codewhitesec / HandleKatz Public

main Go to file Code

cwdel Create DISCLAIMER.md a615fef · 2 years ago 16 Commits

bin	Create .gitkeep	3 years ago
imgs	Initial commit	3 years ago
loader	Various Makefile improvements	3 years ago
src	Fixing #8 ; replacing direct syscalls with...	2 years ago
DISCLAIMER.md	Create DISCLAIMER.md	2 years ago
Decoder.py	Initial commit	3 years ago
PICYourMalware.pdf	Added BruCon0x0D slides	3 years ago
extract.sh	Initial commit	3 years ago
makefile	Fixing #8 ; replacing direct syscalls with...	2 years ago
readme.md	Fixing #8 ; replacing direct syscalls with...	2 years ago

README

HandleKatz

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.

About

PIC lsass 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

cwdel David Elze (Code White GmbH)
thefLink
hillu Hilko Bengen

Languages

C 93.7% C++ 2.7% Makefile 1.6% Python 1.1% Other 0.9%

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:

```
DWORD handleKatz(BOOL b_only_recon, char* ptr_output_path, uint32_t
```

- **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.obfuscated
```

```
>> .\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
- React OS for the implementation of MiniDumpWriteDump
- Hilko Bengen for improving the makefile