PPLdump Is Dead. Long Live PPLdump!

elastic security labs

Gabriel Landau
Principal, Elastic

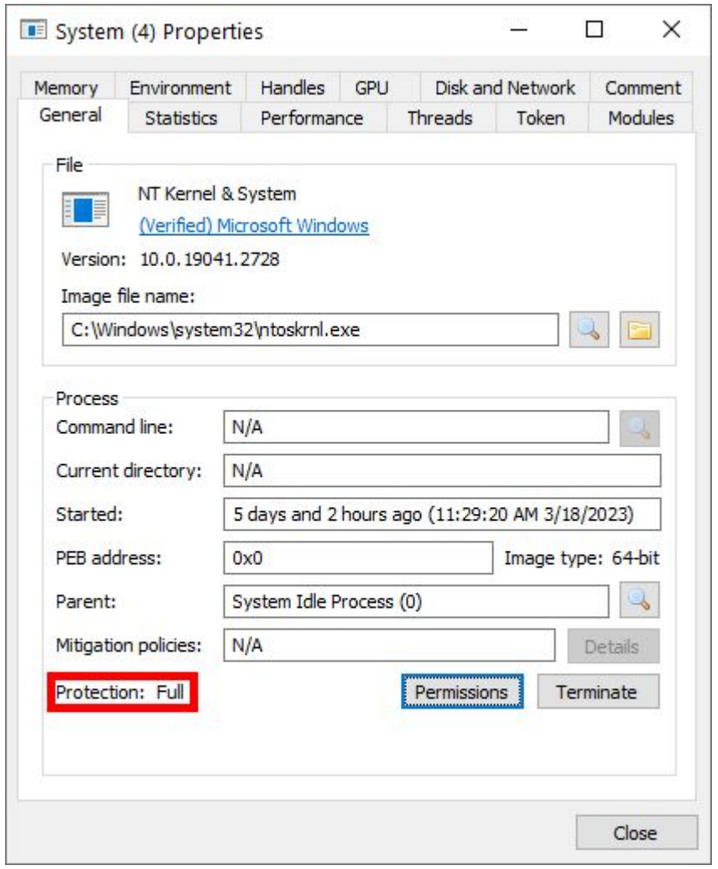
Gabriel Landau is a principal at Elastic Security. His public research includes Process Ghosting, AV sandboxing attacks, Kernel Mode Threats and Practical Defenses (Black Hat USA), Hide Your Valuables - Mitigating Physical Credential Dumping Attacks (Shmoocon), PPLGuard, and CI Spotter. His non-public work includes endpoint protections, exploit mitigation, product and DRM evaluation, and malware reversing. Though he mostly wears blue these days, his heart will always be red.

Outline

- **Introduction**
  - **What is a protected process?**
  - **Implementation**
- Attacks
  - Historical
  - Current
- New Research
  - Novel Attack
  - Chaining Exploits
  - Mitigation

Protected Process (PP)

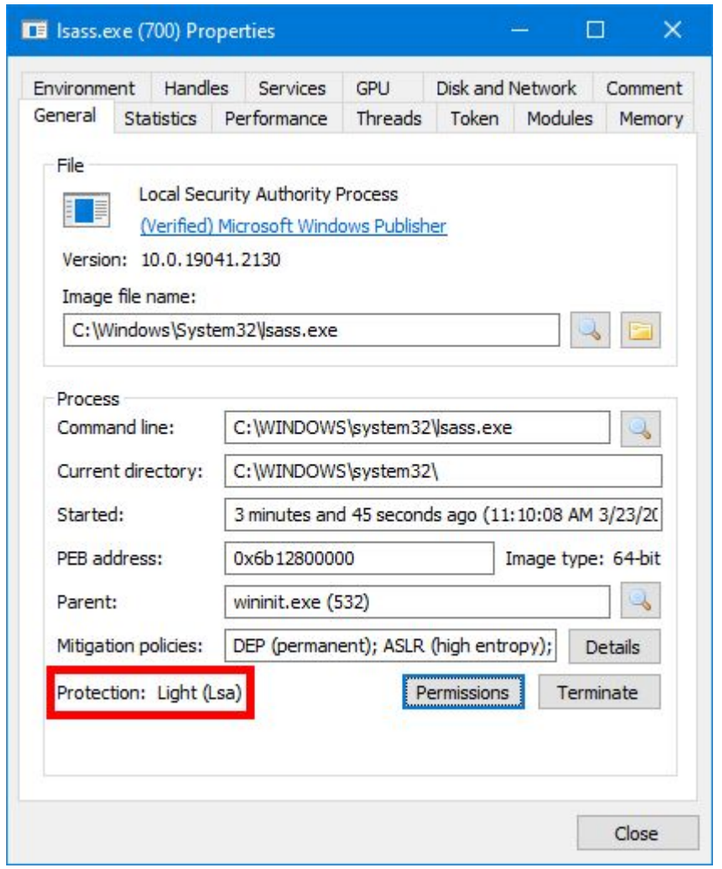
- Introduced in Windows 8
- Process hardened against code injection and memory tampering
- Created to isolate DRM processing from piracy tools with admin rights
- Will only load specially-signed code (EXEs/DLLs)
  - No DLL side-loading
- Handles are hardened:
  - No PROCESS_VM_WRITE, THREAD_SET_CONTEXT, etc
- Also protects System, Registry, and and System Guard Runtime processes



Protected Process Light (PPL)

- Introduced in Windows 8.1 as an extension of PP
- Similar signature requirements and process/thread HANDLE hardening
- Protect OS internals and AV from tampering
  - CSRSS - highly trusted by kernel
  - LSASS - credential dumping
  - SCM - service control manager
  - AntiMalware - prevent trivial termination of AV
- Later extended to prevent application tampering
  - Hyper-V Shielded VMs

- *The rest of this talk is about PPL*



PPL Implementation - EPROCESS

- Structure within kernel EPROCESS
- Assigned at process creation
- Protection type
  - None, Protected Process, or PPL
- Protection signer
  - See diagram

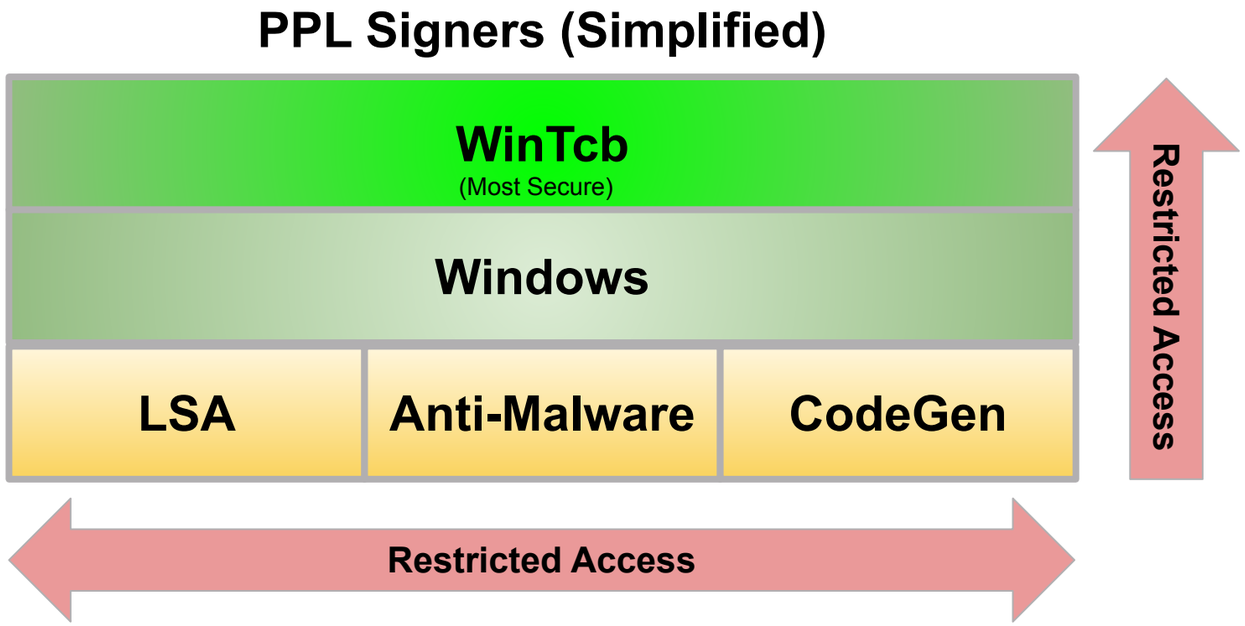
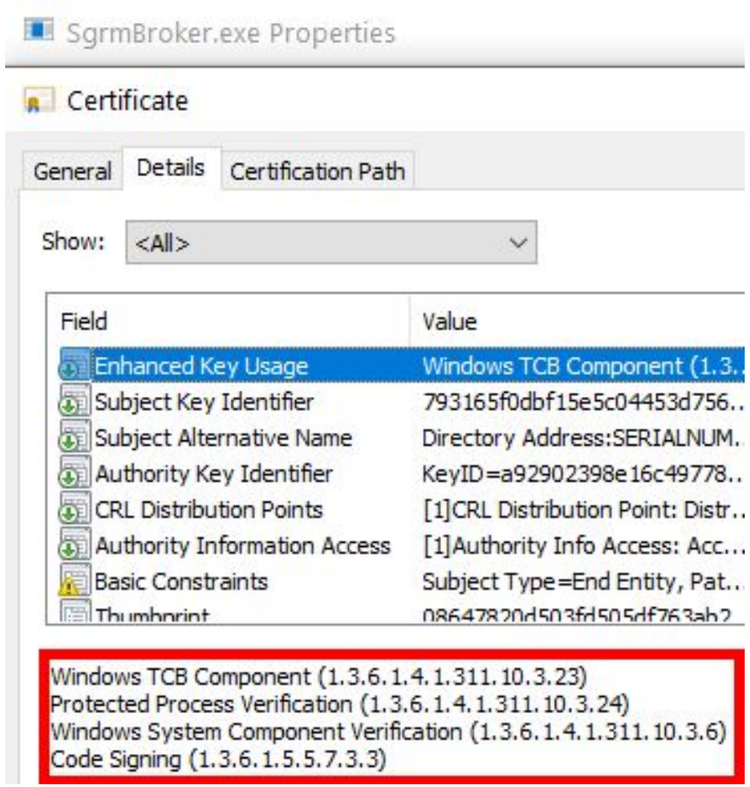
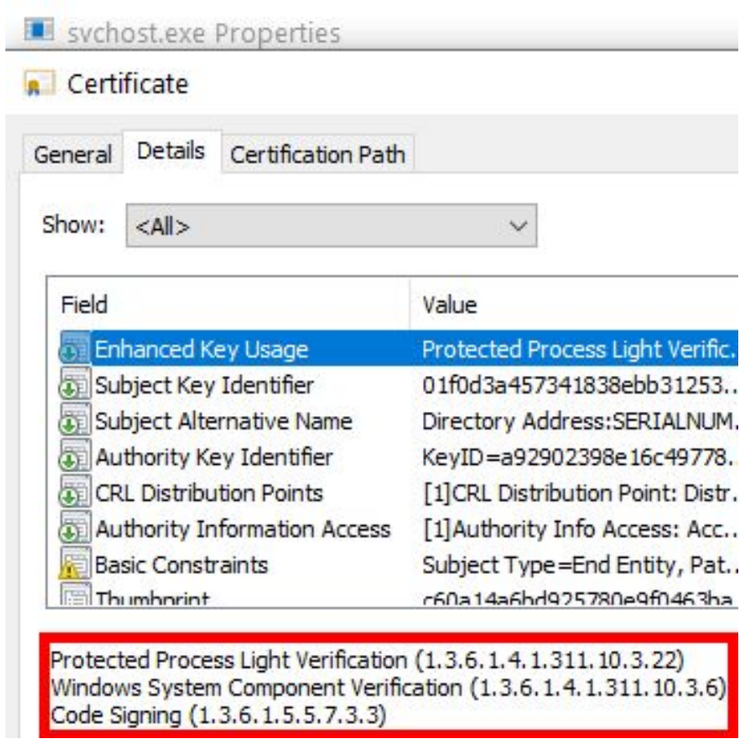
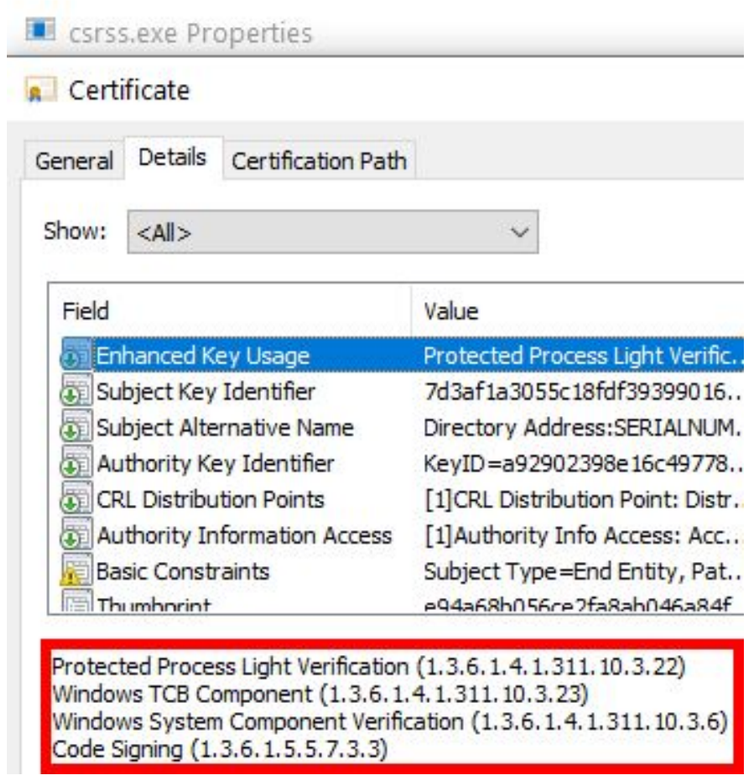


Diagram adapted from James Forshaw then updated: https://googleprojectzero.blogspot.com/2018/10/injecting-code-into-windows-protected.html
Thanks to @sixyvividtails for clarification: https://twitter.com/sixtyvividtails/status/1644098456951087104

Code Integrity - Signatures







PPL Implementation - EPROCESS

```
7: kd> dx -g @$cursession.Processes.Select(p => new {Name = p.Name, Type = p.KernelObject.
```

	Name	Type	Signer	SectionSignatureLevel
[0x4]	System	0x2	0x7	0xc
[0x8c]	Registry	0x2	0x7	0x0
[0x970]	SgrmBroker.exe	0x2	0x6	0x8
[0x1d8]	smss.exe	0x1	0x6	0x8
[0x22c]	csrss.exe	0x1	0x6	0x8
[0x278]	wininit.exe	0x1	0x6	0x8
[0x280]	csrss.exe	0x1	0x6	0x8
[0x2cc]	services.exe	0x1	0x6	0x8
[0xba8]	svchost.exe	0x1	0x5	0x8
[0x11bc]	svchost.exe	0x1	0x5	0x8
[0x21f8]	SecurityHealthService.exe	0x1	0x5	0x8
[0x21dc]	elastic-endpoint.exe	0x1	0x3	0x8
[0x3a0]	svchost.exe	0x0	0x0	0x0
[0x3c8]	fontdrvhost.exe	0x0	0x0	0x8

Processes and Thread Protection

- Process and Thread Hardening
  - Read/write access rights blocked to less-privileged callers
    - No PROCESS_TERMINATE, PROCESS_VM_WRITE, PROCESS_VM_READ, etc.
    - Checked in kernel by RtlTestProtectedAccess
    - No exceptions for SeDebugPrivilege
  - New limited-access rights
    - PROCESS_QUERY_**LIMITED**_INFORMATION, PROCESS_SET_**LIMITED**_INFORMATION
    - THREAD_QUERY_**LIMITED**_INFORMATION, THREAD_SET_**LIMITED**_INFORMATION

Processes and Thread Protection

```
PS C:\Windows\System32> Get-NtToken | Select User, IntegrityLevel

User                 IntegrityLevel
----                 --------------
NT AUTHORITY\SYSTEM          System


PS C:\Windows\System32> (Get-NtToken).Groups| Where {$_.Name -like "*TrustedInstaller"}

Name                         Attributes
----                         ----------
NT SERVICE\TrustedInstaller EnabledByDefault, Enabled, Owner


PS C:\Windows\System32> (Get-NtToken).Privileges | where {$_.Name -eq "SeDebugPrivilege"} | Select Name, Enabled

Name             Enabled
----             -------
SeDebugPrivilege    True


PS C:\Windows\System32> Get-NtProcess -Name services.exe -Access All
PS C:\Windows\System32> Get-NtProcess -Name services.exe -Access QueryLimitedInformation

Handle Name         NtTypeName Inherit ProtectFromClose
------ ----         ---------- ------- ----------------
3244   services.exe Process    False   False
```

Resource Protection

- Token Trust Level
  - New token attribute which indicates the trust level of the acting process or thread

```
PS C:\Windows\System32> $explorer = Get-NtProcess -Name explorer.exe -Access QueryLimitedInformation
PS C:\Windows\System32> (Get-NtToken -Process $explorer).TrustLevel
PS C:\Windows\System32>
PS C:\Windows\System32> $services = Get-NtProcess -Name services.exe -Access QueryLimitedInformation
PS C:\Windows\System32> (Get-NtToken -Process $services).TrustLevel

Name                               Sid
----                               ---
TRUST LEVEL\ProtectedLight-WinTcb S-1-19-512-8192
```

Resource Protection

- Trust Labels
  - New System Access Control List Entry (SACL ACE) type that allow trust level test for any securable object
  - Examples:
    - Protecting KnownDlls against modification by malicious administrators
    - Protect PPL process tokens against sandboxing by malicious administrators*

```
PS C:\Windows\System32> (Get-NtDirectory \KnownDlls).SecurityDescriptor.Sacl

Type              User                                Flags Mask
----              ----                                ----- ----
ProcessTrustLabel TRUST LEVEL\ProtectedLight-WinTcb None  00020003


PS C:\Windows\System32> $services = Get-NtProcess -Name services.exe -Access QueryLimitedInformation
PS C:\Windows\System32> (Get-NtToken -Process $services).SecurityDescriptor.Sacl | Where {$_.Type -eq "ProcessTrustLabel"}

Type              User                                Flags Mask
----              ----                                ----- ----
ProcessTrustLabel TRUST LEVEL\ProtectedLight-WinTcb None  0002001E
```

* Recent addition. See my work: https://www.elastic.co/security-labs/sandboxing-antimalware-products

Outline

- Introduction
  - What is a protected process?
  - Implementation
- **Attacks**
  - **Historical**
  - **Current**
- New Research
  - Novel Attack
  - Chaining Exploits
  - Mitigation

Attack: Cached Signing Level

- NtSetCachedSigningLevel race condition
- CI caches signing information for performance reasons
- Cache entries are automatically invalidated by NTFS if file is modified
- Race condition in CI allowed file to be modified before cache entry is finalized
- Fixed as CVE-2017-11830

Source: Unknown Known DLLs and other Code Integrity Trust Violations

Attack: Counterfeit \KnownDlls via Silos

- Windows containers (aka silos) are similar to docker containers.
- Containers created ability to “chroot” a process into a new object manager namespace
- “chroot” ability creates a unique namespace for all named objects including drives, network shares, events, mutexes, named pipes, etc
- \KnownDlls section object cache is part of the Object Manager namespace
  - Protected by trust label so this cannot normally be modified by attackers
- Windows treats \KnownDlls as verified - no additional checks before loading into PPL
- Attacker can create a counterfeit KnownDlls directory then spawn a new “chrooted” PPL, which will use their KnownDlls, loading DLLs specified therein
- Fixed in 7/2022 by removing KnownDlls support from PPL

Source: Unknown Known DLLs and other Code Integrity Trust Violations

Attack: Script Engine COM Hijack

- Some script interpreter DLLs will automatically load scripts specified in the registry
- Use DotNetToJScript to convert .NET payload to JS
- Find COM used by PPL, and hijack its registry run a script interpreter DLL instead
- Script interpreter loads attacker JS based on registry key, which loads .NET payload
- Fixed in 1803 by blocking script interpreters from loading into PPL
  - New function nt!CipMitigatePPLBypassThroughInterpreters blocks PPL from loading interpreter DLLs

Source: Unknown Known DLLs and other Code Integrity Trust Violations

Attack: Bring Your Own Vulnerable EXE

- Windows Error Reporting process memory dumper (WerFaultSecure) encrypts dumps to protect PP and PPL confidentiality
- Bug in Windows 8.1 build can lead to creation of unencrypted dumps
- Microsoft fixed the WerFaultSecure bug ~2014
- Latest Win11 will still run old vulnerable builds as WinTcb-Full
  - Easy RunAsPPL LSASS defeat

Source: http://publications.alex-ionescu.com/NoSuchCon/NoSuchCon%202014%20-%20Unreal%20Mode%20-%20Breaking%20Protected%20Processes.pdf

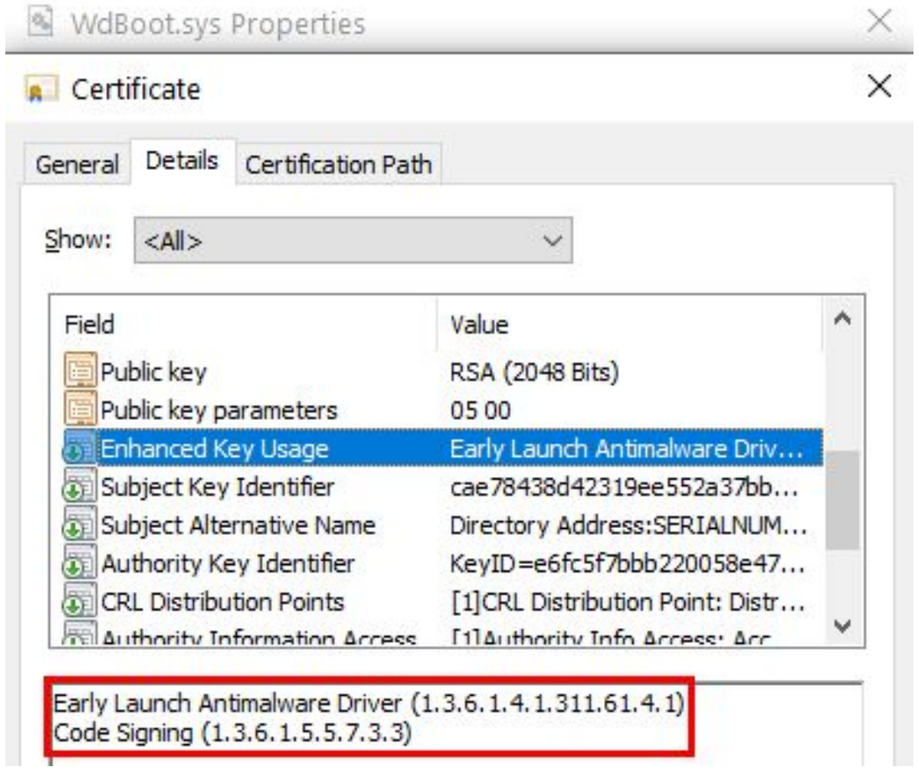
Attack: COM IRundown::DoCallback

- Use vulnerable Windows 8.1 WerFaultSecure to dump process and find secrets and addresses
- Use COM hijack to exploit undocumented COM feature: IRundown::DoCallback
- Use acquired secrets and addresses to call an arbitrary function within WerFault.exe
- Call existing code in process, achieving arbitrary write primitive
- Use arbitrary write primitive to overwrite LdrpKnownDllDirectoryHandle
- With counterfeit KnownDlls installed, attack proceeds like DefineDosDevice exploit

Source: https://googleprojectzero.blogspot.com/2018/11/injecting-code-into-windows-protected.html

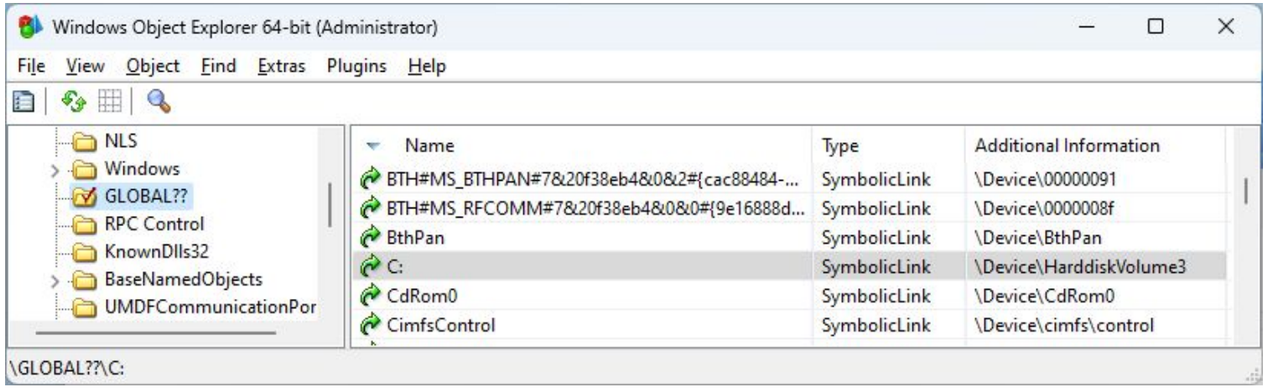
Attack: AntiMalware Blight

- ELAM - Early Launch AntiMalware Driver
  - Driver containing certificate hashes
  - Special signature from Microsoft
  - Any certificate listed in an ELAM driver can sign a file to run as AntiMalware-Light
- Overly-permissive ELAM
  - Some Antimalware vendors included hashes of certificates third-party certificates
  - Microsoft didn't vet certificate lists before signing ELAM drivers
- There are many overly-permissive ELAM drivers
  - Microsoft CAs included
- Example: You can run msbuild.exe as AntiMalware-Light with arbitrary parameters



Source: https://github.com/mattifestation/AntimalwareBlight

Attack: DefineDosDevice Bug



- The DefineDosDevice API defines, redefines, or deletes MS-DOS device names
- Implemented via RPC to WinTcb-PPL CSRSS
  - Remember this is the highest level of PPL
- TOCTOU enables attackers to trick CSRSS into creating entries in \KnownDlls
- Attacker can inject entries into KnownDlls, which PPL will load without verification
- Publicly documented in 2018 by James Forshaw
- Turnkey implementation released in April 2021 by Clément Labro as PPLdump
- Fixed in 7/2022 by removing KnownDlls support from PPL

Source: Unknown Known DLLs and other Code Integrity Trust Violations

Attack: COM Proxy Type Library Confusion

- .NET Runtime Optimization Service runs as CodeGen PPL and hosts COM service
- Modify COM proxy configuration for service to trigger type confusion
- Use type confusion to trigger arbitrary write, replacing KnownDlls handle with counterfeit directory that is pre-loaded with attacker's DLL
- With counterfeit KnownDlls installed, attack proceeds like DefineDosDevice exploit
- Leverage CodeGen PPL access to create a signing cache entry making any DLL as trusted so it can be side-loaded into WinTcb PPL (highest level)
- Variant implemented as turnkey PPLmedic tool in March 2023 by Clément Labro
- Microsoft: KnownDlls handle mitigation coming in June 2023

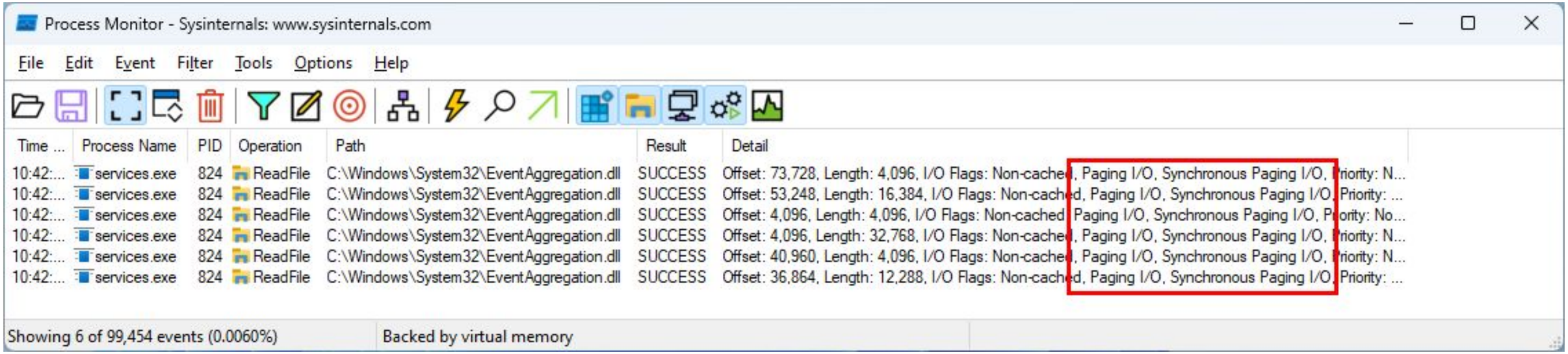
Source: https://googleprojectzero.blogspot.com/2018/10/injecting-code-into-windows-protected.html

Outline

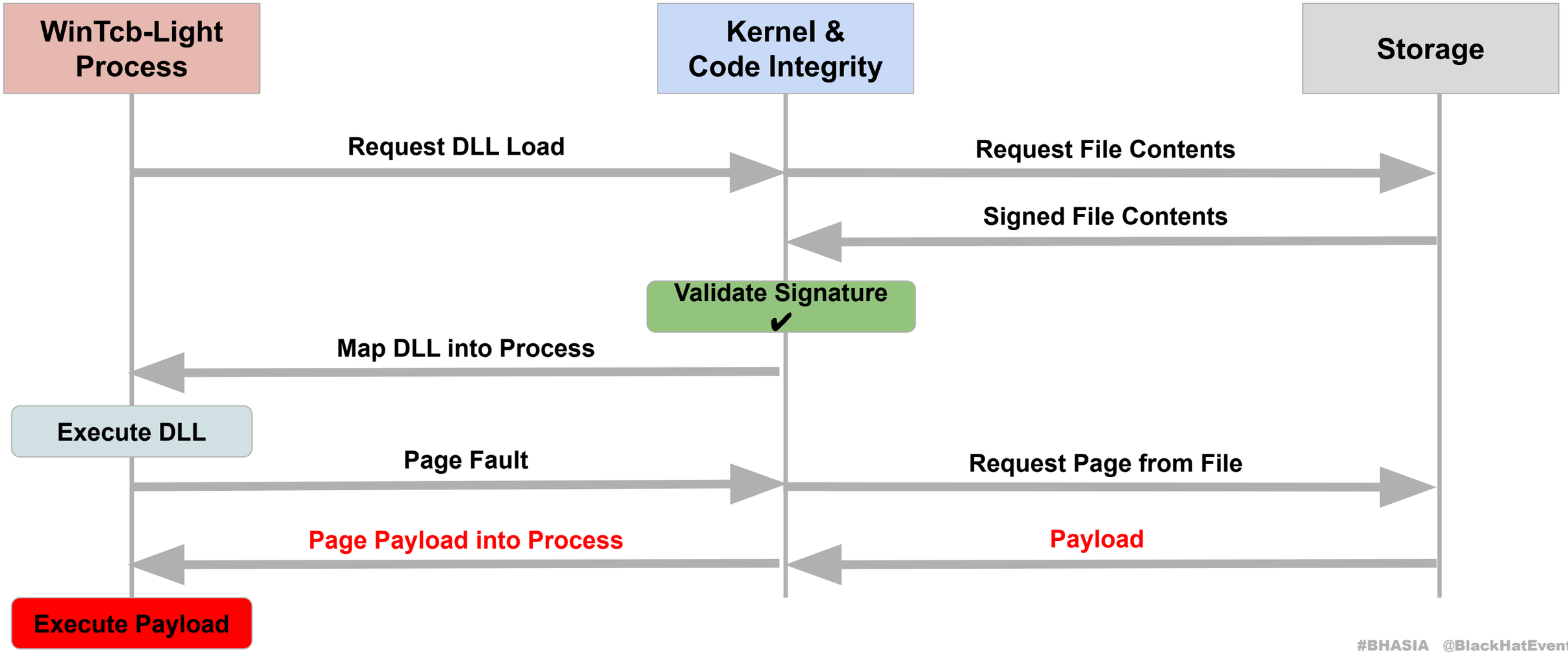
- Introduction
  - What is a protected process?
  - Implementation
- Attacks
  - Historical
  - Current
- **New Research**
  - **Novel Attack**
  - **Chaining Exploits**
  - **Mitigation**

Planning the Attack

- Attacks so far focus on:
  - CachedSigningLevel
  - KnownDlls
  - COM
- Let's try a different approach
  - Bait and Switch aka Time of Check, Time of Use (TOCTOU)



CI TOCTOU: Planning the Attack



CI TOCTOU: Page Hashes

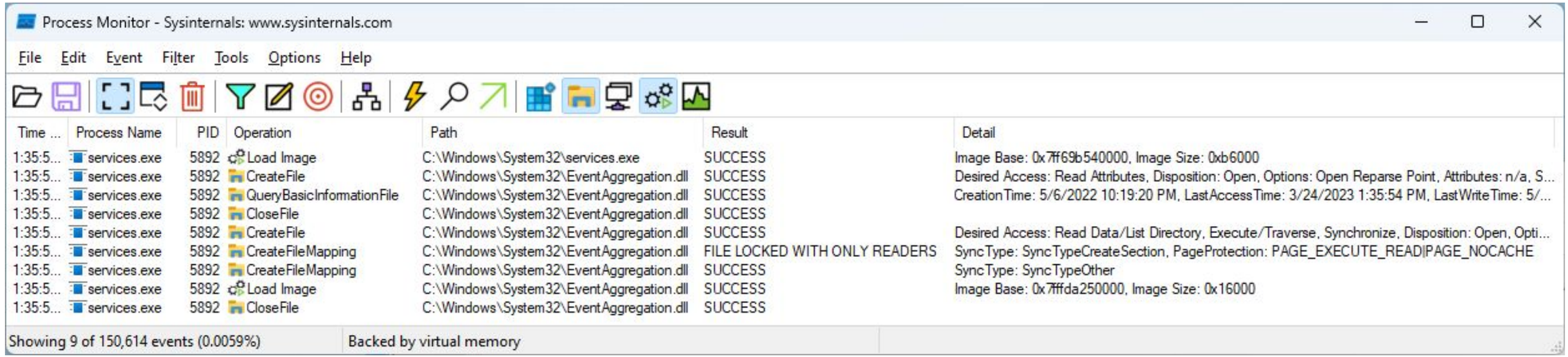
- Page hashes present in services.exe but not EventAggregation.dll

```
C:\Windows\System32>signtool verify /v /ph services.exe | grep -A10 "Page hash"
Page hashes:
0x00000000 973911F5DEABEFCF45A87E948DE1DF57DBE1C6C22D12559F2754862CEC5BB516
0x00000400 40048953BD60329AC1486A957A4EEC5D3A14ABC4E0E359BBAD063097495C3AB9
0x00001400 DA4D752F6C5EAA717CD127E8C4D4491F1D87CD2D73E2B7F38BC8A01336FE76E4
0x00002400 A8A85175F216A21BF270A65CCF26CD623E95FC88DA08FE8747606A16710A655F
0x00003400 A95303468AB638FD630C643265C819C14224805B954CAE98701D9428A2C6C1E9
0x00004400 161A709452170F6659EE9639432402D4E3454A31F5F0F5AAA3E3E29D9E5249C1
0x00005400 B8ADD3652917342812D22A573E75AFC85060321E30F15D7895058C2152847750
0x00006400 0755C750AD27D96B7F2D68D83216C4183B625E6CF768DF98F1C4F62F340D1D1C
0x00007400 A3A63DA8FB35B218BA9E3F116789D81D84CEB35B4F3FFE1D5A1003E5A9DA07AA
0x00008400 3A24070DDBE08C046B1A4BC2B183CF41D35EF623C0E0E2F3058E1EDE3BCD2C87

C:\Windows\System32>signtool verify /a /v /ph EventAggregation.dll | grep -A10 "Page hash"
SignTool Warning: No page hashes are present.
```

CI TOCTOU: Hunting for Local Paging

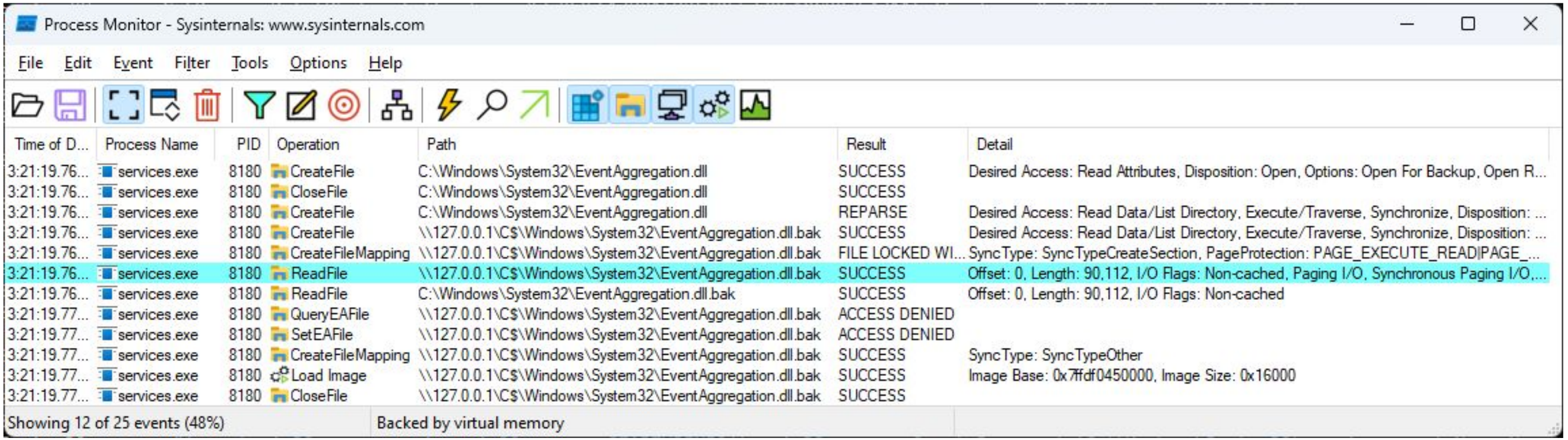
- Start simple - run services.exe as WinTcb-PPL
  - ❌ No file reads, and no paging I/O

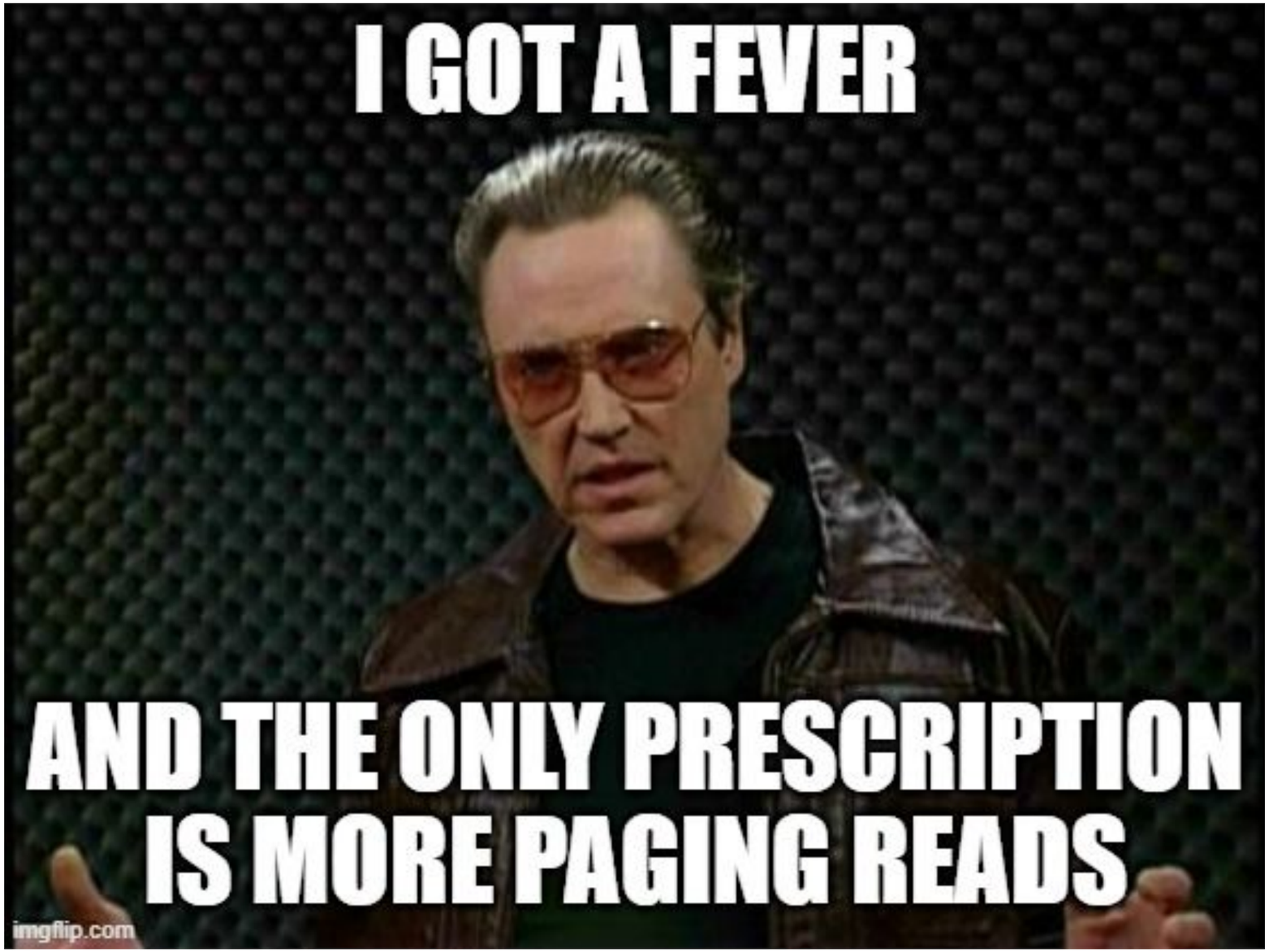



WE NEED TO GO
DEEPER

CI TOCTOU: Hunting for Remote Paging

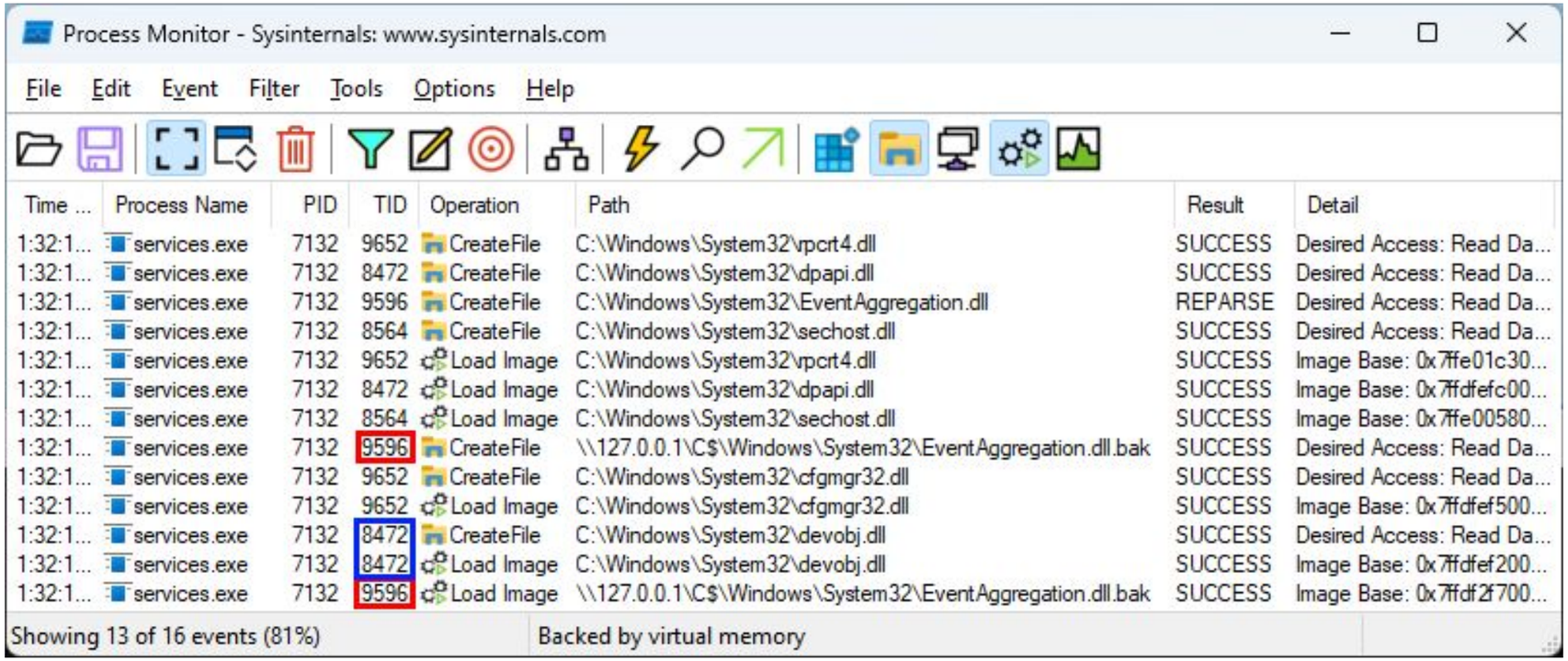
- What about SMB? Replace EventAggregation.dll with a symlink to loopback SMB
- We can see a paging operation over SMB




I GOT A FEVER
AND THE ONLY PRESCRIPTION
IS MORE PAGING READS
imgflip.com

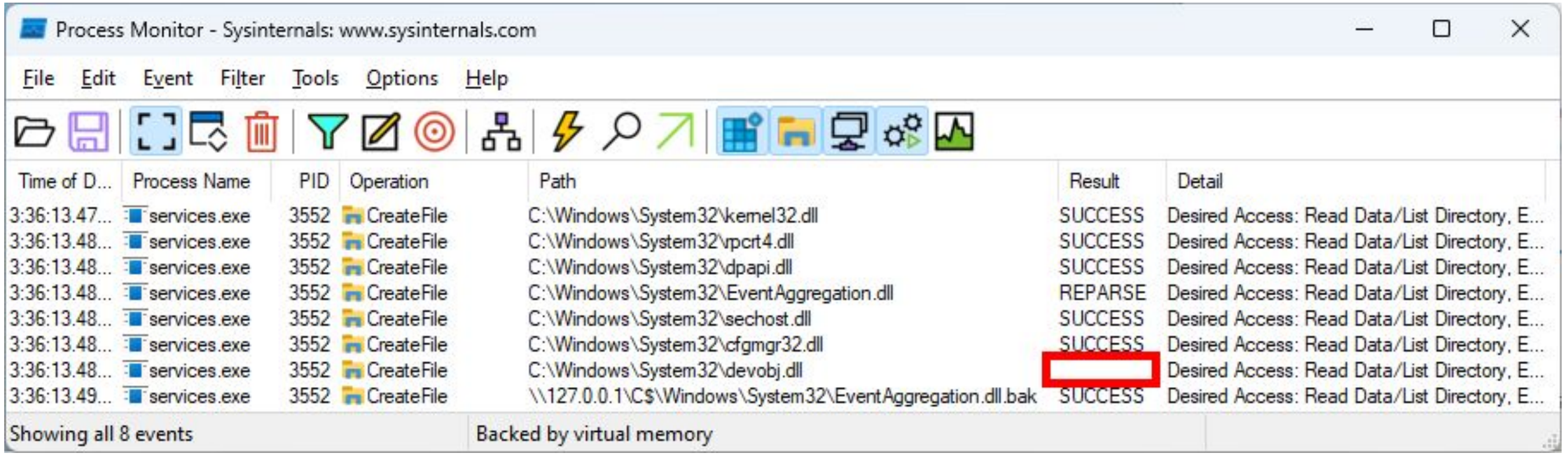
CI TOCTOU: Oplock Candidates

- Can we slow down process launch to allow time for paging?
- What about an opportunistic lock (oplock)?
  - Non-cooperative NTFS/SMB file locking mechanism
- Let's look for a CreateFile operation that we can interrupt



CI TOCTOU: Oplock Results

- Set an oplock on devobj.dll and launch services.exe
- IRP has no result - operation is still pending



CI TOCTOU: Oplock Results

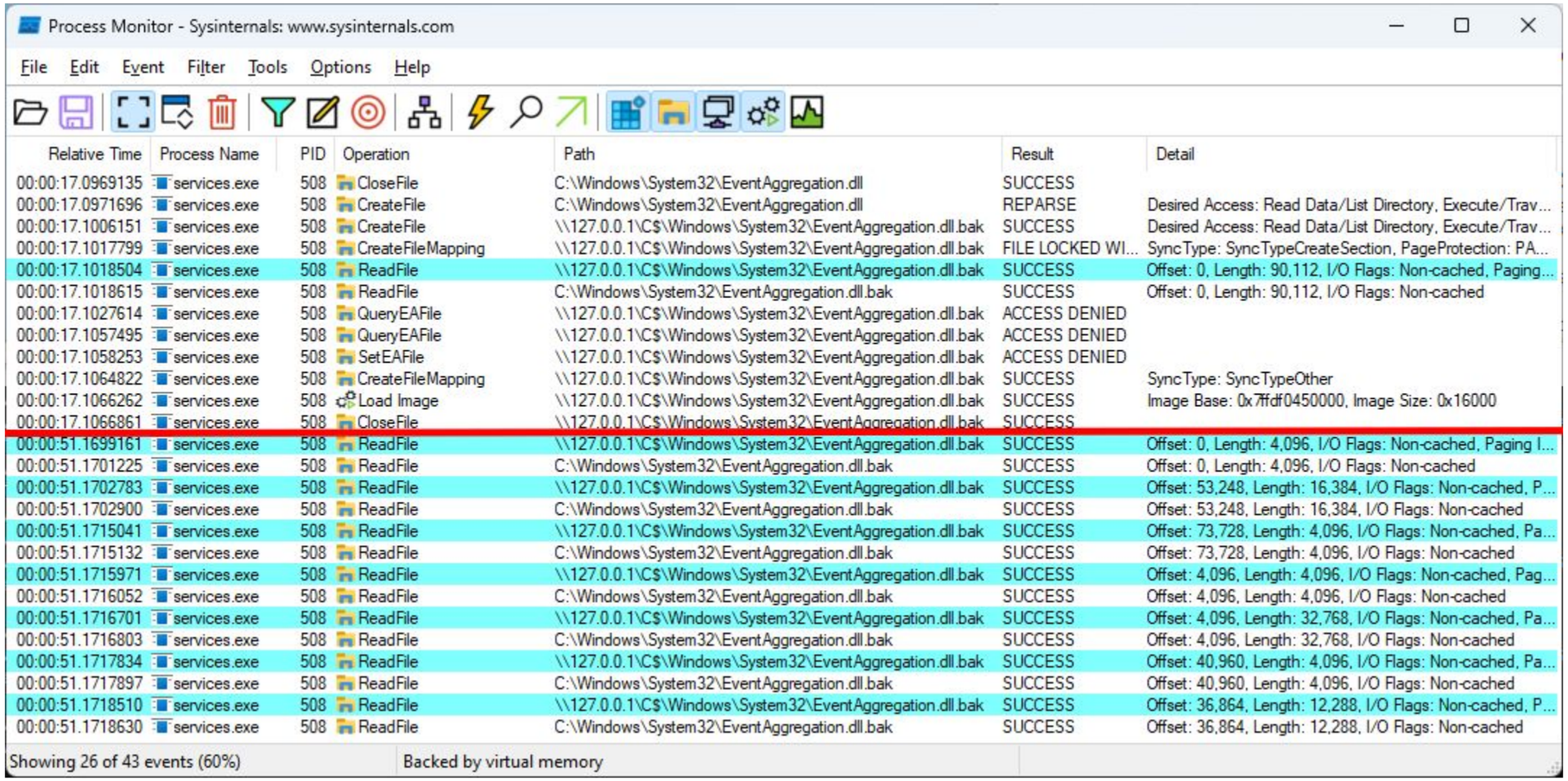
```
3: kd> k
  *** Stack trace for last set context - .thread/.cxr resets it
 # Child-SP          RetAddr               Call Site
00 ffffb88e`ac5be210 fffff807`7e4cb6c5     nt!KiSwapContext+0x76
01 ffffb88e`ac5be350 fffff807`7e4ccae7     nt!KiSwapThread+0xb05
02 ffffb88e`ac5be4a0 fffff807`7e4cf106     nt!KiCommitThreadWait+0x137
03 ffffb88e`ac5be550 fffff807`7e95be2c     nt!KeWaitForSingleObject+0x256
04 ffffb88e`ac5be8f0 fffff807`7e95bae7     nt!FsRtlCancellableWaitForMultipleObjects+0xcc
05 ffffb88e`ac5be960 fffff807`822c16c8     nt!FsRtlCancellableWaitForSingleObject+0x27
06 ffffb88e`ac5be9a0 fffff807`82256222     Ntfs!NtfsWaitForOplockCompletionEvent+0x24
07 ffffb88e`ac5be9e0 fffff807`7e4d00a5     Ntfs!NtfsFsdCreate+0x272
08 ffffb88e`ac5bec60 fffff807`813d9f5b     nt!IofCallDriver+0x55
09 ffffb88e`ac5beca0 fffff807`8140eff3     FLTMGR!FltpLegacyProcessingAfterPreCallbacksCompleted+0x15b
0a ffffb88e`ac5bed10 fffff807`7e4d00a5     FLTMGR!FltpCreate+0x323
0b ffffb88e`ac5bedc0 fffff807`7e8e2979     nt!IofCallDriver+0x55
0c ffffb88e`ac5bee00 fffff807`7e8de4f1     nt!IopParseDevice+0x8c9
0d ffffb88e`ac5befd0 fffff807`7e8dd4d2     nt!ObpLookupObjectName+0xae1
0e ffffb88e`ac5bf170 fffff807`7e8c1cf9     nt!ObOpenObjectByNameEx+0x1f2
0f ffffb88e`ac5bf2a0 fffff807`7e8bdfc8     nt!IopCreateFile+0x439
10 ffffb88e`ac5bf360 fffff807`7e63e1e8     nt!NtOpenFile+0x58
11 ffffb88e`ac5bf3f0 00007fff`dd26f2b4     nt!KiSystemServiceCopyEnd+0x28
12 00000083`4a77f0f8 00007fff`dd1e064c     ntdll!NtOpenFile+0x14
13 00000083`4a77f100 00007fff`dd1e0bb8     ntdll!LdrpMapDllNtFileName+0xe8
14 00000083`4a77f200 00007fff`dd1e0f80     ntdll!LdrpMapDllSearchPath+0x1d0
15 00000083`4a77f460 00007fff`dd1e0dbb     ntdll!LdrpProcessWork+0x148
16 00000083`4a77f4b0 00007fff`dd23236a     ntdll!LdrpWorkCallback+0xbb
17 00000083`4a77f4e0 00007fff`dd205976     ntdll!TppWorkpExecuteCallback+0x13a
18 00000083`4a77f530 00007fff`dcf626bd     ntdll!TppWorkerThread+0x8f6
19 00000083`4a77f810 00007fff`dd22a9f8     KERNEL32!BaseThreadInitThunk+0x1d
1a 00000083`4a77f840 00000000`00000000     ntdll!RtlUserThreadStart+0x28
```

CI TOCTOU: Forcing Paging

- Where do we go from here?
- We have a frozen WinTcb PPL process. We want it to page-in code over the network.
- Can we page it out using EmptyWorkingSet?
  - ❌ Requires PROCESS_SET_QUOTA, which we can't get
- What about paging out the whole OS?
  - Empty system working set and standby lists
    - NtSetSystemInformation(SystemMemoryListInformation)*
    - Requires SeProfileSingleProcessPrivilege, which Admins have

* https://github.com/elastic/Silhouette/blob/main/2023-01%20Silhouette%20Shmoocon%20Presentation.pdf

CI TOCTOU: Paged Reads



CI TOCTOU: Delivering the Payload

- Now that we can reliably force page faults, let's try to inject some code
  a. Disable the local SMB server (LanManServer service) and reboot
  b. Run local SMB server that serves two versions of EventAggregation.dll
     - First, serve original DLL for CI verification
     - Later, patch in special sauce over DllMain for subsequent requests

```
# This payload requires a kernel debugger to view
# If you use this payload, type this in WinDbg afterwards:
# db @rip; dx @$curprocess->Name; dx @$curprocess->KernelObject->Protection
$Payload = "CC" + ("90" * 15) + ("CAFEC0DE" * 64)
```

CI TOCTOU: Code Execution

```
Break instruction exception - code 80000003 (first chance)
0033:00007fff`addb1550 cc              int     3
5: kd> db @rip
00007fff`addb1550  cc 90 90 90 90 90 90 90-90 90 90 90 90 90 90 90  ................
00007fff`addb1560  ca fe c0 de ca fe c0 de-ca fe c0 de ca fe c0 de  ................
00007fff`addb1570  ca fe c0 de ca fe c0 de-ca fe c0 de ca fe c0 de  ................
00007fff`addb1580  ca fe c0 de ca fe c0 de-ca fe c0 de ca fe c0 de  ................
00007fff`addb1590  ca fe c0 de ca fe c0 de-ca fe c0 de ca fe c0 de  ................
00007fff`addb15a0  ca fe c0 de ca fe c0 de-ca fe c0 de ca fe c0 de  ................
00007fff`addb15b0  ca fe c0 de ca fe c0 de-ca fe c0 de ca fe c0 de  ................
00007fff`addb15c0  ca fe c0 de ca fe c0 de-ca fe c0 de ca fe c0 de  ................
5: kd> dx @$curprocess->Name
@$curprocess->Name : services.exe
    Length           : 0xc
5: kd> dx @$curprocess->KernelObject->Protection
@$curprocess->KernelObject->Protection                 [Type: _PS_PROTECTION]
    [+0x000] Level            : 0x61 [Type: unsigned char]
    [+0x000 ( 2: 0)] Type             : 0x1 [Type: unsigned char]
    [+0x000 ( 3: 3)] Audit            : 0x0 [Type: unsigned char]
    [+0x000 ( 7: 4)] Signer           : 0x6 [Type: unsigned char]
```

CI TOCTOU: Removing the Reboot

- LanManServer configuration change is noisy. Can we remove the reboot?
  a. ❌ SMB - port fixed. LanManServer takes it early in boot. No way to release it
  b. ❌ WebDAV - file is read once at mapping and cached locally
- Cloud Filter API
  a. Available by default in Client SKUs of 1709+
  b. Create small/empty placeholder files marked with reparse tags
  c. When read requests come, minifilter drive detects reparse tags and calls UM callback to request data
  d. UM callback provides the requested file contents
     - You decide what bytes to serve to the client in your rehydration callback
  e. Simple-to-use usermode API
     - No COM 🥂
  f. No special signing requirements
  g. James Forshaws provided working sample code.

https://googleprojectzero.blogspot.com/2021/01/windows-exploitation-tricks-trapping.html

CI TOCOTU: Putting it All Together

- Final attack flow:
  a. Use CloudFilter to create an empty placeholder file with a callback function we control
  b. Redirect EventAggregation.dll to our placeholder through loopback SMB via symbolic link
  c. Set oplock on devobj.dll to interrupt process initialization
  d. Run the target PPL
  e. The target PPL attempts to load EventAggregation.dll
  f. CloudFilter callback fires to rehydrate placeholder
     - Serve up original EventAggregation.dll for CI verification
     - Page everything out by emptying working set and standby lists
     - Release oplock
  g. The PPL resumes and leads to paging reads over SMB, which are forwarded to the placeholder
  h. CloudFilter callback fires to rehydrate placeholder
     - Serve up patched copy of EventAggregation.dll
  i. The PPL executes our PIC payload inside services.exe as WinTcb-Light, which dumps the process of your choosing

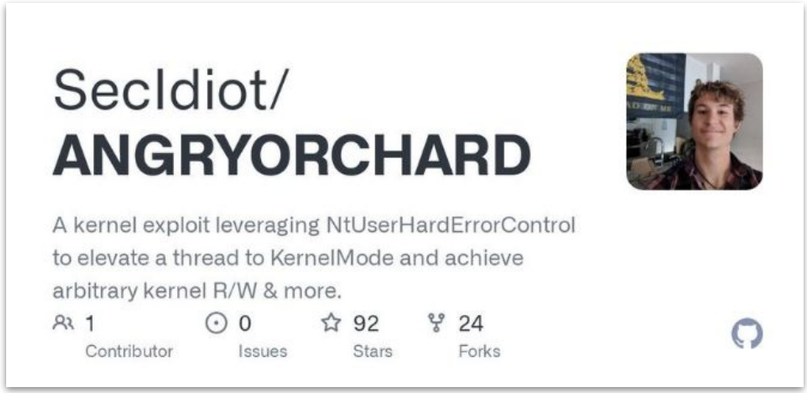
- This is PPLFault

PPLFault: DEMO

- DEMO

Why Stop at LSASS? ANGRYORCHARD

- Released in July 2022 by Austin Hudson when Microsoft patched PPLdump
- Exploits PPLdump bug to achieve code execution in CSRSS (WinTcb PPL)
- Exploits bug in NtUserHardErrorControl to perform arbitrary kernel decrement
  a. Only exploitable within CSRSS
- Decrement KTHREAD.PreviousMode from UserMode (1) to KernelMode (0)
  a. KernelMode disables most memory and security access checks on the system
  b. GodMode - syscalls treat you like a kernel worker thread
  c. Examples:
     - hSystemProcess = OpenProcess(4, PROCESS_ALL_ACCESS)
     - WriteProcessMemory(SomeKernelAddress)
     - NtOpenSection(\Device\PhysicalMemory, SECTION_ALL_ACCESS)



Exploit Chain Demo - GodFault

- DEMO

Mitigations - Windows

- Root of problem is a TOCTOU where signature validation is decoupled from paging
- If only Windows had some way to validate the hashes of pages…

```
C:\Windows\System32>signtool verify /v /ph "C:\Windows\System32\ntdll.dll" | grep -A10 "Page hash"
Page hashes:
0x00000000 63A2FF4AF0FE4AB373879E79C5B6AD71F87921D7785A76DCDA9EA7251D6A5CEB
0x00000400 387F45BEE453C35ED971806041D0A9D71A30DFA5590E05435ABB2D099849C64B
0x00001400 441CDBF430CAFB55AEAB82A0767D81422145245E772FAE5855777F52D5E0D20D
0x00002400 8381A86212C8DDC6048C49523DFDA8416169FFF7BFD141A58FFBAB4A8296BFC9
0x00003400 3A55CF1DE8C04DAAF6DA6D4216C020A9766D5E2EE346A00019B7D2B84BA6FDF4
0x00004400 D0BC1D0FE6C5C4B0206ED7E34107BA16D75C21D884E00410B27FBCC94D68AF3A
0x00005400 BD6AAF7C53EFEB822A16A016C915D907E9A7C4A9A45FB9AF461A9F05D059E365
0x00006400 A620F0E12B712862CCC7BED40134A3978169F523EA52C36424E3BC52BC6EAF57
0x00007400 16C9D10265D816C9EA3790BEA07C3A58DCAE6164ABC1794EF0471D5268805647
0x00008400 D2F08CE6751388ED85E2453994C01392558DDAD7E496D0C5B0D941BBBB36FE4F
```

Mitigations - AV Vendors

- AntiMalware vendors can’t
  a. Modify the memory manager to require page hashes for all images loaded into PPL
  b. Re-sign Microsoft binaries with PPL certs to add page hashes

- AntiMalware vendors can still break the PPLFault exploit chain

Mitigation - NoFault

- NoRemoteImages
  a. Exploit mitigation to prevent loading of DLLs from network locations (SMB, WebDAV, etc)
  b. Originally introduced with EMET. Later integrated directly into Windows
- Set-ProcessMitigation PowerShell cmdlet
  a. Persists key in registry
  b. Useless against attacker who controls registry
- NoFault.sys
  a. Enables NoRemoteImages policy early in process lifecycle via process creation callback

https://learn.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-process_mitigation_image_load_policy
https://learn.microsoft.com/en-us/powershell/module/processmitigations/set-processmitigation

NoFault - DEMO

- DEMO

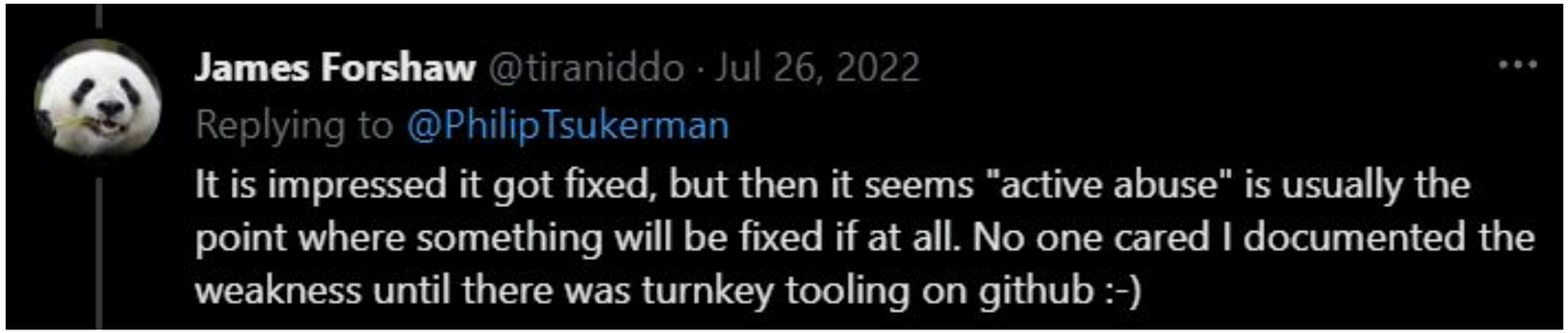
Disclosure Timeline

- Timeline
  - 2022-09-22 Reported PPLFault and GodFault to MSRC as VULN-074311
  - 2022-10-21 MSRC case closed without action
  - 2023-02-28 I publicly announced this BlackHat talk on Twitter
  - 2023-03-01 Windows Defender team reached out to me via Twitter

- Exploits still functional against:
  - Windows 11 22H2 22621.1702 (May 2023)
  - Windows 11 Insider Canary 25346.1001 (April 2023)

Conclusions / Black Hat Sound Bytes

- Defending against administrators is hard
  - Lots of power and attack surface
- Little things add up
  - Non-Elevated => Admin (UAC bypass) is not a security boundary
  - Admin => PPL is not a security boundary
  - PPL => Kernel RW is not a security boundary
  - Transitively: Non-Elevated => Kernel RW is not a security boundary
- When MSRC doesn't care, the Defender team still might
- Public tooling get bugs fixed
  - It required "active abuse" to force Microsoft's hand on the DefineDosDevice vulnerability



Source: https://twitter.com/tiraniddo/status/1551966781761146880

Conclusions: Patching

ppp

Available for: iPhone 8 and later, iPad Pro (all models), iPad Air 3rd generation and later, iPad 5th generation and later, iPad mini 5th generation and later

Impact: An app with root privileges may be able to execute arbitrary code with kernel privileges

Description: A use after free issue was addressed with improved memory management.

CVE-2022-42829: an anonymous researcher

ppp

Available for: iPhone 8 and later, iPad Pro (all models), iPad Air 3rd generation and later, iPad 5th generation and later, iPad mini 5th generation and later

Impact: An app with root privileges may be able to execute arbitrary code with kernel privileges

Description: The issue was addressed with improved memory handling.

CVE-2022-42830: an anonymous researcher

ppp

Available for: iPhone 8 and later, iPad Pro (all models), iPad Air 3rd generation and later, iPad 5th generation and later, iPad mini 5th generation and later

Impact: An app with root privileges may be able to execute arbitrary code with kernel privileges

Description: A race condition was addressed with improved locking.

CVE-2022-42831: an anonymous researcher

CVE-2022-42832: an anonymous researcher

Questions?

- Gabriel Landau at Elastic Security Labs
- Twitter: @GabrielLandau

- PoC code: https://github.com/gabriellandau/PPLFault

elastic security labs