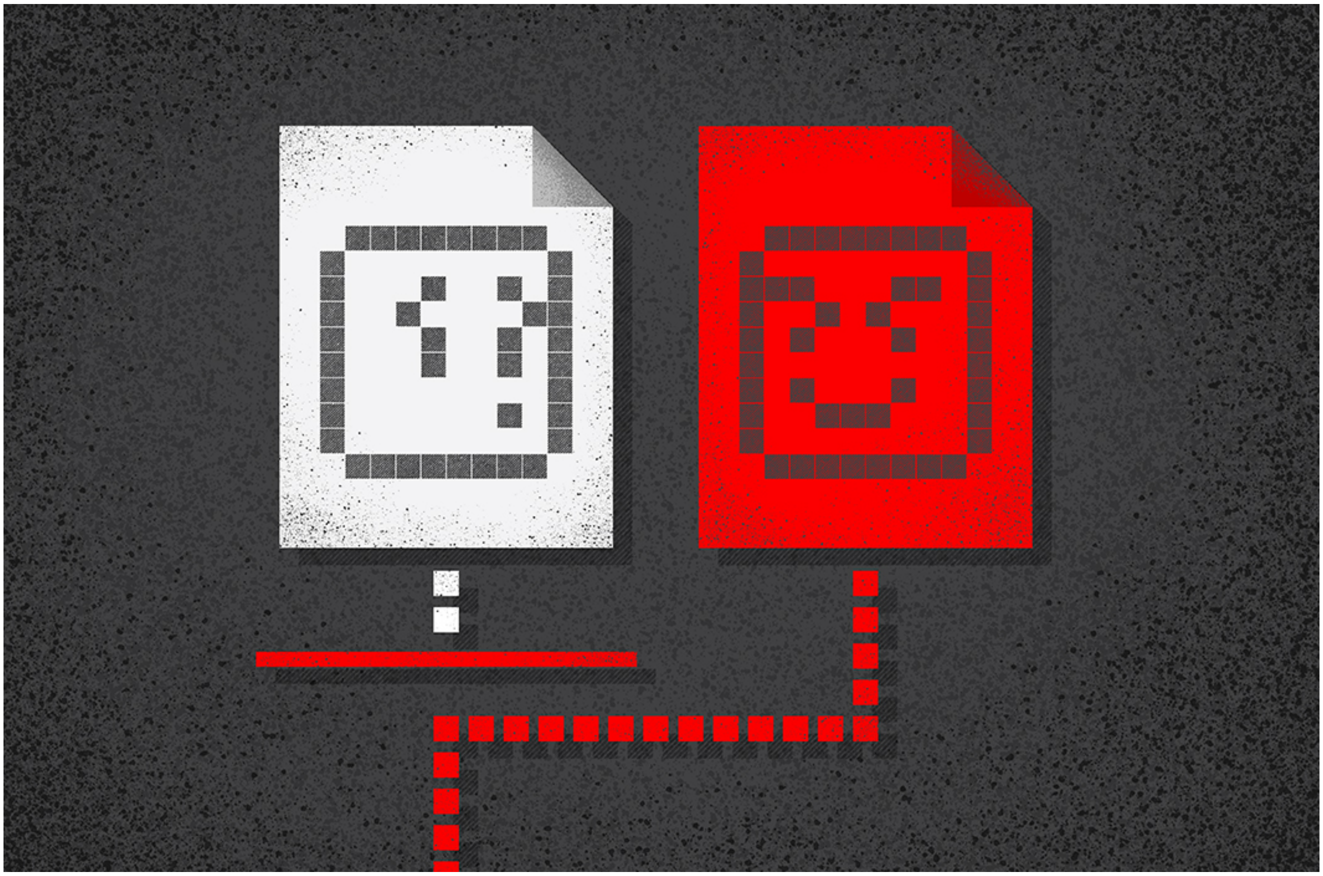
Herpaderping: Security Risk or Unintended Behavior?

crowdstrike.com/blog/herpaderping-security-risk-or-unintended-behavior/

Johnny Shaw

January 21, 2021



The answer to that question often depends on who you ask.

By definition, **process herpaderping** is a hacking technique in which digital adversaries modify on-disk content after the image has been mapped in order to obscure the process. This obscurity can confuse security products or the operating system itself, sometimes allowing malicious code to be executed. For more information on process herpaderping, please read my earlier explainer post here.

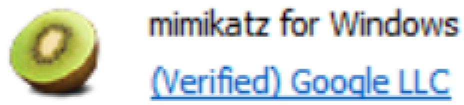
At a Glance: How Process Herpaderping Occurs

1. Write target binary to disk, keeping the handle open. This is what will execute in memory.
2. Map the file as an image section (NtCreateSection, SEC_IMAGE).
3. Create the process object using the section handle (NtCreateProcessEx).
4. Using the same target file handle, obscure the file on disk.
5. Create the initial thread in the process (NtCreateThreadEx). At this point the process creation callback in the kernel will fire. The contents on disk do not match what was mapped. Inspection of the file at this point will result in incorrect attribution.

6. Close the handle. IRP_MJ_CLEANUP will occur here. Since we’ve hidden the contents of what is executing, inspection at this point will result in incorrect attribution.

From an OS’s perspective, herpaderp attacks are often categorized as unintentional activity. But from a developer’s point of view, it’s a clear security threat—and a potent one at that.

To illustrate this principle, let’s take a look at a recent example. Over the summer, it may have appeared that Google distributed a signed copy of Mimikatz. Sounds strange? It should because Google did not distribute a signed copy of Mimikatz. Rather, if you saw this, you were witnessing a case of process herpaderping, an exploit technique that obscures the intentions of a process by modifying the content on disk after the image has been mapped.



mimikatz for Windows

(Verified) Google LLC

My team disclosed this process herpaderping attack to the Microsoft Security Response Center (MSRC) in mid-July. A case was opened a few days later. MSRC concluded their investigation near the end of August and determined the findings of the MSRC investigation were valid but did not require immediate servicing.

And with that, the case was closed—without a real resolution or timeline for future review, despite our reiterated belief to MSRC that this bug is severe.

Review the details here.

What Does Herpaderping Mean for Developers?

The consensus from the cybersecurity community and researchers sees the situation quite differently. The POC shows this is exploitable and as a “defense evasion” or “masquerading” technique.

When an attack of this nature is miscategorized by an OS as unintentional activity it has major cybersecurity implications for any and all developers, not just security vendors. For example, where a Microsoft OS categorizes this attack as unintentional, Microsoft Defender Antivirus scans completed on the image backing the process may not prevent execution by a known malicious binary. Once the file is copied to the desktop, Microsoft Defender will likely realize there is a problem, but by then, the damage is done.

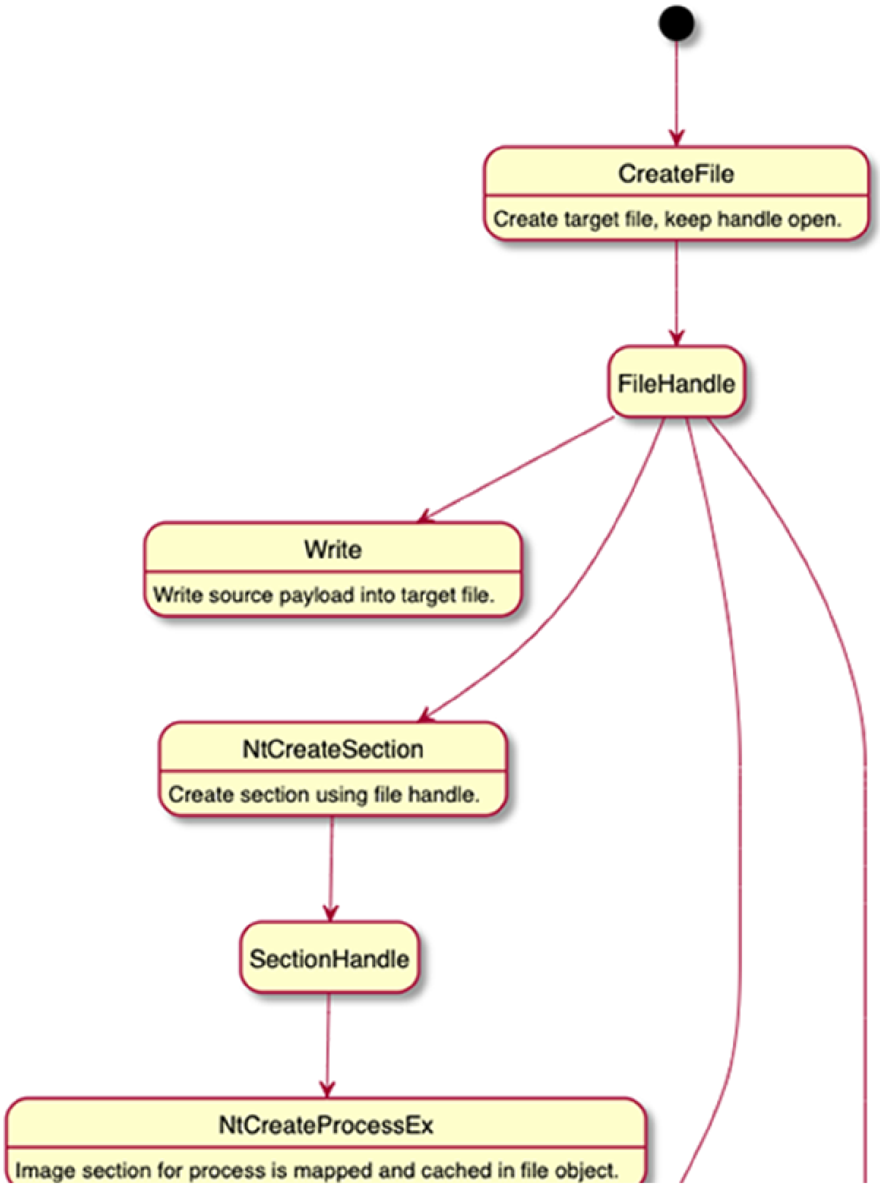
Many antivirus vendors are thought to be vulnerable to this type of attack, which means that many organizations are at risk of experiencing a herpaderping attack.

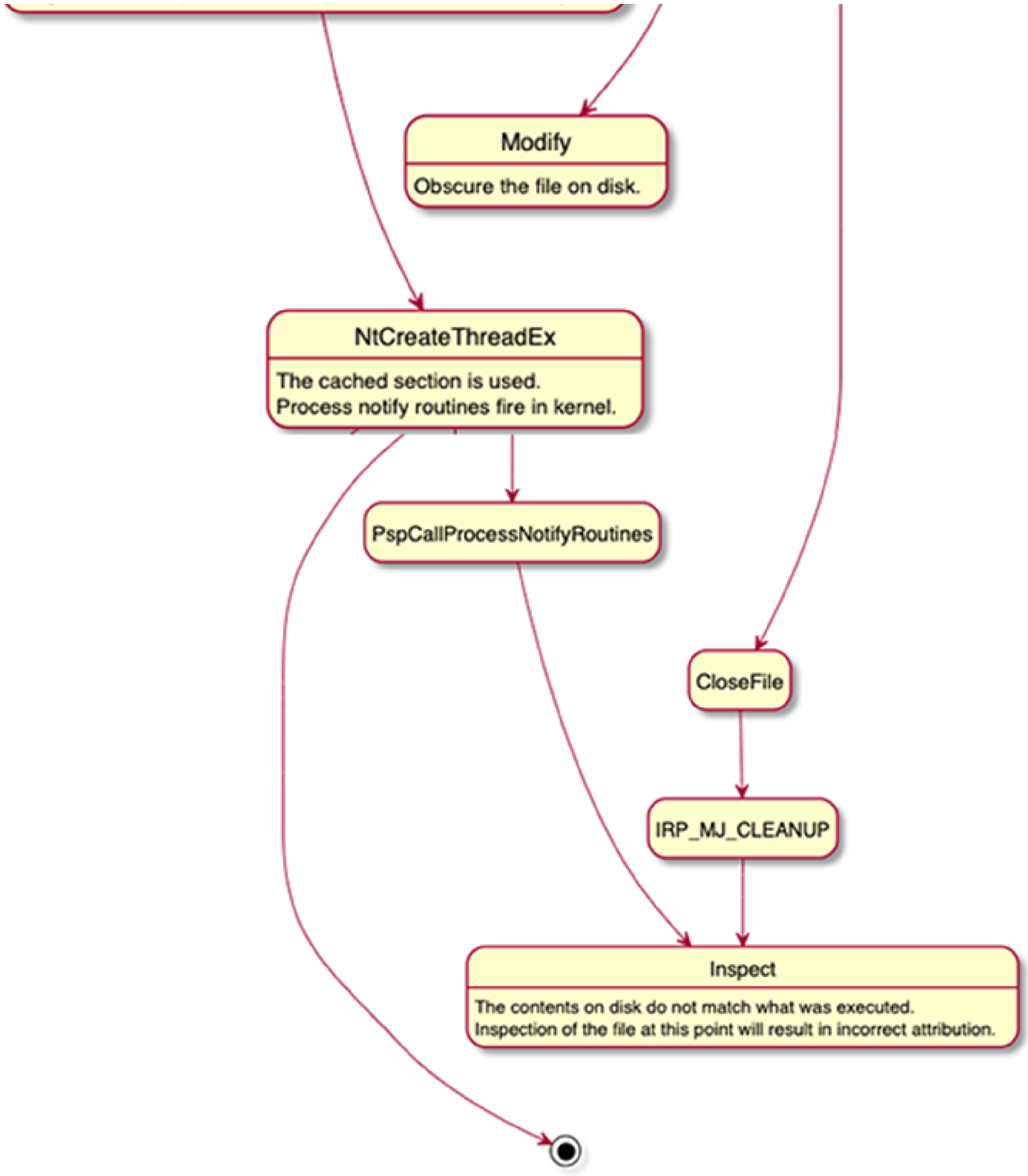
Herpaderping: How It’s Done

When the OS characterizes herpaderping as unintentional activity, it fails to address such exploits, and thus the burden of the solution falls on developers, both cybersecurity vendors and general app developers. But in order to defend against such attacks, we must first understand how they work.

Generally, a security product takes action on process creation by registering a callback in the OS kernel (PsSetCreateProcessNotifyRoutineEx). At this point, a security product may inspect the file that was used to map the executable and determine if this process should be allowed to execute. However, the kernel callback is invoked when the initial thread is inserted, not when the process object is created.

Because of this disconnect, an actor can create and map a process, modify the content of the file, then create the initial thread. A product that does inspection at the creation callback would then see the modified content.


CreateFile
Create target file, keep handle open.
FileHandle
Write
Write source payload into target file.
NtCreateSection
Create section using file handle.
SectionHandle
NtCreateProcessEx
Image section for process is mapped and cached in file object.


Modify
Obscure the file on disk.
NtCreateThreadEx
The cached section is used.
Process notify routines fire in kernel.
PspCallProcessNotifyRoutines
CloseFile
IRP_MJ_CLEANUP
Inspect
The contents on disk do not match what was executed.
Inspection of the file at this point will result in incorrect attribution.

Click image to enlarge

In addition, some products use an on-write scanning approach that consists of monitoring for file writes. A familiar optimization is recording the file as written and deferring the actual inspection until IRP_MJ_CLEANUP occurs (e.g. the file handle is closed). Thus, an actor using a write -> map -> modify -> execute -> close workflow will subvert on-write scanning that solely relies on inspection at IRP_MJ_CLEANUP.

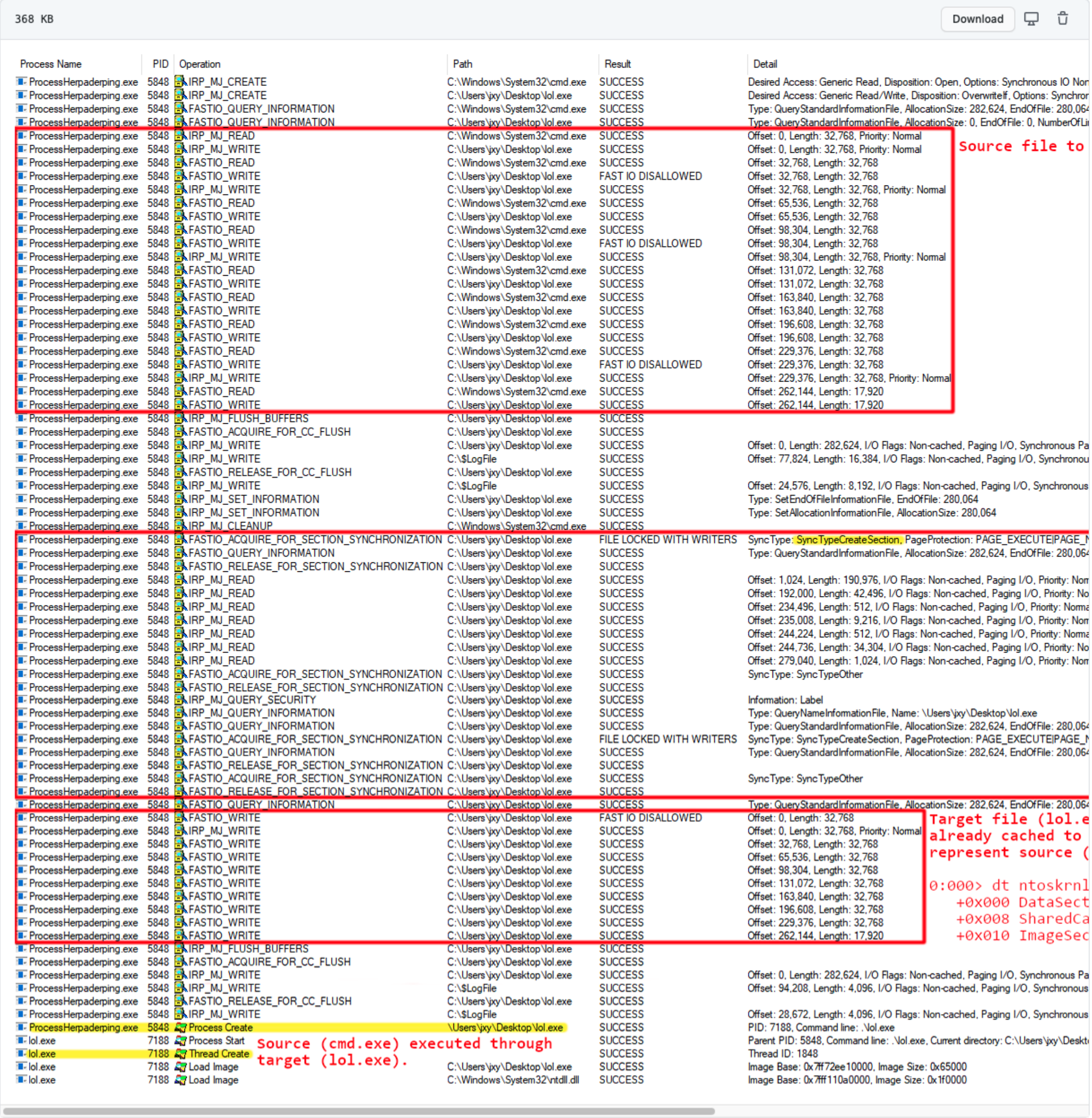
To abuse this convention, adversaries first write a binary to a target file on disk. Then, they map an image of the target file and provide it to the OS to use for process creation. The OS then maps the original binary. Using the existing file handle, and before creating the initial thread, the adversary then modifies the target file content to obscure or fake the file backing the image. Later, the initial thread is created in order to begin execution of the original binary. Finally, the target file handle can be closed.

Let’s look at an example of herpaderping in action.

As you can see in the demo linked above, Cmd.exe is used as the execution target. The first run overwrites the bytes on disk with a pattern. The second run overwrites .exe with ProcessHacker.exe.

The herpaderping tool then fixes the binary to look as close to ProcessHacker.exe as possible, even retaining the original signature. Note the multiple executions of the same binary and how the process looks to the user compared to what is in the file on disk.

Looking at this flow from Process Monitor, we observe the source file being written and the OS mapping the image at the section creation. Using the same handle, the exploit overwrites the source binary with whatever it likes before creating the initial thread. By the time the process creation notification fires in the kernel the file backing the image is not what was mapped:



Process Name	PID	Operation	Path	Result	Detail
ProcessHerpaderping.exe	5848	IRP_MJ_CREATE	C:\Windows\System32\cmd.exe	SUCCESS	Desired Access: Generic Read, Disposition: Open, Options: Synchronous IO Nor
ProcessHerpaderping.exe	5848	IRP_MJ_CREATE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Desired Access: Generic Read/Write, Disposition: OverwriteIf, Options: Synchror
ProcessHerpaderping.exe	5848	FASTIO_QUERY_INFORMATION	C:\Windows\System32\cmd.exe	SUCCESS	Type: QueryStandardInformationFile, AllocationSize: 282,624, EndOfFile: 280,064
ProcessHerpaderping.exe	5848	FASTIO_QUERY_INFORMATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Type: QueryStandardInformationFile, AllocationSize: 0, EndOfFile: 0, NumberOfLi
ProcessHerpaderping.exe	5848	IRP_MJ_READ	C:\Windows\System32\cmd.exe	SUCCESS	Offset: 0, Length: 32,768, Priority: Normal
ProcessHerpaderping.exe	5848	IRP_MJ_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 0, Length: 32,768, Priority: Normal
ProcessHerpaderping.exe	5848	FASTIO_READ	C:\Windows\System32\cmd.exe	SUCCESS	Offset: 32,768, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	FAST IO DISALLOWED	Offset: 32,768, Length: 32,768
ProcessHerpaderping.exe	5848	IRP_MJ_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 32,768, Length: 32,768, Priority: Normal
ProcessHerpaderping.exe	5848	FASTIO_READ	C:\Windows\System32\cmd.exe	SUCCESS	Offset: 65,536, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 65,536, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_READ	C:\Windows\System32\cmd.exe	SUCCESS	Offset: 98,304, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	FAST IO DISALLOWED	Offset: 98,304, Length: 32,768
ProcessHerpaderping.exe	5848	IRP_MJ_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 98,304, Length: 32,768, Priority: Normal
ProcessHerpaderping.exe	5848	FASTIO_READ	C:\Windows\System32\cmd.exe	SUCCESS	Offset: 131,072, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 131,072, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_READ	C:\Windows\System32\cmd.exe	SUCCESS	Offset: 163,840, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 163,840, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_READ	C:\Windows\System32\cmd.exe	SUCCESS	Offset: 196,608, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 196,608, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_READ	C:\Windows\System32\cmd.exe	SUCCESS	Offset: 229,376, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	FAST IO DISALLOWED	Offset: 229,376, Length: 32,768
ProcessHerpaderping.exe	5848	IRP_MJ_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 229,376, Length: 32,768, Priority: Normal
ProcessHerpaderping.exe	5848	FASTIO_READ	C:\Windows\System32\cmd.exe	SUCCESS	Offset: 262,144, Length: 17,920
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 262,144, Length: 17,920
ProcessHerpaderping.exe	5848	IRP_MJ_FLUSH_BUFFERS	C:\Users\jxy\Desktop\lol.exe	SUCCESS	
ProcessHerpaderping.exe	5848	FASTIO_ACQUIRE_FOR_CC_FLUSH	C:\Users\jxy\Desktop\lol.exe	SUCCESS	
ProcessHerpaderping.exe	5848	IRP_MJ_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 0, Length: 282,624, I/O Flags: Non-cached, Paging I/O, Synchronous Pa
ProcessHerpaderping.exe	5848	IRP_MJ_WRITE	C:\$LogFile	SUCCESS	Offset: 77,824, Length: 16,384, I/O Flags: Non-cached, Paging I/O, Synchronou
ProcessHerpaderping.exe	5848	FASTIO_RELEASE_FOR_CC_FLUSH	C:\Users\jxy\Desktop\lol.exe	SUCCESS	
ProcessHerpaderping.exe	5848	IRP_MJ_WRITE	C:\$LogFile	SUCCESS	Offset: 24,576, Length: 8,192, I/O Flags: Non-cached, Paging I/O, Synchronous
ProcessHerpaderping.exe	5848	IRP_MJ_SET_INFORMATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Type: SetEndOfFileInformationFile, EndOfFile: 280,064
ProcessHerpaderping.exe	5848	IRP_MJ_SET_INFORMATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Type: SetAllocationInformationFile, AllocationSize: 280,064
ProcessHerpaderping.exe	5848	IRP_MJ_CLEANUP	C:\Windows\System32\cmd.exe	SUCCESS	
ProcessHerpaderping.exe	5848	FASTIO_ACQUIRE_FOR_SECTION_SYNCHRONIZATION	C:\Users\jxy\Desktop\lol.exe	FILE LOCKED WITH WRITERS	SyncType: SyncTypeCreateSection, PageProtection: PAGE_EXECUTE|PAGE_N
ProcessHerpaderping.exe	5848	FASTIO_QUERY_INFORMATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Type: QueryStandardInformationFile, AllocationSize: 282,624, EndOfFile: 280,064
ProcessHerpaderping.exe	5848	FASTIO_RELEASE_FOR_SECTION_SYNCHRONIZATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	
ProcessHerpaderping.exe	5848	IRP_MJ_READ	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 1,024, Length: 190,976, I/O Flags: Non-cached, Paging I/O, Priority: Nor
ProcessHerpaderping.exe	5848	IRP_MJ_READ	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 192,000, Length: 42,496, I/O Flags: Non-cached, Paging I/O, Priority: No
ProcessHerpaderping.exe	5848	IRP_MJ_READ	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 234,496, Length: 512, I/O Flags: Non-cached, Paging I/O, Priority: Norma
ProcessHerpaderping.exe	5848	IRP_MJ_READ	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 235,008, Length: 9,216, I/O Flags: Non-cached, Paging I/O, Priority: Nor
ProcessHerpaderping.exe	5848	IRP_MJ_READ	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 244,224, Length: 512, I/O Flags: Non-cached, Paging I/O, Priority: Norma
ProcessHerpaderping.exe	5848	IRP_MJ_READ	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 244,736, Length: 34,304, I/O Flags: Non-cached, Paging I/O, Priority: No
ProcessHerpaderping.exe	5848	IRP_MJ_READ	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 279,040, Length: 1,024, I/O Flags: Non-cached, Paging I/O, Priority: Nor
ProcessHerpaderping.exe	5848	FASTIO_ACQUIRE_FOR_SECTION_SYNCHRONIZATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	SyncType: SyncTypeOther
ProcessHerpaderping.exe	5848	FASTIO_RELEASE_FOR_SECTION_SYNCHRONIZATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	
ProcessHerpaderping.exe	5848	IRP_MJ_QUERY_SECURITY	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Information: Label
ProcessHerpaderping.exe	5848	IRP_MJ_QUERY_INFORMATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Type: QueryNameInformationFile, Name: \Users\jxy\Desktop\lol.exe
ProcessHerpaderping.exe	5848	FASTIO_QUERY_INFORMATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Type: QueryStandardInformationFile, AllocationSize: 282,624, EndOfFile: 280,064
ProcessHerpaderping.exe	5848	FASTIO_ACQUIRE_FOR_SECTION_SYNCHRONIZATION	C:\Users\jxy\Desktop\lol.exe	FILE LOCKED WITH WRITERS	SyncType: SyncTypeCreateSection, PageProtection: PAGE_EXECUTE|PAGE_N
ProcessHerpaderping.exe	5848	FASTIO_QUERY_INFORMATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Type: QueryStandardInformationFile, AllocationSize: 282,624, EndOfFile: 280,064
ProcessHerpaderping.exe	5848	FASTIO_RELEASE_FOR_SECTION_SYNCHRONIZATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	
ProcessHerpaderping.exe	5848	FASTIO_ACQUIRE_FOR_SECTION_SYNCHRONIZATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	SyncType: SyncTypeOther
ProcessHerpaderping.exe	5848	FASTIO_RELEASE_FOR_SECTION_SYNCHRONIZATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	
ProcessHerpaderping.exe	5848	FASTIO_QUERY_INFORMATION	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Type: QueryStandardInformationFile, AllocationSize: 282,624, EndOfFile: 280,064
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	FAST IO DISALLOWED	Offset: 0, Length: 32,768
ProcessHerpaderping.exe	5848	IRP_MJ_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 0, Length: 32,768, Priority: Normal
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 32,768, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 65,536, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 98,304, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 131,072, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 163,840, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 196,608, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 229,376, Length: 32,768
ProcessHerpaderping.exe	5848	FASTIO_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 262,144, Length: 17,920
ProcessHerpaderping.exe	5848	IRP_MJ_FLUSH_BUFFERS	C:\Users\jxy\Desktop\lol.exe	SUCCESS	
ProcessHerpaderping.exe	5848	FASTIO_ACQUIRE_FOR_CC_FLUSH	C:\Users\jxy\Desktop\lol.exe	SUCCESS	
ProcessHerpaderping.exe	5848	IRP_MJ_WRITE	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Offset: 0, Length: 282,624, I/O Flags: Non-cached, Paging I/O, Synchronous Pa
ProcessHerpaderping.exe	5848	IRP_MJ_WRITE	C:\$LogFile	SUCCESS	Offset: 94,208, Length: 4,096, I/O Flags: Non-cached, Paging I/O, Synchronous
ProcessHerpaderping.exe	5848	FASTIO_RELEASE_FOR_CC_FLUSH	C:\Users\jxy\Desktop\lol.exe	SUCCESS	
ProcessHerpaderping.exe	5848	IRP_MJ_WRITE	C:\$LogFile	SUCCESS	Offset: 28,672, Length: 4,096, I/O Flags: Non-cached, Paging I/O, Synchronous
ProcessHerpaderping.exe	5848	Process Create	\Users\jxy\Desktop\lol.exe	SUCCESS	PID: 7188, Command line: .\lol.exe
lol.exe	7188	Process Start		SUCCESS	Parent PID: 5848, Command line: .\lol.exe, Current directory: C:\Users\jxy\Deskt
lol.exe	7188	Thread Create		SUCCESS	Thread ID: 1848
lol.exe	7188	Load Image	C:\Users\jxy\Desktop\lol.exe	SUCCESS	Image Base: 0x7ff72ee10000, Image Size: 0x65000
lol.exe	7188	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7ff110a0000, Image Size: 0x1f0000

Source file to

Target file (lol.e
already cached to
represent source (

```
0:000> dt ntoskrnl
   +0x000 DataSect
   +0x008 SharedCa
   +0x010 ImageSec
```

Source (cmd.exe) executed through target (lol.exe).

Click image to enlarge

The SECTION_OBJECT_POINTERS of the FILE_OBJECT play a key role here. The OS will cache the initial image mapping and re-use the already mapped section, even if there is active write access to the FILE_OBJECT. This also means that a user can open the original file with exclusive access. While this won’t necessarily create an issue for the kernel callback, it does affect downstream logic in that it assumes that when the user opens the file with read access it will be broken. However, such logic does not apply, given that the file content has been overwritten. Further, the kernel callback is hopeless too, since reading directly from the file using that FILE_OBJECT will read the wrong data.

This also means if a user tries to execute that process again it will result in a sharing violation. From user mode, without access to that original target file handle, no one may conventionally execute the process.

Resolving This Issue: A Short-term Fix and Long-term Solution

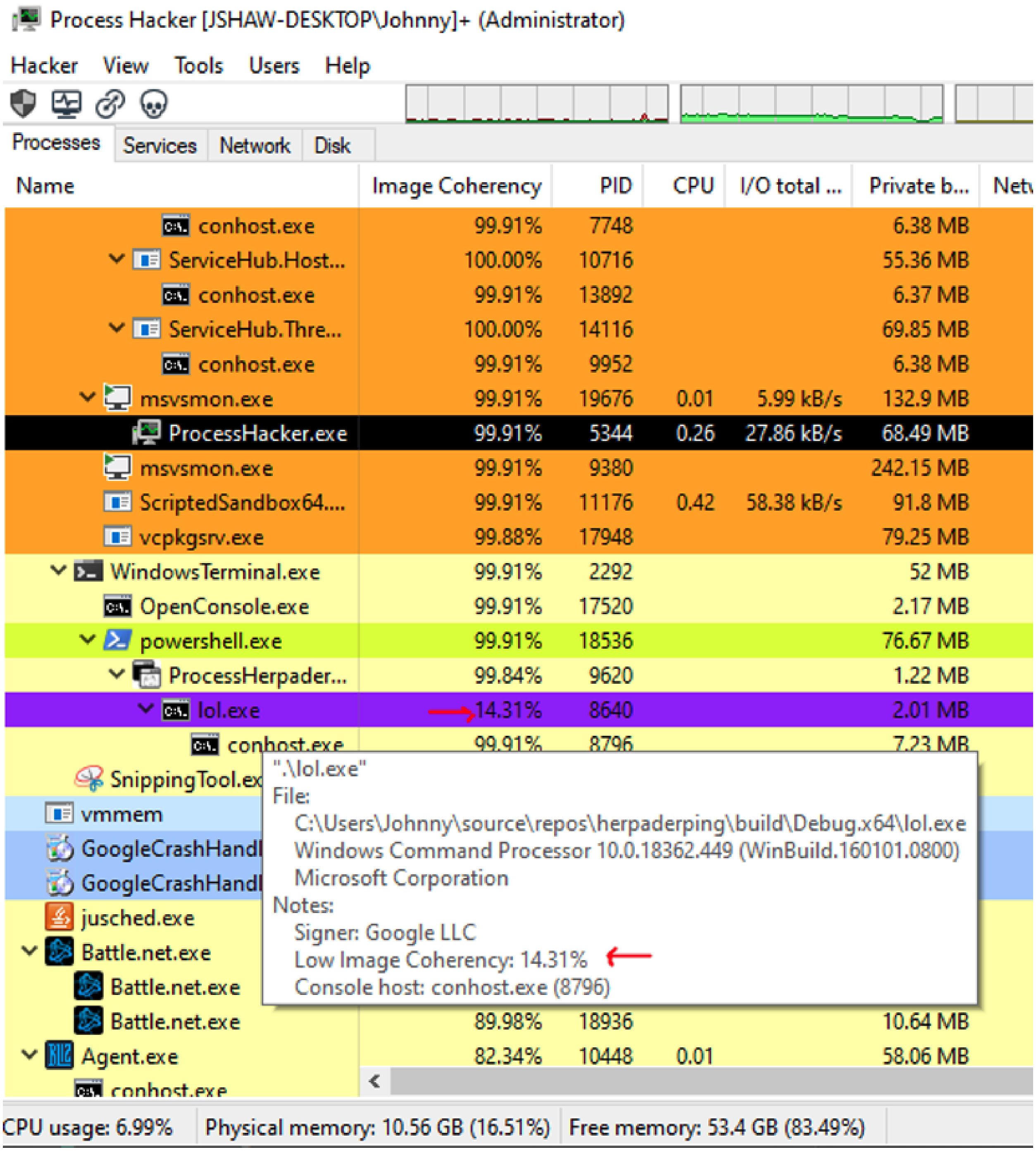
Unfortunately, there is not a clear fix for herpaderping attacks. It seems reasonable that preventing an image section from being mapped/cached when there is write access to the file should close the hole. However, that may or may not be a practical solution.

There is a frustration among the development community about the incoherency between what is on disc and what it is going to execute, which is a common issue with all major operating systems. As such, that is something that should be considered when designing a security product.

While there isn’t a clear mechanism to guarantee that coherency, there are ways that security vendors, as well as security engineers and analysts, can foresee this issue and minimize the risk.

One way to detect this type of exploit is to look for write access to the related file object when the process is created. The exploit seems achievable through a higher-level API call. That said, the API juggling involved usually requires a native-call, which allows the attacker to tightly control the process creation flow.

Another way to detect this type of exploit is checking for coherency between the file on disk and the mapped process. Viewing the image coherency between the file on disk and the mapped process may be seen in an addition I’ve made to Process Hacker, and also shows the spoofed display of “Google” as the “signer” due to the exploit.



Name	Image Coherency	PID	CPU	I/O total ...	Private b...	Netw
conhost.exe	99.91%	7748			6.38 MB	
ServiceHub.Host...	100.00%	10716			55.36 MB	
conhost.exe	99.91%	13892			6.37 MB	
ServiceHub.Thre...	100.00%	14116			69.85 MB	
conhost.exe	99.91%	9952			6.38 MB	
msvsmon.exe	99.91%	19676	0.01	5.99 kB/s	132.9 MB	
ProcessHacker.exe	99.91%	5344	0.26	27.86 kB/s	68.49 MB	
msvsmon.exe	99.91%	9380			242.15 MB	
ScriptedSandbox64....	99.91%	11176	0.42	58.38 kB/s	91.8 MB	
vcpkgsrv.exe	99.88%	17948			79.25 MB	
WindowsTerminal.exe	99.91%	2292			52 MB	
OpenConsole.exe	99.91%	17520			2.17 MB	
powershell.exe	99.91%	18536			76.67 MB	
ProcessHerpader...	99.84%	9620			1.22 MB	
lol.exe	14.31%	8640			2.01 MB	
conhost.exe	99.91%	8796			7.23 MB	
SnippingTool.ex						
vmmem						
GoogleCrashHandl						
GoogleCrashHandl						
jusched.exe						
Battle.net.exe						
Battle.net.exe						
Battle.net.exe	89.98%	18936			10.64 MB	
Agent.exe	82.34%	10448	0.01		58.06 MB	
conhost.exe						

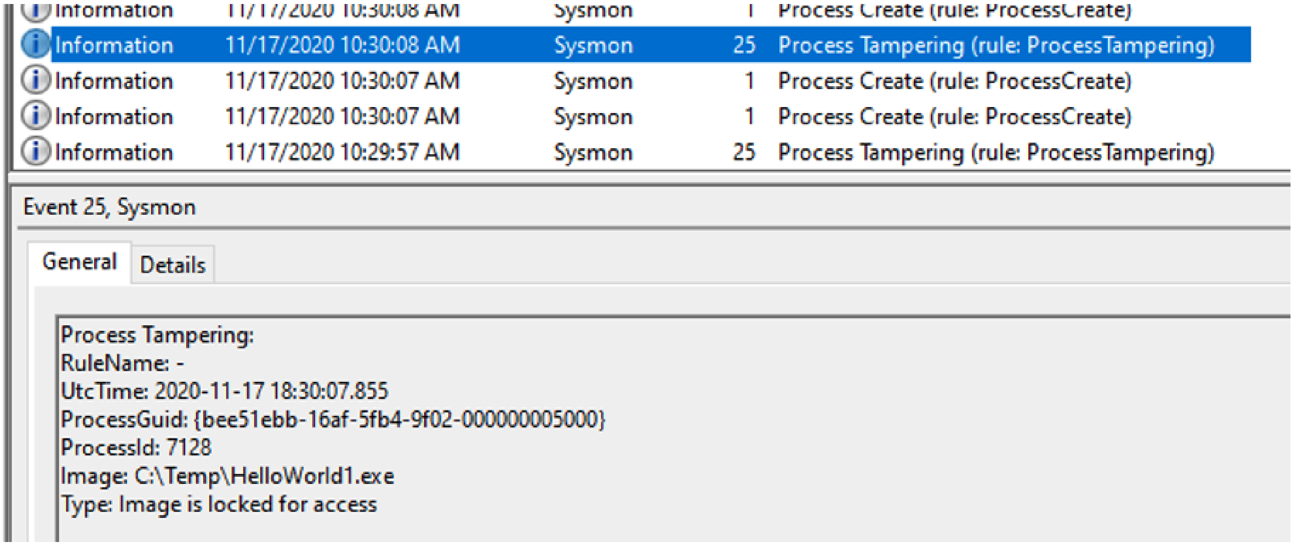
".\lol.exe"
File:
    C:\Users\Johnny\source\repos\herpaderping\build\Debug.x64\lol.exe
    Windows Command Processor 10.0.18362.449 (WinBuild.160101.0800)
    Microsoft Corporation
Notes:
    Signer: Google LLC
    Low Image Coherency: 14.31% ←
    Console host: conhost.exe (8796)

CPU usage: 6.99% | Physical memory: 10.56 GB (16.51%) | Free memory: 53.4 GB (83.49%)

Click image to enlarge

Here is a link to the enhancement proposal I've made to Process Hacker: https://github.com/processhacker/processhacker/issues/744

Process Tampering may also be seen through a recent addition to SysMon by Mark Russinovich, CTO of Microsoft Azure, reproduced below and in this tweet.

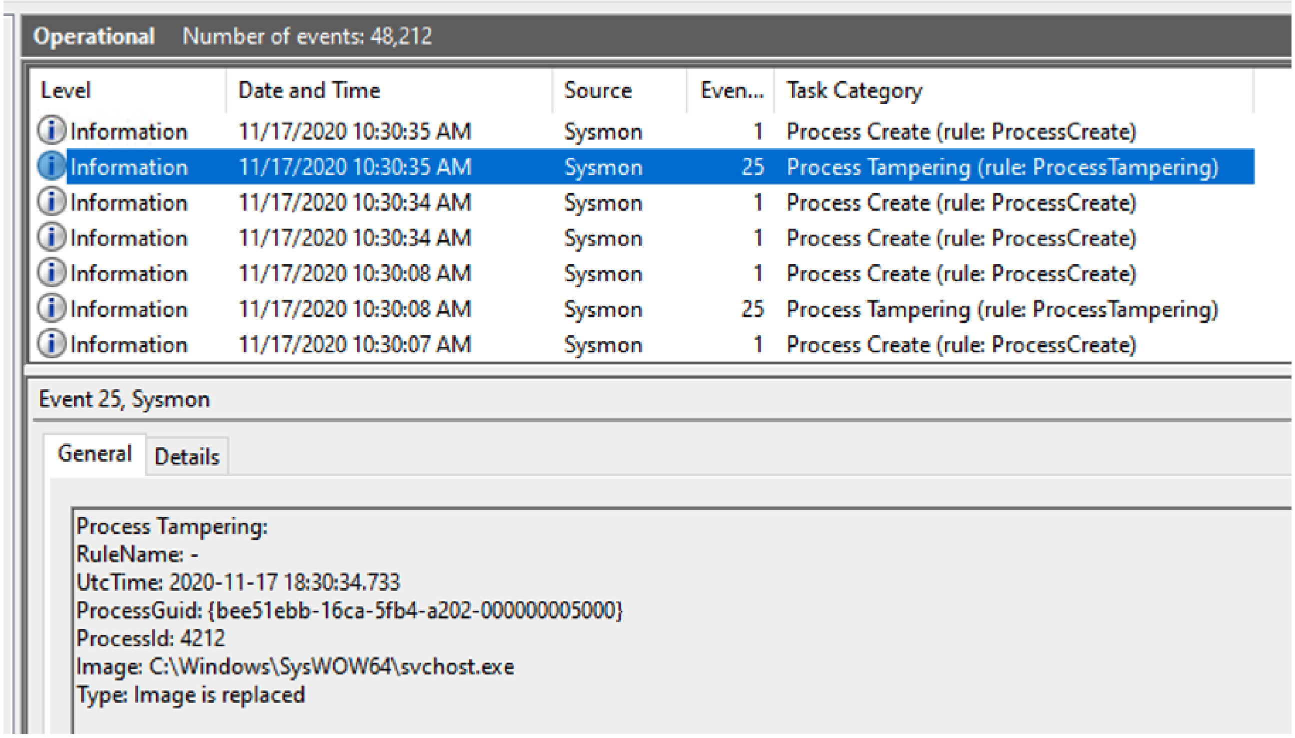


Information	11/17/2020 10:30:08 AM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	11/17/2020 10:30:08 AM	Sysmon	25	Process Tampering (rule: ProcessTampering)
Information	11/17/2020 10:30:07 AM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	11/17/2020 10:30:07 AM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	11/17/2020 10:29:57 AM	Sysmon	25	Process Tampering (rule: ProcessTampering)

Event 25, Sysmon

Process Tampering:
RuleName: -
UtcTime: 2020-11-17 18:30:07.855
ProcessGuid: {bee51ebb-16af-5fb4-9f02-000000005000}
ProcessId: 7128
Image: C:\Temp\HelloWorld1.exe
Type: Image is locked for access

Click image to enlarge



Operational Number of events: 48,212

Level	Date and Time	Source	Even...	Task Category
Information	11/17/2020 10:30:35 AM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	11/17/2020 10:30:35 AM	Sysmon	25	Process Tampering (rule: ProcessTampering)
Information	11/17/2020 10:30:34 AM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	11/17/2020 10:30:34 AM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	11/17/2020 10:30:08 AM	Sysmon	1	Process Create (rule: ProcessCreate)
Information	11/17/2020 10:30:08 AM	Sysmon	25	Process Tampering (rule: ProcessTampering)
Information	11/17/2020 10:30:07 AM	Sysmon	1	Process Create (rule: ProcessCreate)

Event 25, Sysmon

Process Tampering:
RuleName: -
UtcTime: 2020-11-17 18:30:34.733
ProcessGuid: {bee51ebb-16ca-5fb4-a202-000000005000}
ProcessId: 4212
Image: C:\Windows\SysWOW64\svchost.exe
Type: Image is replaced

Click image to enlarge

As for a long-term solution, that would require working with the OS to agree on a joint approach. We're hopeful that as the technique gains broader awareness, the OS vendors will work with the cybersecurity community to adopt more stringent countermeasures directly in the OS.

What are your thoughts? How do you reduce the risk of herpaderping in the short-term? What steps do you think the OS community must take to address this issue more systematically? For additional context and insight, see what industry professionals are saying and join the conversation on Twitter.

https://twitter.com/gentilkiwi/status/1321001331755286529?s=20

https://twitter.com/Mordor_Project/status/1320949216018112514?s=20

https://twitter.com/jxy__s/status/1320873966752329729?s=20

https://twitter.com/analyzev/status/1320914701514084352?s=20

https://twitter.com/markrussinovich/status/1328769178233237504?s=20

Does this work sound interesting to you? Visit CrowdStrike's Engineering and Technology page to learn more about our engineering team, our culture and current open positions: https://www.crowdstrike.com/careers/engineering-technology-team/