winterknife / **PINKPANTHER** Public

Code Issues Pull requests Actions Projects Wiki Secu

master Go to file Code

winterknife minor edit ... 22 hours ago 2

Misc	initial commit	22 hours ago
Src	minor edit	22 hours ago
.gitignore	initial commit	22 hours ago
LICENSE	initial commit	22 hours ago
PINKPANTHER.sln	initial commit	22 hours ago
PINKPANTHER.vcx…	initial commit	22 hours ago
PINKPANTHER.vcx…	initial commit	22 hours ago
PINKPANTHER.vcx…	initial commit	22 hours ago
README.md	initial commit	22 hours ago

README.md

PINKPANTHER

Brief

`Windows x64` kernel-mode handcrafted shellcode to replace primary access token of executing process with `SYSTEM` process token for `Elevation of Privilege(EoP)`.

Supported OS Versions

- Windows 7/Windows Server 2008 R2 Build 7601
- Windows 8/Windows Server 2012 Build 9200
- Windows 8.1/Windows Server 2012 R2 Build 9600
- Windows 10 1507/TS1 Build 10240
- Windows 10 1511/TS2 Build 10586
- Windows 10 1607/RS1/Windows Server 2016 Build 14393
- Windows 10 1703/RS2 Build 15063
- Windows 10 1709/RS3 Build 16299
- Windows 10 1803/RS4 Build 17134
- Windows 10 1809/RS5/Windows Server 2019 Build 17763
- Windows 10 1903/19H1 Build 18362
- Windows 10 1909/19H2 Build 18363
- Windows 10 2004/20H1 Build 19041
- Windows 10 2009/20H2 Build 19042
- Windows 10 2104/21H1 Build 19043
- Windows 10 2110/21H2 Build 19044

Building and Deployment

The prerequisites for building this project are:

1. Visual Studio 2019(any edition will do fine)
2. Windows 10 SDK, version 2004
3. Windows 10 WDK, version 2004
4. Python3

After installing the above, it should be as easy as opening the solution with `Visual Studio` and building for `x64` target.

After a successful build, binaries can be found inside the `Bin` directory under the appropriate bitness sub-directory.

Alternatively, you may download ready-to-deploy position independent shellcode from `Releases`.

Please do **NOT** try to deploy the payload on a machine that you rely on to get work done if you are unsure of how it works.

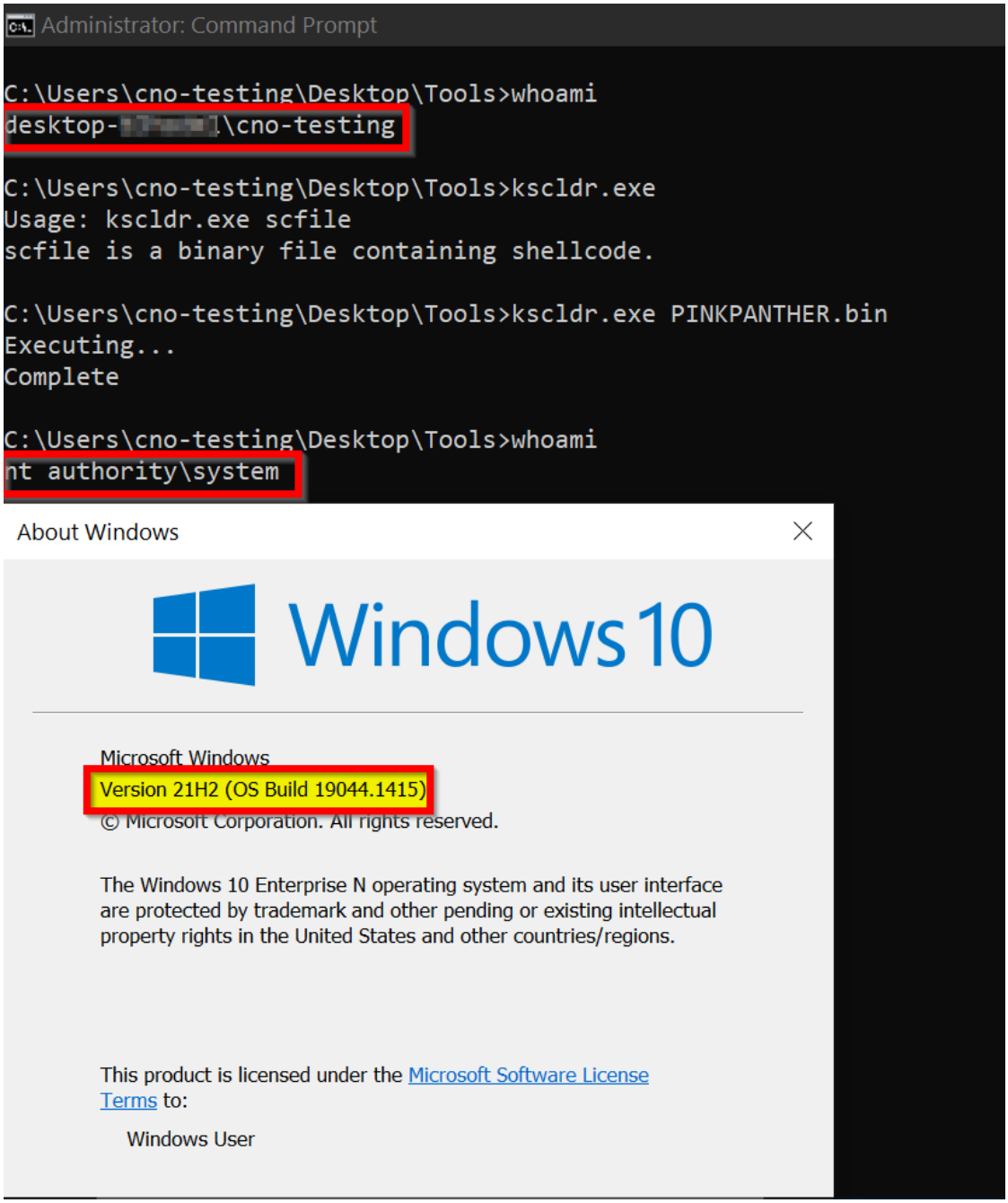
Refer to Microsoft docs for any additional information.

Testing

For testing purposes, I would highly recommend using flare-kscldr to deploy the kernel-mode shellcode on a test `VM` and CodeMachine System setup for kernel development and debugging guide to set up a `Hyper-V Guest VM` with full kernel debugging support.

Optionally, you may also consider automating the process with kdbg-driver-vagrant to quickly spin up a test `VM` with full kernel debugging using Vagrant.

Screenshots



Related Works

1. Exploit Development: Panic! At The Kernel - Token Stealing Payloads Revisited on Windows 10 x64 and Bypassing SMEP
2. Starting with Windows Kernel Exploitation – part 3 – stealing the Access Token
3. [Kernel Exploitation] 2: Payloads
4. Windows Kernel Shellcodes - a compendium

5. Windows Kernel Shellcode on Windows 10 – Part 1
6. Windows Kernel Shellcode : TokenStealer
7. x64 Kernel Privilege Escalation

About

Windows x64 handcrafted token stealing kernel-mode shellcode

- Readme
- GPL-3.0 license
- **211** stars
- **6** watching
- **18** forks

Releases 1

pinkpanther-v1.0.0 Latest
22 hours ago

Packages

No packages published

Languages

- **Assembly** 92.1%
- **Python** 7.9%