

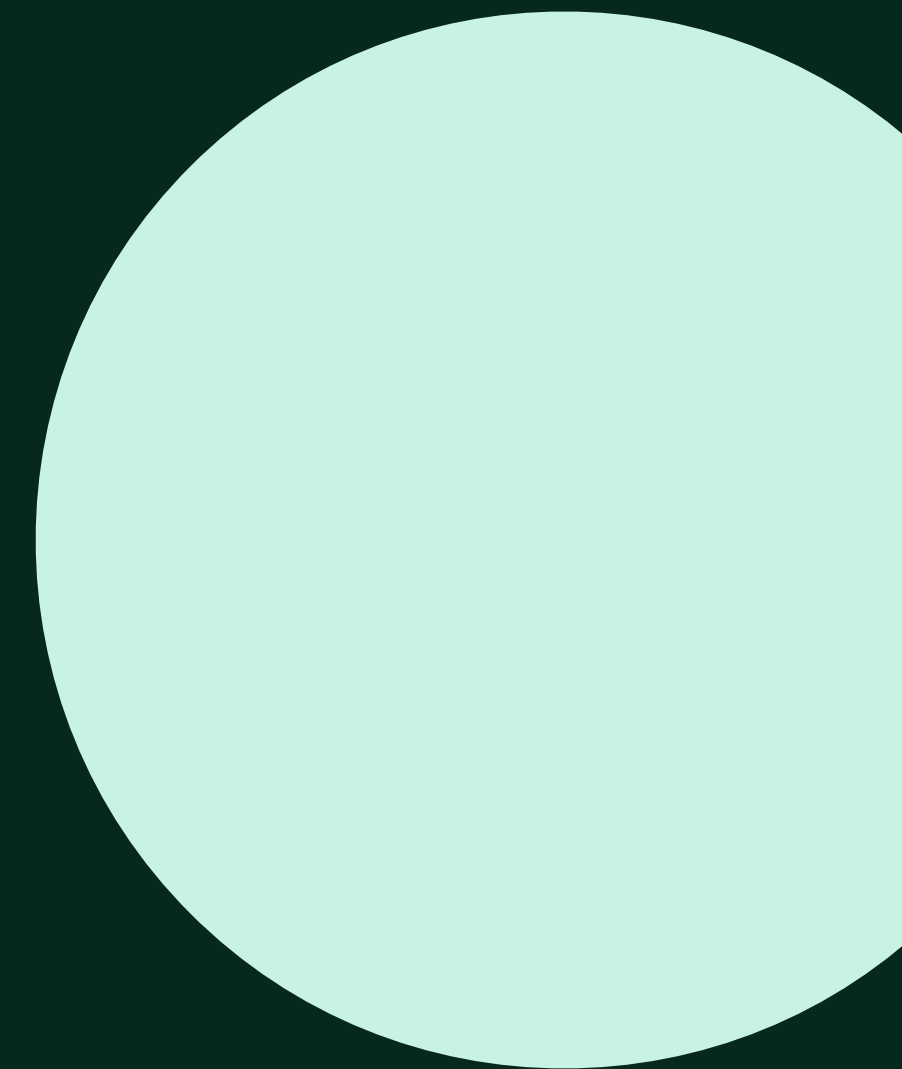
A WithSecure™ Consulting whitepaper



# Keeping the attackers out

**Golden tickets, silver tickets,  
and full domain recovery**

Author: Johann Scheepers  
Incident Response Consultant



Active Directory (AD)-based attacks that lead to the compromise of domain controllers (DCs) and the use of golden tickets have been covered extensively by different authors. So why write another article about them?

During the recovery phase of an incident, focus is primarily placed around the possibility of golden ticket attacks and the associated remediation steps. This isn't the only way to exploit Kerberos functionality, however; silver ticket attacks, though less spoken of, are equally important in the context of a domain

controller compromise. Secondly, it's often the case that teams may not have considered the true impact of a domain controller compromise or have a recovery plan in place following a Kerberos-based attack, which leads to greater damage being caused to their organization. This paper covers the risks

associated with silver ticket attacks and explains the post-exploitation impact of Kerberos-based attacks, from an incident response perspective. It will also provide recommendations on how to manage the domain recovery phase effectively.

## Kerberos authentication: A quick primer

Note: The below section is a condensed representation of what happens behind the scenes when a user authenticates and requests access to a service. The full Kerberos protocol and its implementation in Windows is outside the scope of this article.

For many Active Directory administrators, the KRBTGT account is merely an account that they have never consciously interacted with. However, the KRBTGT account plays a critical role in the operation of Active Directory's underlying authentication processes. The account's password hash is used to encrypt, and subsequently verify, each Kerberos Ticket-Granting-Ticket (TGT) issued by Active Directory. As such, being in possession of the KRBTGT account password hash would grant you the ability to create valid Kerberos TGTs.

In order to appreciate the concepts covered in this article, it is necessary to understand how Kerberos tickets are issued.

The process is summarized in the diagram below:

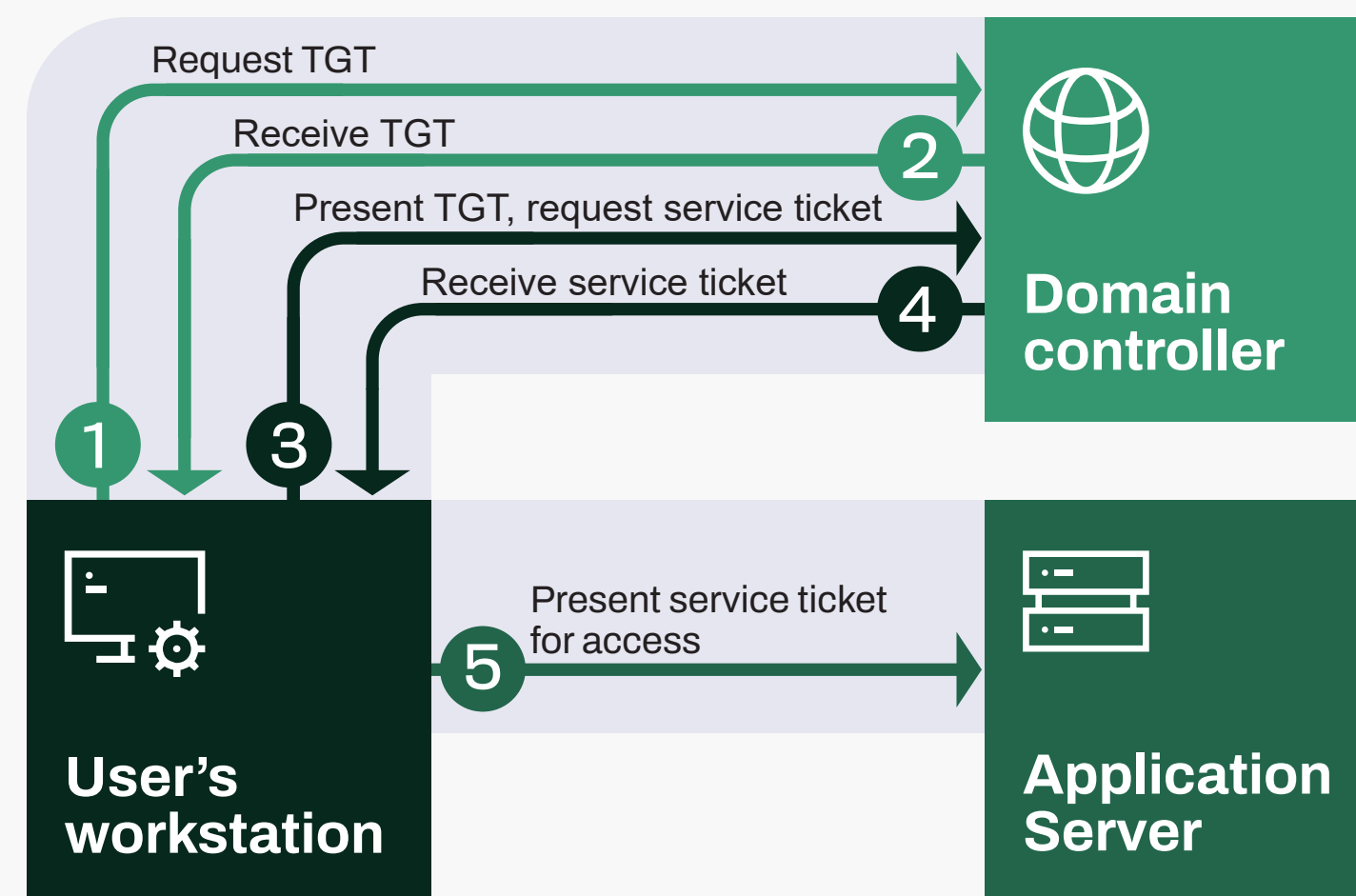


Fig. 1. The Kerberos authentication process

1. A user performs initial authentication on their workstation. This action sends a request to the Key Distribution Centre (KDC) service on a domain controller.
2. The KDC service gathers all information tied to the user's account and returns a TGT. This TGT is itself signed and encrypted with the password hash of the KRBTGT account.
3. The user decides they would like to access a document hosted on an internal application server. A request is sent to the domain controller, along with the TGT, requesting a Ticket-Granting-Service (TGS) ticket (otherwise called "service ticket"). This causes the KDC service to validate the original TGT, which it can do since it originally encrypted and signed it using the KRBTGT account password hash.
4. Once validated, a service ticket is issued by encrypting and signing the ticket with the account password hash of the Service Principal Name (SPN) associated with the requested service. It returns a service ticket validating the user's permissions.
5. The user contacts the file server and presents the service ticket. The server validates the ticket locally, which it can do since it has knowledge of the service account password hash. The user gains access to the service.

# The KRBTGT account and golden tickets

Let's assume for a moment that one of your domain controllers had been accessed by an attacker. They logged on with an account belonging to the Domain Admins group, and as a result, managed to extract the Active Directory database. The most common ways of going about this task would be either creating a copy of the NTDS.DIT file with the ntdsutil utility (present on domain controllers) or using MimiKatz to extract all accounts and their associated password hashes. Don't forget that your domain controllers are likely being backed up to a storage repository that the attacker may have gained access to as well.

With the attacker having gained this information, it would allow them to generate an authentication ticket with any set of privileges, while impersonating any user account (existing or not). This token would be valid throughout the Active Directory forest for an indefinite amount of time, or at least until steps have been taken to prevent its use. An attacker with the ability to forge authentication tickets would possess the ability to connect via Kerberos-enabled services, such as Windows Management Instrumentation (WMI), Windows Remote Management (WinRM), and Server Message Block (SMB) to name a few.

To illustrate this concept, the screenshot shows authentication against a domain controller as a non-existent user with administrative privileges, made possible by crafting a golden ticket:

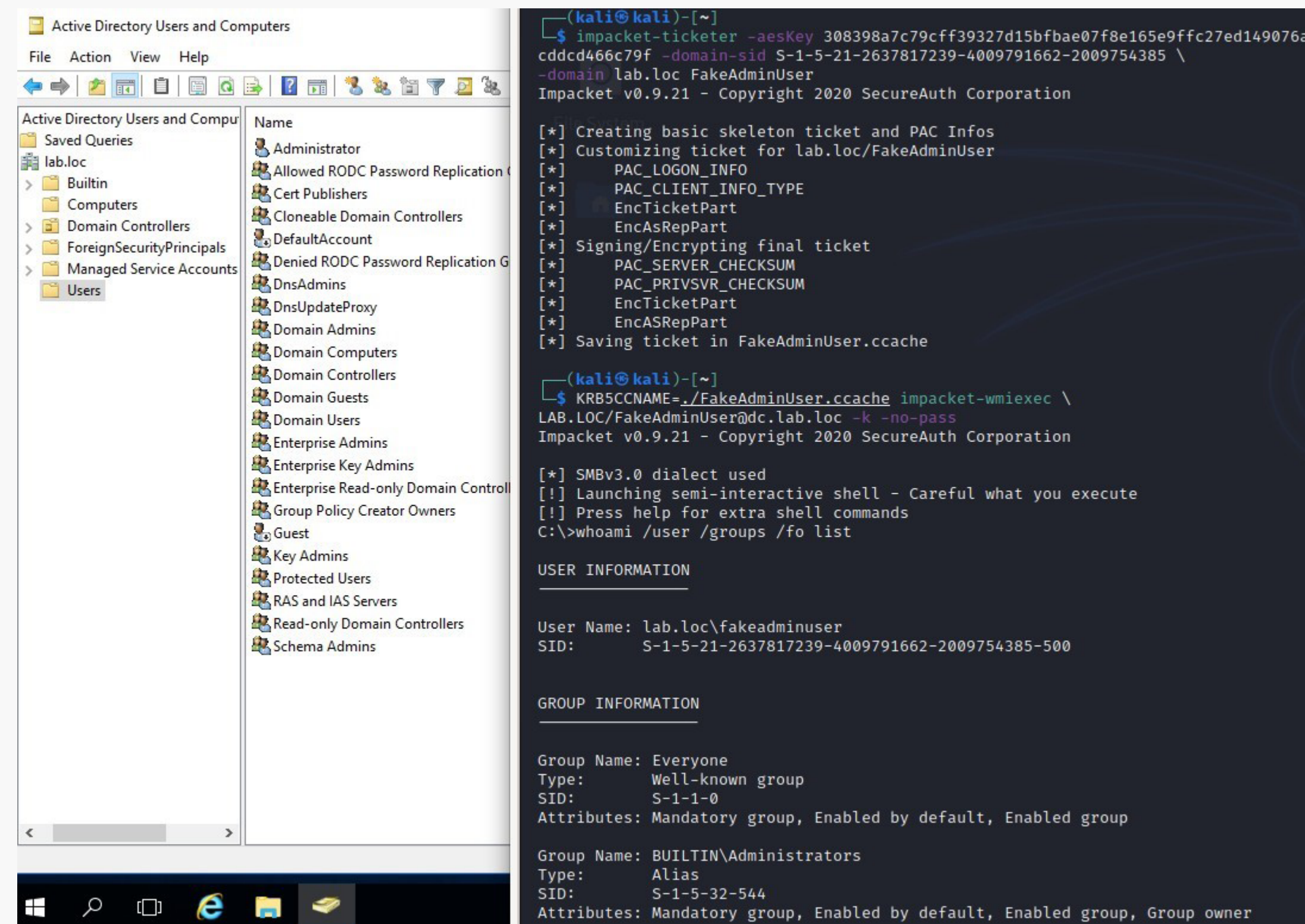


Fig. 2. Authentication against a domain controller using a golden ticket

## So, how do you detect the existence of golden tickets?

The inherent issue with golden tickets is that they are technically valid Kerberos tickets, appearing as if issued by a domain controller. This tactic is known as “exploitation of legitimate functionality”. It is difficult to differentiate between a ticket issued to a legitimate user and one forged via a stolen KRBTGT hash, as the differences generally come down to abnormalities specific to your operating environment.

Such examples could include:

- The source IP address (do administrators normally connect from this IP subnet?)
- The account name in the ticket (does the account name match the security identifier?)

Another way of determining validity would be to scrutinize the security event logs for the issuance of a TGT (4768), followed by a service ticket (4769), and finally a logon event (4624). This is what the expected process flow would look like for a user initially authenticating and gaining access to a service:




Keywords	Date and Time	Source	Event ID	Task Category
 Audit Success	2020/05/24 6:26:32 PM	Microsoft Windows security auditing.	4624	Logon
 Audit Success	2020/05/24 6:26:32 PM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations
 Audit Success	2020/05/24 6:26:32 PM	Microsoft Windows security auditing.	4768	Kerberos Authentication Service

Fig. 3. Windows security logs showing a user authenticating and gaining access to a service

However, if the ticket source is generated by the compromised KRBTGT hash, one would only see a service ticket issued (4769), followed by a logon event (4624), since the TGT token was already generated offline by the attacker.



Keywords	Date and Time	Source	Event ID	Task Category
 Audit Success	2020/05/24 6:57:17 PM	Microsoft Windows security auditing.	4624	Logon
 Audit Success	2020/05/24 6:57:17 PM	Microsoft Windows security auditing.	4769	Kerberos Service Ticket Operations

Fig. 4. Windows security logs showing the same process using a previously generated TGT

There is further nuance, as Kerberos tickets are usually valid for 10 hours by default, or 4 hours when an account is part of the Protected Users<sup>1</sup> group. Therefore, if a legitimate TGT had been issued between 4 and 10 hours beforehand, one would still only see service ticket (4769) and logon (4624) events. As such, the detection of golden tickets used to impersonate legitimate user activity during business hours would be near impossible, unless the user’s activity is laboriously scrutinized. Tracking TGT, service tickets, and logon events would therefore require more complex analytics in order to find anomalies.

1 <https://docs.microsoft.com/en-us/windows-server/security/credentials-protection-and-management/protected-users-security-group>

## What steps could then be taken to remediate golden ticket use in your environment?

In cases where golden ticket abuse is suspected, the only effective solution is to invalidate the KRBTGT account hash, by changing the password. Twice.

One might rightfully suspect that this would be a critical operation with the potential to cause service disruptions, in that cycling this password would invalidate all existing Kerberos tickets. This would include tickets for all users with active sessions and service accounts. Practically any user or service that makes use of Kerberos authentication within Active Directory could be affected by this operation. This could cause authentication failures while accessing services, leading to service disruption. One such example is the Remote Desktop service, where processing of logon sessions would fail if users attempted to authenticate with pre-existing TGTs.

While the default password policy in Active Directory is set to maintain a history of 24 iterations on regular accounts to prevent password re-use, at least one iteration is also kept for computer accounts. However, unlike regular accounts, this password history intends for authentication to still succeed if a computer account password is updated and replication delayed throughout the domain for any reason.

In such cases, the domain controller will also attempt to authenticate with the previous hash, but nothing prior.

The above is true for TGT tickets and the KRBTGT account too, as the KDC service will attempt to validate an issued TGT using the previous KRBTGT password hash. As such, in order to prevent an attacker from using golden tickets to maintain persistence, the KRBTGT account's password needs to be updated twice in succession to invalidate tickets based on the stolen credentials. The KRBTGT account password can be set manually. In fact, it does not matter what you set it to, as Active Directory will automatically change it to a lengthy, randomized string, as per KB2549833<sup>2</sup>. However, the challenge then is to wait for the change to synchronize to all domain controllers forming part of the domain or forest before setting it a 2nd time.

If sufficient time for replication is not given, users and service accounts may run into an issue where a TGT is issued by domain controller A, but a service ticket is requested from domain controller B. This would result in failure, as domain controller B would not be able to decrypt and validate the TGT provided by domain controller A. The only solution in such an event would be to wait for replication to complete in full. This manual approach is not a reliable remediation in an under-attack scenario, because an attacker could still create service tickets against any domain controllers that had not received the KRBTGT changes. Performing resets with active attackers on your network is covered in the following section.

## A better way

A PowerShell script<sup>3</sup> was published on Microsoft's GitHub account that provides a more controlled way to perform KRBTGT password resets. It allows for checking whether all domain controllers are reachable, and once the password has been reset, performs a single object replication. This is followed by a check to confirm that all domain controllers have in fact replicated the change.

The use of this script allows for quick updates in an under-attack scenario, typically within seconds or minutes, while informing you of the potential impact related to offline or unreachable domain controllers. The script provides multiple modes of execution, such as informational mode, simulation mode, and reset mode. Simulation mode allows you to perform an effective 'dry run' of the reset process, whereby it checks for domain controller availability and triggers a single object replication without making any changes to your environment. It is worth noting that a domain functional level (DFL) of 'Windows Server 2008' or higher is required.

A screenshot of the script being executed in our lab environment is included on the next page.

<sup>2</sup> <https://docs.microsoft.com/en-za/troubleshoot/windows/win32/change-krbtgt-password-may-fail>

<sup>3</sup> <https://github.com/microsoft/New-KrbtgtKeys.ps1>

```

+++ RESET KRBTGT ACCOUNT PASSWORD FOR RWDCs/RODCs +++
[2020-11-25 14:59:26] : --> RWDC To Reset Password On.....: 'DC.lab.loc'
[2020-11-25 14:59:26] : --> sAMAccountName Of KrbTgt Account.....: 'krbtgt'
[2020-11-25 14:59:26] : --> Distinguished Name Of KrbTgt Account...: 'CN=krbtgt,CN=Users,DC=lab,DC=lab.loc'
[2020-11-25 14:59:26] : --> Number Of Chars For Pwd Generation....: '64'
[2020-11-25 14:59:26] : --> Previous Password Set Date/Time.....: '2020-11-25 11:57:04'
[2020-11-25 14:59:26] : --> New Password Set Date/Time.....: '2020-11-25 14:59:26'
[2020-11-25 14:59:26] : --> Previous Originating RWDC.....: 'DC.lab.loc'
[2020-11-25 14:59:26] : --> New Originating RWDC.....: 'DC.lab.loc'
[2020-11-25 14:59:26] : --> Previous Originating Time.....: '2020-11-25 11:57:04'
[2020-11-25 14:59:26] : --> New Originating Time.....: '2020-11-25 14:59:26'
[2020-11-25 14:59:26] : --> Previous Version Of Attribute Value...: '2'
[2020-11-25 14:59:26] : --> New Version Of Attribute Value.....: '3'
[2020-11-25 14:59:26] : --> The new password for [CN=krbtgt,CN=Users,DC=lab,DC=lab.loc] HAS BEEN SET on
RWDC [DC.lab.loc]!...
[2020-11-25 14:59:26] :
[2020-11-25 14:59:26] : ===== CHECK 1 =====
[2020-11-25 14:59:26] :
[2020-11-25 14:59:27] : - Contacting DC in AD domain ...[DC.LAB.LOC]...(SOURCE RWDC)
[2020-11-25 14:59:27] :   * DC is Reachable...
[2020-11-25 14:59:27] :   * The new password for object [CN=krbtgt,CN=Users,DC=lab,DC=lab.loc] exists
in the AD database
[2020-11-25 14:59:27] :
[2020-11-25 14:59:27] : --> Start Time.....: 2020-11-25 14:59:26
[2020-11-25 14:59:27] : --> End Time.....: 2020-11-25 14:59:27
[2020-11-25 14:59:27] : --> Duration.....: 0,08 Seconds
[2020-11-25 14:59:27] :
[2020-11-25 14:59:27] : List Of DCs In AD Domain 'lab.loc' And Their Timing...
[2020-11-25 14:59:27] :
[2020-11-25 14:59:27] :
Host Name      PDC Site Name      DS Type      IP Address      Reachable Source RWDC FQDN Time
-----
DC.lab.loc    True Default-First-Site-Name Read/Write 192.168.135.200      True N.A.          0

[2020-11-25 14:59:27] :
[2020-11-25 14:59:27] :
[2020-11-25 14:59:27] : Log File Path...: C:\2020-11-25_14.58.40_DC_Reset-KrbTgt-Password-For-RWDCs-And-RODCs.log
[2020-11-25 14:59:27] :
PS C:\>

```

Fig. 5. Completion of the KRBTGT reset process using Microsoft's script

Following the process of resetting the KRBTGT account password twice in succession, Kerberos ticket authentication failures should be handled gracefully in most cases. However, there is a likelihood that service accounts and their related services may fail to authenticate due to not automatically requesting a new TGT once the tickets have been rejected by domain controllers. Users may also need to request new TGTs by logging out of their workstations and re-authenticating if this process was followed throughout business hours.

As such, if your requirement to reset the KRBTGT account's password is one not relating to duress, it is recommended to wait at least 10 hours between reset operations. This will allow for any existing TGTs and service tickets to continue functioning, gracefully expire, and be re-issued between resets. After 10 hours, the KRBTGT password can be reset a second time without the concern of affecting existing TGTs.

# What about silver tickets?

Silver tickets should arguably be of more concern to organizations, though the path to compromise is quite different. Whereas golden tickets are forged TGTs, silver tickets are forged service tickets. A service ticket is the type of Kerberos token generated when requesting access to a service, such as when accessing a file share or applications, as illustrated in steps 3 through 5 below.

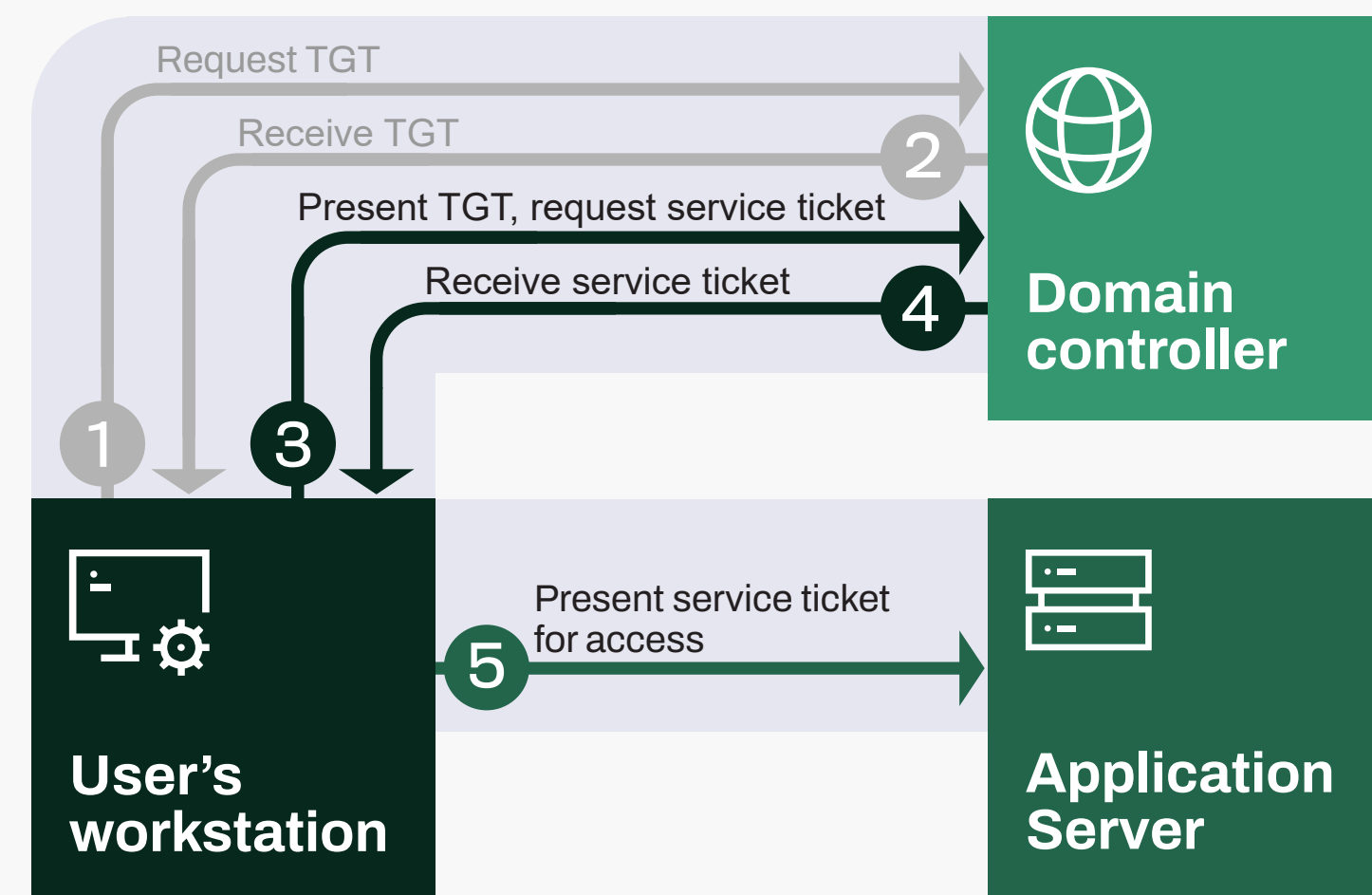


Fig. 6. The process of issuing a (TGS) service ticket

As with golden tickets, silver tickets require a service account's password hash in order to generate a ticket valid for a service. However, the compromise of a domain controller is not required, as explained here.

Each Kerberos-enabled service has an associated Service Principal Name. This is to allow Active Directory to link a service account to the service itself and is required to grant a user access to its resources. While service accounts and their related password hashes are only stored in the Active Directory database, there is an exception: the host SPN.

Host SPNs are generated automatically when a device is joined to an Active Directory domain, and as such, would exist for all endpoints. However, the interesting part is that the host SPN's associated password is in fact the computer account password. Due to how mutual authentication between hosts and domain controllers work, the host's computer account hash is stored both in the Active Directory database and on the host itself within the Security Accounts Manager (SAM) database. Therefore, if an attacker manages to compromise and elevate privileges to Administrator or SYSTEM on a host, they are in a position to retrieve the password hash. This would allow them to forge service tickets to maintain access to the host.

## So, what does a silver ticket allow one to do?

A machine's computer account affords it access to SYSTEM-level privileges, therefore a host SPN silver ticket can grant you the same privileges. An attacker could take advantage of this by retaining the information gathered during their initial path to compromise of the domain. It is reasonable to assume that having moved laterally to a host, they would perform credential-theft attacks on it. The credentials retrieved would also provide the computer account password hash.

It is important to note that if a domain controller is compromised, and the attacker manages to exfiltrate the Active Directory database, they would be in possession of the machine account password hash for every host on the network. This would then allow them to forge service tickets valid for any system and, in turn, regain SYSTEM-level access at any point.

An attacker regaining access to a host that domain administrators authenticate against could allow them to maintain persistence, dump hashes, gather cached credentials, keylog the host, or grab active Kerberos tickets for reuse in their quest to compromise the domain.

On the right is an example of the Impacket<sup>4</sup> toolkit being used to authenticate against a host using a service ticket generated from the computer account password hash:

<sup>4</sup> <https://github.com/SecureAuthCorp/impacket>

```
(kali㉿kali)-[~]
└─$ impacket-ticketer -nthash 2f23deacb95a79564bf382ecfc85eaca -domain-sid S-1-5-21-2637817239-4009791662-2009754385 -domain lab.loc -spn host/webapp.lab.loc FakeServerAdmin
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for lab.loc/FakeServerAdmin
[*]   PAC_LOGON_INFO
[*]   PAC_CLIENT_INFO_TYPE
[*]   EncTicketPart
[*]   EncTGSRepPart
[*] Signing/Encrypting final ticket
[*]   PAC_SERVER_CHECKSUM
[*]   PAC_PRIVSVR_CHECKSUM
[*]   EncTicketPart
[*]   EncTGSRepPart
[*] Saving ticket in FakeServerAdmin.ccache

(kali㉿kali)-[~]
└─$ KRB5CCNAME=./FakeServerAdmin.ccache impacket-wmiexec LAB.LOC/FakeServerAdmin@webapp.lab.loc -k -no-pass
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami /user /groups /fo list

USER INFORMATION
-----

User Name: lab.loc\fakeserveradmin
SID:      S-1-5-21-2637817239-4009791662-2009754385-500

GROUP INFORMATION
-----

Group Name: Everyone
Type:      Well-known group
SID:      S-1-1-0
Attributes: Mandatory group, Enabled by default, Enabled group

Group Name: BUILTIN\Users
Type:      Alias
SID:      S-1-5-32-545
Attributes: Mandatory group, Enabled by default, Enabled group

Group Name: BUILTIN\Administrators
Type:      Alias
SID:      S-1-5-32-544
```

Fig. 7. Authentication against a host using a forged silver ticket



## How do you detect the use of silver tickets?

As with golden tickets, it's not simple to detect the use of silver ticket usage. In fact, it's more difficult, as no authentication event will be logged against domain controllers. The service ticket Kerberos diagram in the previous section shows that the normal authentication process follows a user presenting a TGT to a domain controller to receive a service ticket. However, if you are able to craft that service ticket yourself, there is no need to contact a domain controller for this operation. As such, authentication events on individual hosts would need to be scrutinized to detect any anomalies, such as logon events with unexpected originating IP addresses.

Faced with such a prohibitive challenge to detect the use of silver tickets, its prevention and a well-planned response to the worst-case scenario are even more crucial.

## What steps can be taken to mitigate the use of silver tickets in your environment?

Not unlike mitigating golden ticket use, where the recommendation is to roll the KRBTGT account password, silver ticket mitigation also requires the computer account password to be changed twice.

Computer account passwords, by default, are updated approximately every 30 days when in a domain-joined state.

However, this behavior is initiated from the client and requires an active connection to a domain controller to succeed. A history of the computer account password is also maintained in the AD database (and on the computer in question) for 1 iteration to avoid situations where the replication process is delayed across the domain. This is to maintain mutual authentication between domain controllers and endpoints.

An important caveat to be aware of is that an attacker that has compromised a host could modify the registry key related to the automatic computer account password renewals. As such, it's important to monitor this registry value for changes:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
Services\Netlogon\ Parameters\DisablePasswordChange
```

Changing the value to '1' will prevent the client machine from contacting a domain controller to perform regular, automated machine account updates. In theory, alteration of this key could result in access to the target host indefinitely, assuming the machine account password is not updated manually. Monitoring for alterations to this key would be useful as an indicator of compromise.

## Manually updating a computer account password

Naturally, if one finds themselves in a post-compromise scenario that requires immediate computer account password changes, relying on the scheduled 30-day check-ins won't do. Luckily, there is a PowerShell command that allows this process to be initiated manually. The `Reset-ComputerMachinePassword`<sup>5</sup> command can also be invoked on a remote host, allowing it to be run as part of a script in order to target a range of hosts.

The below method is only valid for computers that are online and reachable during the process, which can lead to complications throughout a recovery scenario. It is therefore necessary to keep track of any hosts whose password reset attempts are not successful, so they can be attempted at a later time, in order to reach 100% coverage.

```
PS C:\Users\Administrator> $cred = Get-Credential
cmdlet Get-Credential at command pipeline position 1
Supply values for the following parameters:
Credential
PS C:\Users\Administrator> Invoke-Command -ComputerName "webapp" -ScriptBlock {Reset-ComputerMachinePassword -Credential $using:cred}
PS C:\Users\Administrator>
```

Fig. 8. Running the `Reset-ComputerMachinePassword` command

<sup>5</sup> <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/reset-computermachinepassword?view=powershell-5.1>

<sup>6</sup> <https://adsecurity.org/?p=1729>

<sup>7</sup> <https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/manage/ad-forest-recovery-reset-computer-account-dc>

## Silver tickets and domain controllers

During a typical domain controller compromise, an attacker will look at creating a copy of the domain controller's Active Directory database (`ntds.dit`), along with the `SECURITY` and `SYSTEM` registry hives required to export the data. The same information could be gathered by performing other AD attacks, such as a `DCSync`<sup>6</sup> attack. This attack initiates the Active Directory database replication process, allowing for direct access to its contents. Whichever path is chosen, the attacker would be in possession of two important hashes:

- The `KRBTGT` password hash
- The domain controller's machine account password hash

The latter would allow them to forge silver tickets for the domain controller's computer account, granting them `SYSTEM`-level access to the host. Therefore, as part of recovering from a domain controller compromise—in addition to rolling the `KRBTGT` account's password twice—the computer account password should be reset twice too.

## How does the computer account password reset process work for domain controllers?

This process should not be performed with the `Reset-ComputerMachine Password` mentioned previously. Domain controllers rely on knowing other domain controller machine account passwords. This enables them to establish the mutual authentication required when performing domain-controller-related activities, such as replicating database updates. If this process is interrupted, they can fall out of sync, with neither domain controller wanting to accept replication changes. Recovering from this state can be a complicated process, especially in larger domains with multiple domain controllers. Microsoft recommends<sup>7</sup> using the `netdom.exe` command to perform domain controller computer account password resets, as this will commit the updated password to both the local and a remote domain controller. A replication process is also initiated to make sure the updates are synchronized across the domain as quickly as possible. The `netdom.exe` command must be run locally on the domain controller whose password needs updating.

# Recovering from a domain controller compromise

It is important to note that the information covered below is not an all-encompassing guide to recovering from a domain compromise. A full incident response plan must still be followed in order to locate and remove all traces of the attacker responsible and their malware. This can include but is not limited to:

- Locating the infection vector of the incident to determine the root cause
- Blocking command and control (C2) domains and IP addresses used by the attacker to prevent communications with their own infrastructure
- Finding and removing all malware and payloads used during the attack to remove the attacker's foothold
- Patching all vulnerable systems to prevent exploitation of systems through publicly available exploits

Only once the attacker has been fully removed from your systems should the below steps be followed. This is to ensure that all avenues for their resurgence have been addressed.

Performing the steps whilst they maintain a foothold could alert them to their presence being known, providing them with opportunity to nullify any eradication or remediation efforts.

The steps should preferably be performed in the order listed below. It is also recommended you create separate, dedicated accounts for performing the operations. This will prevent users' accounts having credentials cached on compromised servers, potentially leading to re-compromise. Monitoring the activities of the dedicated accounts also allows for quick detection of unauthorized or unexpected activity.

1. Reset the password for every known compromised account and each account holding administrative privileges over domain controllers
2. Reset domain controller machine account passwords to prevent silver ticket abuse.  
Run the netdom.exe command twice on each domain controller
3. Reset the KRBTGT accounts to prevent golden ticket abuse. Use the script<sup>8</sup> provided by Microsoft to perform two reset operations

4. Reset the computer account passwords for computer objects joined to the domain to prevent silver ticket abuse. Use the Reset-ComputerMachinePassword PowerShell command to perform reset operations for all computer objects on the domain. This operation needs to be completed twice for each computer

5. Reset the password for every other account on the domain

Without sufficient logs and suitable endpoint detection and response tools, it might not be possible to determine exactly what an attacker had access to following a domain controller compromise. In addition, if an account with sufficient privileges is compromised, that attacker would not necessarily need to gain an interactive logon session on a domain controller to access information contained in the Active Directory database, as is the case with DCSync<sup>9</sup> and DCShadow<sup>10</sup> attacks.

It is recommended for this very reason that you execute the above recovery steps following the compromise of an account that had domain administrator privileges.

8 <https://github.com/microsoft/New-KrbtgtKeys.ps1>

9 <https://attack.mitre.org/techniques/T1003/006/>

10 <https://attack.mitre.org/techniques/T1207/>

## Some final thoughts...

By this point, you hopefully recognize that recovering from an Active Directory compromise requires plenty of planning and forethought.

An attacker, having compromised a domain controller or an account with domain administrator privileges, could possess information that would allow them to regain access to your infrastructure by abusing both golden and silver tickets. Implementing the mitigating steps covered in this article is a means to avoid this.

In our experience, from assisting clients in domain recovery scenarios, the recovery steps have generally required a coordinated execution effort across multiple teams, countries, and time zones. For many, it was their first attempt at implementing the proposed remediation steps. This can of course be challenging, but with guidance and a measured approach it was successfully executed each time. As daunting as these processes sound, they are possible, and will reduce the chances of a resurgence following an incident.

Thinking outside the box may also be necessary at times. In an example from one engagement, an unreliable link to a domain controller located in another continent complicated coordinating the reset of the KRBTGT account. This was handled by way of demoting the domain controller and re-promoting it once the remediation steps were implemented.

The efficiency with which Microsoft's script allows for testing the replication and availability of domain controllers allows the KRBTGT account to be reset as part of disaster recovery testing operations. We recommend that you regularly test this functionality to become familiar with the implementation thereof, and to locate and remediate potential blockers in the event that it should be executed as part of a full domain recovery plan.

# Who We Are

WithSecure™ is cyber security's reliable partner. IT service providers, MSSPs and businesses along with the largest financial institutions, manufacturers, and thousands of the world's most advanced communications and technology providers trust us for outcome-based cyber security that protects and enables their operations. Our AI-driven protection secures endpoints and cloud collaboration, and our intelligent detection & response is powered by experts who identify business risks by proactively hunting for threats and confronting live attacks. Our consultants partner with enterprises and tech challengers to build resilience through evidence-based security advice. With more than 30 years of experience in building technology that meets business objectives, we've built our portfolio to grow with our partners through flexible commercial models.

WithSecure™ is part of F-Secure Corporation, founded in 1988, and listed on the NASDAQ OMX Helsinki Ltd.

