










Sign in


 **codewhitesec / HandleKatz** Public


 Notifications


 Fork 103


 Star 572


 **Code**


 Issues 1


 Pull requests 1



 Actions

 Projects


 Security

 Insights

 main ▾


 


Go to file


 Code ▾


About


PIC Isass dumper using cloned handles


 Readme

 Activity

 Custom properties

 572 stars

 12 watching

 103 forks

Report repository


Releases


No releases published


Packages


No packages published


Contributors 3











 bin


 imgs


 loader


 src


 DISCLAIMER.md


 Decoder.py


 PICYourMalware.pdf

 extract.sh

 makefile

 readme.md

 **README**



HandleKatz

Page 1 of 4

This tool was implemented as part of our Brucon2021 conference talk and demonstrates the usage of **cloned handles to Lsass** in order to create an obfuscated memory dump of the same.

It compiles down to an executable **living fully in its text segment**. Thus, the extracted .text segment of the PE file is fully position independent code (=PIC), meaning that it can be treated like any shellcode.

The execution of HandleKatz in memory has a very small footprint, as itself does not allocate any more executable memory and can therefore efficiently be combined with concepts such as (Phantom)DLL-Hollowing as described by [@_ForrestOrr](#). This is in contrast to PIC PE loaders, such as Donut, SRDI or Reflective Loaders which, during PE loading, allocate more executable memory. Additionally, it makes use of a modified version of ReactOS MiniDumpWriteDumpA and bypasses userlandhooks using [RecycledGate](#).

For detailed information please refer to the PDF file **PICYourMalware.pdf** in this repository.

Usage

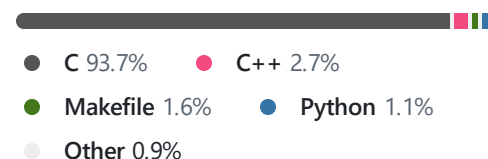
- **make all** to build HandleKatzPIC.exe, HandleKatz.bin and loader.exe

Please note that different compiler (versions) yield different results. This might produce a PE file with relocations.

All tests were carried out using `x86_64-w64-mingw32-gcc` `mingw-gcc version 11.2.0 (GCC)`. The produced PIC was successfully tested on: Windows 10 Pro 10.0.17763. On other versions of windows, API hashes might differ.

To use the PIC, cast a pointer to the shellcode in executable memory and call it according to the definition:

Languages



```
DWORD handleKatz(BOOL b_only_recon, char* ptr_out
```

- **b_only_recon** If set, HandleKatz will only enumerate suitable handles without dumping
- **ptr_output_path** Determines where the obfuscated dump will be written to
- **pid** What PID to clone a handle from
- **ptr_buf_output** A char pointer to which HandleKatz writes its internal output

For deobfuscation of the dump file, the script **Decoder.py** can be used.

Loader implements a sample loader for HandleKatz:

```
loader.exe --pid:7331 --outfile:C:\Temp\dump.obf
```

```
>> .\loader.exe --pid:768 --outfile:C:\Users\user\Desktop\obf.dmp
[*] Recon only: 0
[*] Path dmp: C:\Users\user\Desktop\obf.dmp
[*] Pid to clone from: 768
[*] HandleKatz return value: 1
[*] HandleKatz output:

[*] Attempting to clone Lsass handle from pid: 768
[*] Outfile: C:\Users\user\Desktop\obf.dmp
[+] Found and successfully cloned handle (768) to Lsass in: Lsass.exe (768)
    [+] Handle Rights: 1478
[*] Now trying to dump Lsass ...
[+] Lsass dump is complete
```

Detection

As cloned handles are used along with modified ReactOS code, no ProcessAccess events can be observed on Lsass. However, ProcessAccess events on programs which hold a handle to Lsass can be observed.

Defenders can monitor for ProcessAccess masks with set **PROCESS_DUP_HANDLE (0x0040)** to identify the usage of this tool.

Credits

- Implementation by our [@thefLinkk](#), see [C-To-Shellcode-Examples](#) for more PIC examples.
- [@Hasherezade](#) for [tutorials](#) on the C-To-Shellcode concept
- [@ParanoidNinja](#) for [tutorials](#) on the C-To-Shellcode concept
- [@_ForrestOrr](#) for his amazing [blogpost series](#) on memory artifacts
- [@rookuu](#) for the idea to use ReactOS MiniDumpWriteDump
- [Outflank](#) for documenting direct syscalls and their [InlineWhispers](#) project

[Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)



© 2024 GitHub, Inc.