Devolutions

Why Active Directory LDAP Unauthenticated Binds Should Be Disabled, and How to Do It

WINDOWS ACTIVE DIRECTORY IS HERE TO STAY

Windows Active Directory has been the undisputed king of corporate network environments for decades, and while Azure Active Directory is slowly taking its place, Windows Active Directory is here to stay. However, just like most products that have been around for a long time, Active Directory has obscure legacy features that look like they were put in a time capsule. In many ways, Active Directory is like an old building that is no longer up to code, making it a fire hazard to those that neglect to harden it properly. For example: LDAP unauthenticated binds are enabled by default in Windows Server 2019, but you should consider disabling them. (By the way, if you don't want the long explanation, you can skip directly to the end of the article, which contains the PowerShell code snippet.)

LDAP in Active Directory

Active Directory is comprised of multiple services, but the primary component is the Lightweight Directory Access Protocol (LDAP) server. This contains information about everything inside the domain (e.g., users, user groups, machines, devices, etc.). When logging in to a Windows domain, part of the authentication process involves sending an LDAP bind request to the domain controller to validate the credentials. It is common for third-party applications to delegate authentication to Active Directory using LDAP.

LDAP Authentication

Typical Windows applications use built-in functions to validate credentials using NTLM or Kerberos with LDAP, or Secure LDAP (LDAPS) if it has been configured. Third-party applications with limited support for NTLM or Kerberos may choose to send the full credentials using the LDAP simple bind type instead. It is similar to HTTP basic authentication, but it is acceptable in LDAPS because of the protection provided by Transport Layer Security (TLS).

Anonymous or Unauthenticated?

The LDAP simple bind has a few tricks up its sleeve: it is possible to use an empty username and password to "authenticate" as an anonymous user. The legitimate use case for this is LDAP configuration discovery: anyone can fetch the same information returned by the Get-ADRootDSE PowerShell command from the LDAP server. LDAP anonymous authentication is when both the username and password are empty strings. LDAP unauthenticated authentication is when the username is non-empty, with an empty password. While both use cases are often confused, the LDAP specification makes **anonymous authentication mandatory** and **unauthenticated authentication optional**, with a recommendation to disable it by default.

Unintended Consequences

If both use cases look almost identical, why is unauthenticated authentication such a big deal? Because it is so unexpected and poorly understood, many developers do not expect LDAP bind requests to succeed with an empty password. Most applications delegating authentication to LDAP expect the LDAP bind to succeed only when given the correct password, which is not the case when the password is empty. True anonymous

authentication uses an empty username, which is likely caught by various checks early on. But do all developers think about checking for empty passwords? Absolutely not, and this has caused multiple security vulnerabilities in the past.

Preventing the Problem

While it is preferable that applications using LDAP authentication explicitly check for empty passwords, it is possible to disable LDAP unauthenticated binds starting from Windows Server 2019. The following PowerShell code snippet is sufficient to make the necessary modification:

```
$RootDSE = Get-ADRootDSE
$ObjectPath = 'CN=Directory Service,CN=Windows NT,CN=Services,{0}' -f $RootDSE.
ConfigurationNamingContext
Set-ADObject -Identity $ObjectPath -Add @{ 'msDS-Other-Settings' =
'DenyUnauthenticatedBind=1' }
```

Closing Thoughts

While the LDAP unauthenticated bind is not a security issue by itself, it is the kind of feature that leads too easily to unintended consequences. Security researchers should always try using empty passwords in applications that perform LDAP authentication in hope of finding a possible authentication bypass vulnerability. This is a perfect example of something relatively easy to fix — but only if you know that it exists in the first place.