skelsec / **pypykatz**

Watch 37 Star 716 Fork 123

Code Issues 8 Pull requests 2 Projects 0 Wiki Security Insights

Mimikatz implementation in pure Python

161 commits **8** branches **0** packages **0** releases **3** contributors MIT

Branch: **master** New pull request Find file Clone or download

skelsec Update README.md Latest commit `5bdf5e3` 2 days ago

pypykatz	meh, rahter not do this	4 days ago
.gitignore	adding registry secrets parsing	4 months ago
LICENSE	Initial commit	2 years ago
Makefile	new version, lot of features	6 days ago
README.md	Update README.md	2 days ago
setup.py	tspkg user-domain switched, live greppable output added.	5 days ago

README.md

pypykatz

Mimikatz implementation in pure Python. At least a part of it :)
Runs on all OS's which support python>=3.6

WIKI

Since version 0.1.1 the command line changed a little. Worry not, I have an awesome WIKI for you.

Installing

Install it via pip or by cloning it from github.
The installer will create a pypykatz executable in the python's Script directory. You can run it from there, should be in your PATH.
Take care, that the github master version might fail because I'm layz to do a proper branch for the new versions. I'll try to create a branch of stable version tho.

Via PIP

```
pip3 install pypykatz
```

Via Github

Install prerequirements

```
pip3 install minidump minikerberos aiowinreg msldap winsspi
```

Clone this repo

```
git clone https://github.com/skelsec/pypykatz.git
cd pypykatz
```

Install it

```
python3 setup.py install
```

Features

General

Platform idependent - all commands have a "live" and a normal version where applicable. The "live" version will use the current system and only works on Windows. The normal commands are platform independent.
Can be used as a library for your projects.

LSASS processing

Can parse the secrets hidden in the LSASS process. This is just like mimikatz's `sekurlsa::` but with different commands.
The main difference here is that all the parsing logic is separated from the data source, so if you define a new reader object you can basically perform the parsing of LSASS from anywhere.

Currently supported data sources:

1. live - reads the LSASS porcess' memory directly
2. minidump - processes a minidump file created by dumping the LSASS process
3. rekall (volatility fork) - processes basically ANY windows memory dumps that rekall can parse
4. pcileech - can dump secrets DIRECTLY via DMA of a live computer
5. remote - this is another project. TBD :)
6. `your project here` seriously, it's super-simple to integrate.

Registry processing

Parses the registry hives to obtain stroed credentials, like NT and LM hashes, domain cached credentials (DCC/DCC2) and LSA secrets.

Currently supported data sources:

1. live - has two techniques to parse live registry. First it's in-memory doesn't touch disk, the second is dumping the hives and parsing them with the offline parser
2. offline (hive files)
3. `your project here` seriously, it's super-simple to integrate.

DPAPI functions - MASTERKEY/BLOB/VAULT/CREDENTIAL

DPAPI is the protector of local secrets of many kinds. Currently the project supports decrypting masterkeys, dpapi blobs, credential files, vault files.
The results are not 100% correct, as there is not much documentation on most of these things. PR is always welcomed!

Currently supported data sources:

1. live - obtains masterkeys directly from LSASS -OR- the user/machine keys from live registry and decrypts the masterkeyfile.
2. hive files (offline)- the user/machine keys from live registry and decrypts the masterkeyfile
3. valid credentials (offline) - can decrypt masterkey files by letting you type in the correct SID and password.

4. `pls don't integrate this part to your project, it's beta`

Impersonating users

Can spawn a new process as any user who has a process running on the machine.
Can assign any available token of choise to your thread
This is just a basic stuff really. Reson is there that I hate to constanly use psexec to get a system shell from admin...

other stuff

yeah... check the code. it has comments and stuff...

Rekall command options

Timestamp override

Reason for this parameter to exist: In order to choose the correct structure for parsing we need the timestamp info of the msv dll file. Rekall sadly doesnt always have this info for some reason, therefore the parsing may be failing.
If the parsing is failing this could solve the issue.

Parameter: `-t`
Values: `0` or `1`
Example:

```
pypykatz.py rekall <momeory_dump_file> -t 0
```

Rekall usage

There are two ways to use rekall-based memory parsing.

Via the `pypykatz rekall` command

You will need to specify the memory file to parse.

Via rekall command line

IMPORTANT NOTICES:

1. If you are just now deciding to install `rekall` please note: it MUST be run in a virtualenv, and you will need to install pypykatz in the same virtualenv!
2. rekall command line is not suitable to show all information acquired from the memory, you should use the `out_file` and `kerberos_dir` command switches!

You can find a rekall plugin file named `pypykatz_rekall.py` in the `plugins` folder of pypykatz.
You will need to copy it in rekall's `plugins/windows` folder, and rename it to `pypykatz.py`.
After this modify the `__init__.py` file located the same folder and add the following line at the end: `from rekall.plugins.windows import pypykatz`
If everything is okay you can use the `pypykatz` command from the `rekall` command line directly.

HELP WANTED

If you want to help me getting this project into a stable release you can send mindiumps of the lsass.exe process to the following link: https://nx5494.your-storageshare.de/s/SJteWj3PPbg8jBA IMPORTANT: please *DO NOT* send dumps of your own machine's lsass process!!! I will be able to see your secrets including hashes/passwords! Send dump files from machines like virtual test systems on which you don't mind that someone will see the credentials. (if you have a test domain system where kerberos is set up that would be the best)
Also I'd apprechiate if you wouldn't spam me...

Why do I need these dumps files?

In order to create mimikatz in Python one would have to create structure definitions of a gazillion different structures (check the original code) without the help of the build-in parser that you'd naturally get from using a native compiler. Now, the problem is that even a single byte misalignemt will render the parsing of these structures run to an error. Problem is mostly revolving around 32 - 64 aligments, so 32 bit Windows version lsass dumps are apprechiated as well!

Summary

I need data I can verify the code on and administer necessary changes on the parsers until everything works fine. Submitting issues on this github page wouldn't help at all without the actual file and github wouldn't like 40-300Mb file attachments.

Prerequisites

Most of my big python projects are aiming for maximum protability, meaning I only use 3rd party packages where absolutely necessary. As of this point three additional packages are used, and I intend to keep it this way.

Python>=3.6
minidump
minikerberos
asn1crypto

Kudos

Benjamin DELPY @gentilkiwi for Mimikatz
Francesco Picasso for the mimikatz.py plugin for volatility
Alberto Solino (@agsolino) for impacket

Crypto

Richard Moore for the AES module
Todd Whiteman for teh DES module

Utils

David Buxton for the timestamp conversion script