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

PureID

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 c
steps:

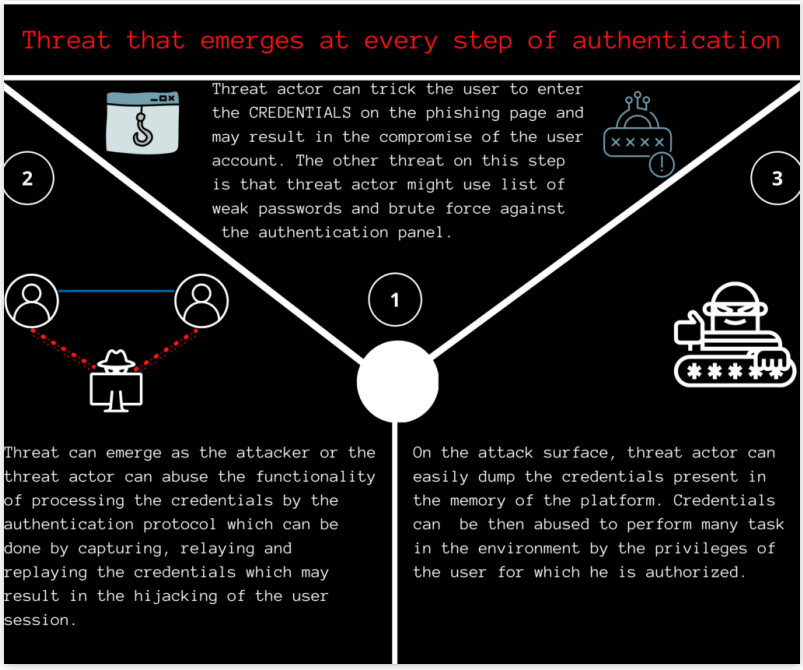
- Providing identity and shared secret.
- Processing the identity and shared secret.
- Storing the credentials for authorization.

Now from start to end in the above-defined steps of authenti
so firstly we will try to briefly discuss what threats do exist at e
windows platform.

PureID

password	
Processing the credentials	Relay attacks, spoofing attacks, poisoning attacks, Session Hijacking
Storing the credentials for authorization	Dumping and abusing the credentials.

Considered threat model.



Threats at each step

Now let's focus on how one authenticates in windows and lec
for better understanding and this will indeed provide a broac
authentication in order to abuse the credentials used in the p

Understanding the Windov authentication process:

1. Starting from the scratch the user is presented wit
   known as Windows Logon UI and provides all the op
   password, PIN etc.) to the user.

illustrates what packages are available to use in Windows.

Let's say for the first scenario if the preferred authentication p
need to understand how this authentication package deals v
user:

- MSV1_0 authentication package can be divided into 2

  1. **The first part,** where the Windows NT client machir
     authenticate computes the hash of password usin
     Function). Once the password is converted into a **h**
     security accounts manager (also referred to as SA
     active directory database in case of a domain environment.

PureID

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

- Let see what goes behind the scene when the user aut
  and the client machine is not part of a domain environ

  1. The first part of the MSV1_0 passes the hash to the
     used to verify with the one present which is present
  2. If the hash computed by the machine is identical t
     SAM database on the machine then the user is gra
     otherwise the user is present with the message of t
     to wrong credentials.

- Now that we have looked into the local authentication
  authentication scenario when the machine is part of a

  1. Similar to the local authentication the hash is com
     to the second part of the MSV1_0 but NetLogon ser
     user's hash to the second part of the MSV1_0 authe
     NetLogon service, it is used for creating a secure ch
     in a domain environment.
  2. Now the authentication is carried out according to
     protocol). The figure below describes the NTLM aut

PureID

But NTLM is not the only authentication protocol that is used as an authentication protocol and as a matter of fact, it is a lesser-used protocol in the case of an active directory environment, in this case, Kerberos is used.

Let's see what happens behind the scene when the Kerberos
preferred one by the LSA. Kerberos is defined as the primitive
domain environment which uses three subprotocol as listed

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

Kerberos uses tickets as the user's network credentials for au
to the resource accordingly. The figure below describes the K

to the part of stealing the credentials present on the attack surface.

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
attacker has the initial access on the domain joined machin
on the box.

Now before starting the demonstration part I would like to als
heavily use Mimikatz, a tool written by Benjamin Delpy in C wl

To start with, lets dump the credentials present in the memo
multiple ways to dump credentials from LSASS, the first one is
use Mimikatz to dump the credentials directly from memory.

But in order to dump the credentials from the memory of a p
the privileges to debug the process. This privilege which allo
program is SeDebugPrivilege and is are generally required b
Mimikatz does provides the functionality of enabling a set of
RtlAdjustPrivilege, a function which is a used by NTDLL.dll in w
privilege from the calling process or thread.

By using the privilege module of mimikatz we can enable Se
process.

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

PureID

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 a
found in the memory of LSASS process, but we can also see th
that is calling the command by the authentication packages

Dumping the credentials of the msv authentication package

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

PureID

But this is not the only way to steal credentials using the LSASS process, this can also be done by dumping by 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
tweet by Grzegorz Tworek (@0gtweet).

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

This command utilizes a system binary rdrleakdiag.exe which
process whose PID (process id) is provided in input. Successfu
result in creation of two files named as minidump_656.dmp
file with .dmp extension]

PureID

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
techniques. But dumping the credentials from the LSASS.exe i
So, let's discuss about the other option that we have, dumpin
registry/HIVE.

In order to dump the credentials from SAM we can use the *sc*
module which can provide us with all the local user account
elevate our privileges to NT AUTORITY\SYSTEM to read the cre
decrypt the SAM hive data].

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

PureID

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


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

This can also be done by dumping the System registry hive a
these two files we can retrieve the passwords stored in the lo
mimikatz we can see how sysKey and samKey are retrieved f

https://github.com/gentilkiwi/mimikatz/blob/ba8d11ebe1e79f2df794fc o.c#L30

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

PureID

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 th
the SAM and SYSTEM registry hive files to their system in order
the victim machine.

PureID

perform the authentication. We can see under the registry location (HKLM\SECURITY\Cache) after Running the registry editor (regedit.msc) with 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 cracke
John-the-Ripper.

These hashes are one of the types of credentials that are sto
other type of credential which is tickets. As discussed above,
which are used in Kerberos authentication mechanism. LSAS
running under the context of LSA (Local Security Authority) st
dumped the hashes present in this process, we can do the sa

Again, we will use the sekurlsa module to dump the tickets fro
These tickets can used in many ways to abuse the Kerberos authentication mechanism. In order

PureID


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

This command will export the tickets present in the lsass proc
command kerberos::list in order to export all the tickets unde
require any high privileges as it doesn't deal with lsass.

PureID

For the further demonstration, we will be using Rubeus, a tool
Kerberos authentication mechanism and abusing it by @spe
tool is that it doesn't touch LSASS process memory and there
privileges on the machine.

Running Rubeus with triage option can list all the tickets pres

```
Rubeus.exe triage
```

PureID

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

We can dump the tickets now using the dump command in F
base64 of all the tickets which can be further used in order to
resulting in lateral movement.

```
Rubeus.exe dump
```

PureID

Side note: add /nowrap to the above command we can get

In the above context, we have seen how we can dump crede
present on the window machine. One of the prominent sourc
lsass.exe process which stores almost every type of credenti
(also for access tokens etc). Now focussing more on the LSAS
features made available to securing the LSASS process from

One of the features that was arrived with windows 8.1 and is
version is Running a process with protection mode named as
Protected Process Light. By adding and enabling a registry ke
"HKLM\SYSTEM\CurrentControlSet\Control\LSA".

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 th
process by removing the protection on the lsass process.

PureID

But there is a workaround for this solution as well and that is to inject mimikatz's ssp (mimilib.dll) in order to steal the credentials.

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

In the above discussed techniques, we have seen how the cr
various sources like registry hive, LSASS process memory. Nov
utilized to perform various attacks like Pass-the-Hash, Over-
etc.

We will see demonstration about the abuse of these dumpe
blog.

Conclusion

Threat actors have always utilized the credentials dumping t
domain environment. Sources of dumping these credentials
LSASS process etc.

References:

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

PureID

Share this article

Recent Blogs

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

Srishti Chaubey October 14, 2024

Disney Leaves Strategic Retr

Srishti Chaubey Sept 2024