# Dumping & Abusing Windows Credentials [Part-1]

**Satyam Dubey**
September 13, 2020


## Introduction:

We all know how crucial our credentials are to us, these shared secrets are basically the access to our resources present on various platforms. The whole process of authentication and authorization is pretty much always dependent on these shared secrets which can be in the format of passwords, access tokens, keys, tickets etc. Today many threat actors target to get these shared secrets by leveraging the authentication and authorization process in order to get access to the victim's resources.

### Motive:

In this blog we are going to see what exactly happens under the hood during the process of authentication and authorization in the case of windows platform and how one can dump and abuse the credentials on the attack surface used in the process of authentication and authorization.

## Authentication & Authorization:

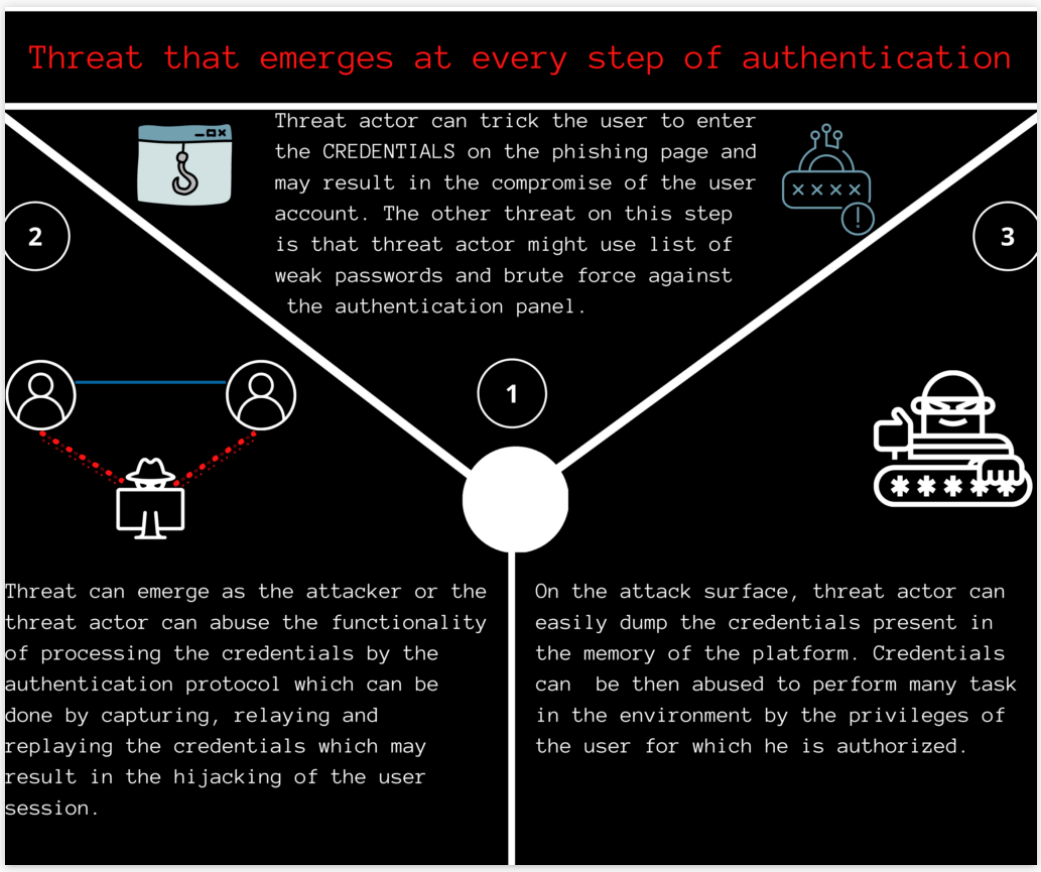
Authentication is the process of verifying the entity on the basis of the information provided by the entity which is identity (identification number, or username) and shared secret. While doing authentication there are various steps that we perform and can be divided into three major steps:

Now from start to end in the above-defined steps of authentication, threats
will try to briefly discuss what threats do exist at every step with respect to t

# Considered Threat Model:

| Steps of authentication | |
|---|---|
| Providing the platform with username and password | Phishing |
| Processing the credentials | Relay attacks, spoofing attacks |
| Storing the credentials for authorization | Dumping and a |

Considered threat model.




Threats at each step

Now let's focus on how one authenticates in windows and learn what happens under the hood for better understanding and this will indeed provide a broader view for us about the authentication in order to abuse the credentials used in the process.

# Understanding the Windows authentication process:

3. Once the user provides the credentials, LSA (known as local security authority) loads the authentication packages like MSV, Kerberos and Negotiate etc. The image below illustrates what packages are available to use in Windows.

Let's say for the first scenario if the preferred authentication package is MSV understand how this authentication package deals with the credentials pro

- MSV1_0 authentication package can be divided into 2 parts:
    1. **The first part,** where the Windows NT client machine on which the computes the hash of password using Windows OWF (One-Way Function). Once the password is converted into a **hash, this hash is stored in the security accounts manager (also referred to** as SAM) database locally or in an active directory database in case of a domain environment.
    2. **The second part** works according to the two scenarios: one where the user authenticates on the system which doesn't exist in the domain environment (local logon scenario) and the other one in which the user authenticates to a windows domain machine or server which is part of the active directory environment (network logon scenario).

https://docs.microsoft.com/en-us/windows/win32/secauthn/images/lsaint4.png

the message of the unsuccessful authentication due to wrong credentials.

- Now that we have looked into the local authentication of the user let's look into the authentication scenario when the machine is part of a domain environment.

    1. Similar to the local authentication the hash is computed by the machine and passed to the second part of the MSV1_0 but NetLogon service does the part of routing the user's hash to the second part of the MSV1_0 authentication package. A little about NetLogon service, it is used for creating a secure channel for authentication purposes in a domain environment.
    2. Now the authentication is carried out according to the NT LanManager (NTLM protocol). The figure below describes the NTLM authentication.

But NTLM is not the only authentication protocol that is used as an authenti
of fact, it is a lesser-used protocol in the case of an active directory environ
used.

Let's see what happens behind the scene when the Kerberos authenticatio
by the LSA. Kerberos is defined as the primitive protocol for authentication i
uses three subprotocol as listed below:

- Authentication Service Exchange
- Ticket-Granting Service Exchange
- Client/Server Exchange

Kerberos uses tickets as the user's network credentials for authentication and provides access to the resource accordingly. The figure below describes the Kerberos authentication flow:

## Stealing the credentials on the attack surface:

One thing to notice about every authentication protocol discussed in the above context is that credentials are stored either on the disk in the form of Database in the above case SAM Database (Registry HIVE) or cached in the memory of process like LSASS (Local Security Authority Subsystem Service) in order to provide access to the network resources seamlessly.

LSASS can store multiple types of credentials that are compatible to the SSP or Authentication Package like:

- LM & NTLM Hash
- Kerberos Tickets
- Keys
- Plaintext Credentials

As this blog deals with the credential stealing and abusing it let's assume c
the initial access on the domain joined machine with the privileges of local

Now before starting the demonstration part I would like to also specify that
Mimikatz, a tool written by Benjamin Delpy in C which deals with windows se

To start with, lets dump the credentials present in the memory of LSASS.exe.
to dump credentials from LSASS, the first one is very straightforward, which i
credentials directly from memory.

But in order to dump the credentials from the memory of a process (lsass.e
privileges to debug the process. This privilege which allows us to debug any
SeDebugPrivilege and is are generally required by the debuggers like OllyD
the functionality of enabling a set of privileges by using the RtlAdjustPrivileg
NTDLL.dll in windows in order to enable a privilege from the calling process

By using the privilege module of mimikatz we can enable SeDebugPrivilege

```
mimikatz # privilege::debug
```

If you want to look more into how to enable SeDebugPrivilege or any other privileges, @jaredatkinson has return PSReflect-Functions to deal with Win32 API functions and the same can be done using the project.

We can now easily dump the credentials from the lsass.exe process as we have enabled the SeDebugPrivilege. Mimikatz provides a module "sekurlsa" which retrieves the user's credentials from the memory of the LSASS process.

```
mimikatz # sekurlsa::logonpasswords
```

Well important thing to notice is that sekurlsa module finds all the credentials which can be found in the memory of LSASS process, but we can also see this authentication packages wise that is calling the command by the authentication packages like:

Dumping the credentials of the msv authentication package only:

```
mimikatz # sekurlsa::msv
```

But this is not the only way to steal credentials using the LSASS process, this the LSASS process using Sysinternals tools like procdump.

```
procdump.exe -accepteula -ma lsass.exe <filepath-output>
```

Apart from that, there are many ways to dump LSASS, one of them ,which I got to know from a tweet by Grzegorz Tworek (@0gtweet).

```
rdrleakdiag.exe /p <pid> /o <outputdir> /fullmemdmp /wait 1
```

This command utilizes a system binary rdrleakdiag.exe which will dump the memory of the process whose PID (process id) is provided in input. Successful execution of the command will result in creation of two files named as minidump_656.dmp and results_656.hlk. [We will use the file with .dmp extension]

In order to use the dump files to retrieve the credentials of the users we need to use the minidump command under the sekurlsa module to make mimikatz aware of the fact that we will be using dump file.

```
mimikatz # privilege::debug
mimikatz # sekurlsa::minidump C:\Users\John\Desktop\minidump_656.dmp
```

All the user's hash who have logon sessions on the machines can be dumpe
But dumping the credentials from the LSASS.exe is not the only option that w
the other option that we have, dumping credentials from SAM registry/HIVE

In order to dump the credentials from SAM we can use the *sam* command u
can provide us with all the local user account hashes, but before that we ne
AUTORITY\SYSTEM to read the credentials [by using SYSKEY to decrypt the S

```
mimikatz # token::elevate
```

```
mimikatz # lsadump::sam
```

Running the above command, we can easily see the hash of the users that
(Security Account Manager) hive.

This can also be done by dumping the System registry hive and SAM registr
files we can retrieve the passwords stored in the local SAM. If we look into th
how sysKey and samKey are retrieved from the Registry HIVE.

https://github.com/gentilkiwi/mimikatz/blob/ba8d11ebe1e79f2df794fcc79d117bda27b75

Saving the SAM & System registry hive in a file to dump the credentials:

```
C:\temp> reg save HKLM\SYSTEM system.hive
C:\temp> reg save HKLM\SAM sam.hive
```

Providing the sam command with the above saved registry hive files we can also dump the hashes from Local SAM registry hive.

```
mimikatz # lsadump::sam /SYSTEM:system.hive /SAM:sam.hive
```

This method can also be referred as Offline method as the threat actor only
SYSTEM registry hive files to their system in order to dump the hash of the us

For this particular operation that involves dumping of credentials, we can a
under the project impacket.

Apart from the following type of hash there exist a different kind of hash i.e.
Domain Cached Credentials which was introduced in windows to keep the
even if the client machine is disconnected from the domain, user can perfo
see under the registry location (HKLM\SECURITY\Cache) after Running the
NT AUTHORITY\SYSTEM privilege the cached credentials keys.

By using command lsadump::cache we can easily dump these hashes.

```
mimikatz # lsadump::cache
```

However, these hashes cannot be passed but can be cracked using tools such as hashcat or John-the-Ripper.

can used in many ways to abuse the Kerberos authentication mechanism. In order to just see what tickets are available on the domain joined machine we can fire the klist command.

```
mimikatz # sekurlsa::tickets /export
```

This command will export the tickets present in the lsass process memory. V
kerberos::list in order to export all the tickets under the context of a user, and
privileges as it doesn't deal with lsass.

For the further demonstration, we will be using Rubeus, a tool made in C# for interacting with Kerberos authentication mechanism and abusing it by @specterops. Best part about Rubeus tool is that it doesn't touch LSASS process memory and therefore doesn't require local admin privileges on the machine.

We can see a detailed output using the klist option in Rubeus.

```
Rubeus.exe klist
```

If we run Rubeus under elevated privileges, we will be able to view and dump the tickets of the other users on the machine as well.

We can dump the tickets now using the dump command in Rubeus. Rubeus dump will get the base64 of all the tickets which can be further used in order to abuse the kerberos authentication resulting in lateral movement.

```
Rubeus.exe dump
```

Side note: add /nowrap to the above command we can get a single line bo

In the above context, we have seen how we can dump credentials from the window machine. One of the prominent sources of dumping credentials wa stores almost every type of credentials for SSO (Single Sign-on) purpose (a focussing more on the LSASS process there were several features made ava process from the threat actors.

One of the features that was arrived with windows 8.1 and is applicable for e Running a process with protection mode named as RunAsPPL which stands Light. By adding and enabling a registry key under "HKLM\SYSTEM\Current

But with mimikatz capability of loading a kernel driver named as mimidrv we can easily remove and add protection to a process. In mimikatz we can load the driver using "!+" and "!-" to unload the driver. I am will not do a deep dive review of the mimikatz's driver but I would suggest going through this awesome blog written by Matt Hand on mimidrv.

Below image shows that we enabled the debug privilege but we are not able to dump the credentials from the LSASS process. Even doing memory dump of the lsass process with the procdump will not be successful.

But loading the mimikatz driver mimidrv will provide us with the capability of removing and enabling the protection of any process.

```
mimikatz #!+
mimikatz #!processprotect /process:lsass.exe /remove
mimikatz #sekurlsa::logonpasswords
```

As we can see in the image above, we are able to dump all the credentials
removing the protection on the lsass process.

But there seems to be the other option available that is much more approa
to dump credentials from the LSASS process which is by running lsass in VS
done by enabling windows Credential Guard. This solves the problem of du
credentials are stored under the LSAISO (Local Security Authority Isolated) p

But there is a workaround for this solution as well and that is to inject mimik
steal the credentials.

Just doing that will inject the SSP in LSASS.exe process and the credentials are listed in log file of mimikatz (mimilsa.log) in the form of clear text.

In the above discussed techniques, we have seen how the credentials can be dumped from various sources like registry hive, LSASS process memory. Now these dumped credentials can be utilized to perform various attacks like Pass-the-Hash, Over-Pass-The-Hash, pass-the-ticket etc.

We will see demonstration about the abuse of these dumped credentials in the next part of the blog.

## Conclusion

Threat actors have always utilized the credentials dumping techniques to move laterally in the domain environment. Sources of dumping these credentials should be heavily monitored like LSASS process etc.

## References:


- https://docs.microsoft.com/en-us/windows/win32/secauthn/msv1-0-authentication-package
- https://support.microsoft.com/en-in/help/102716/ntlm-user-authentication-in-windows

Share this article

## Recent Blogs

**Storm-0501: Unveiling the Tactics Behind Multi-Stage Hybrid Cloud Attacks**

Srishti Chaubey October 14, 2024

**Disney Leaves Slack: Strategic Retreat**

Srishti Chaubey September 24

## Connect with Us!

Subscribe to receive new blog post from PureID in your mail box

Enter your email SUBSCRIBE