this policy will update **HKLM\SYSTEM\CurrentControlSet\Services\NTDS\Parameters\LDAPServerIntegrity**. (None = 1, Require Signing = 2). If your domain controllers do not have this policy defined go ahead and enable it to set it to **None**. That will ensure all domain controllers are negotiating signing, but it will not break any clients that are not configured to support signing.

Configuring TLS for Simple Binds

As I mentioned before, making a LDAP simple bind without TLS will result in the password being sent over the network in clear text unless Layer 3 security (e.g. Ipsec) is used to encrypt the traffic. Additionally, the rest of the session will be in the clear, not signed and subject to AiTM exploits.

The standard way to implement TLS with Simple LDAP Binds is to configure your applications to use LDAPS which uses port 636 instead of 389. Another possibility is to leverage StartTLS which will use port 389 even after the TLS handshake. In either case it will be necessary to install a certificate on your domain controller. Once your domain controllers support TLS you will need to update your applications to start using LDAPS. Making that happen is going to require working with your applications owners. The configuration steps on the applications are going to vary so you will not be able to give your application owners “one size fits all” instructions on how to make that change.

Load balancing LDAP

Today many organizations use load balancer solutions to distribute LDAP connections across multiple domain controllers. While the use load balancers for LDAP is considered by some as best practice, in reality they often result in reduced security. Here are the concerns to keep in mind:

- If the load balancer is being used for TLS Offloading, the messages are transported over the network in the clear between the load balancer and the domain controller.
- If the load balancer is terminating the TLS session then starting a new TLS session between it and the domain controller, LDAP channel binding cannot be enforced on the domain controller.

- Kerberos will fail when using an alias or VIP name because the matching SPN cannot be added to multiple domain controller computer objects. The same is true when using a robin-round alias configuration without a load balancer. For more details checkout this article on why Active Directory and load balancers don't mix.

Often load balancers are used when an **application** only accepts one target name or IP address, and do not support fault tolerant concepts as implemented by DC Locator. The load balancer is then needed to make the new connection to a different domain controller if the current session fails. A load balancer is not needed when LDAP applications understand the DNS records used for dclocator logic. That approach achieves redundancy while not breaking Kerberos.

Auditing for unsigned LDAP sessions

Many organizations have been concerned about unsigned LDAP sessions but have been hesitant to enforce LDAP signing out of fear of breaking critical applications. Fortunately, there are auditing events you can use in lieu of a scream test. The first step is to spot check the Directory Service log of your domain controllers for **2887** events. Every 24 hours each domain controller will create that event to show the total number of unsigned LDAP sessions in the last day. This will give you an understanding of the volume of unsigned connections but will not tell you which device or account is making them.

To identify the source of the unsigned LDAP connections you need to enable LDAP Interface diagnostic logging. Many Windows diagnostics logging settings come with a boldly worded disclaimer about a performance impact. That is not a concern with this setting. However, it may cause a measurable increase in Directory Services events. The command below can be used to enable diagnostic logging in the registry of your domain controllers. To be comprehensive, the registry key should exist on all domain controllers in the domain.

reg add HKLM\SYSTEM\CurrentControlSet\Services\NTDS\Diagnostics /v "16 LDAP Interface Events" /t REG_DWORD /d 2

Once diagnostic logging has been enabled, each unsigned LDAP connection will trigger a **2889** event. As you can see from the following example, the event will capture the client's IP address and the name of the account used to authenticate. Additionally, the **Binding Type** field will tell you if the unsigned connection was made using SASL without signing or a simple bind without TLS (0 = SASL / 1 = 1 Simple bind). That is a very important data point given the remediation approach will vary based on the binding type.

Hopefully you have a SIEM that provides centralized logging and has rich query capabilities. If not, the following script can be used to extract the data from the Directory Service log of your domain controllers. In addition to the information in the events, the script will attempt to resolve the client's name (DNS reverse record) then perform a lookup the device in Active Directory and export out helpful attributes to like OS version and Distinguished Name.

```
# Get the 2889 events from the Directory Service log on the domain controller
$events = Get-WinEvent -FilterHashtable @{LogName='Directory Service';Id=2889}
# Create an empty array to store the output
$output = @()
# Loop through each event
foreach ($event in $events) {
    # Extract the client IP address without port number
    $clientIP = ($event.Properties[0].Value -split ':')[0]
    # Extract the user account name
    $userAccount = $event.Properties[1].Value
    # Extract the numerical binding type
    $numericalBindingType = $event.Properties[2].Value
    # Extract the timestamp of the event
    $timestamp = $event.TimeCreated
    # Perform a DNS reverse lookup of the client IP address
    try {
        $dnsName = [System.Net.Dns]::GetHostEntry($clientIP).HostName
    } catch {
        $dnsName = "Unresolved"
    }
    # Get additional computer object attributes using Get-ADComputer
    try {
        $computerObject = Get-ADComputer -Filter { DNSHostName -eq $dnsName } -
Properties OperatingSystem, DistinguishedName
    } catch {
        $computerObject = @{
            OperatingSystem = "Unknown"
```

```
            DistinguishedName = “Unknown”
        }
    }
    # Create a custom object with the output fields
    $obj = [PSCustomObject]@{
        ClientIP = $clientIP
        UserAccount = $userAccount
        BindingType = $numericalBindingType
        Timestamp = $timestamp
        DNSName = $dnsName
        OperatingSystem = $computerObject.OperatingSystem
        DistinguishedName = $computerObject.DistinguishedName
    }
    # Add the object to the output array
    $output += $obj
}
# Write the output to a .csv file in the c:\script folder
$output | Export-Csv -Path ‘c:\script\resolved_ldap_clients.csv’ -NoTypeInformation
Write-Host “Script completed. Output file created.”
```

MDI Reporting

If you have **Microsoft Defender for Identity** deployed, you can leverage a **Secure Score** report to identify which credentials have been exposed via Simple Binds without TLS. Currently MDI does not report unsigned SASL sessions.

Do’s and Don’ts for enforcing LDAP signing.

Hopefully this post helped clear up any confusion about LDAP signing and has given you confidence to start remediation in your environment. Just follow these rules and you should be good to go.

- **Don’t** configure your domain controllers to **Require Signing** until auditing shows there are no more unsigned binds in your environment.
- **Do** configure your LDAP clients to support signing. If your domain controllers are negotiating signing you can safely bump all your Windows devices to up to **Require Signing**.
- **Do** pay close attention to the **Binding Type** field in 2889 events. If you have any Binding Type = 0 events, then you have devices performing SASL authentication that are not negotiating signing. Binding Type = 1 means you have applications making simple binds without a TLS session. You might as well treat the two like separate projects given the remediation steps will be completely different.
- **Don’**t assume that enforcing LDAP signing is the same thing as forcing all LDAP traffic to use port 636 instead of 389. LDAP sessions with StartTLS and SASL binds with signing on port 389 are secure **as well**.
- **Don't** assume that SASL with signing is less secure than TLS. However, not all SASL authentication methods are equal. Strive to eliminate the use of weak protocols like NTLM where possible.
- **Do** understand that even some native Window clients do not request sealing. Examples of those are DS search (in the shell), Group Policy and Certificate Auto-Enrollment. Those clients will have integrity protection assuming the OS has been configured to request signing.
- **Don't** confuse LDAP signing with LDAP Channel Binding. Both are important security controls for securing LDAP, but they are managed separately. Of the two LDAP signing is easier to enforce, which is why I am covering it first.
- **Do** know the client IP address in 2889 events will be the address of the load balancer if it is forwarding unsecure LDAP traffic to the domain controller. In those cases, you will need to work with the load balancer team to resolve the issue.
- **Do** monitor for 2888 events after you have enforced signing on your domain controllers. Those events will inform you if an LDAP client attempted to make an unsigned LDAP bind but was rejected. Like a 2887 event, a 2888 will show volume but does not log the device making the unsecure bind request.
- **Do** share your lessons learned from enabling LDAP signing in the comments below.
- **Do** check out these other resources on securing LDAP:

Disclaimer

The sample scripts are not supported under any Microsoft standard support program or service. The sample scripts are provided AS IS without warranty of any kind. Microsoft further disclaims all implied warranties including, without limitation, any implied warranties of merchantability or of fitness for a particular purpose. The entire risk arising out of the use or performance of the sample scripts and documentation remains with you. In no event shall Microsoft, its authors, or anyone else involved in the creation, production, or delivery of the scripts be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use the sample scripts or documentation, even if Microsoft has been advised of the possibility of such damages.

Updated Nov 08, 2024

Version 6.0

\n \n\n \n\n \n\n \n\n \n\n\n\n \n\n\n\n \n\n \n\n\n\n \n\n \n\n \n\n\n\n \n\n \n\n \n\n \n \n\n \n\n \n\n\n\n \n\n \n\n \n\n\n\n \n\n \n \n\n \n \n\n \n \n\n \n \n\n \n \n\n \n \n\n \n \n\n \n \n \n \n \n\n \n \n \n \n \n \n \n \n \n\n \n \n \n \n \n \n \n \n \n\n \n \n \n\n \n \n \n\n \n\n \n\n\n\n\n \n\n \n\n \n \n\n \n

\n \n\n \n\n \n\n \n\n \n\n\n\n \n\n\n\n \n\n \n\n\n\n \n\n \n\n \n\n\n\n \n\n \n \n\n \n \n\n \n\n \n\n\n\n \n\n \n\n \n\n\n\n \n\n \n \n\n \n \n\n