

ropkit

Your device ran into a problem and needs to restart.

100% complete

Nathan Sawyer

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

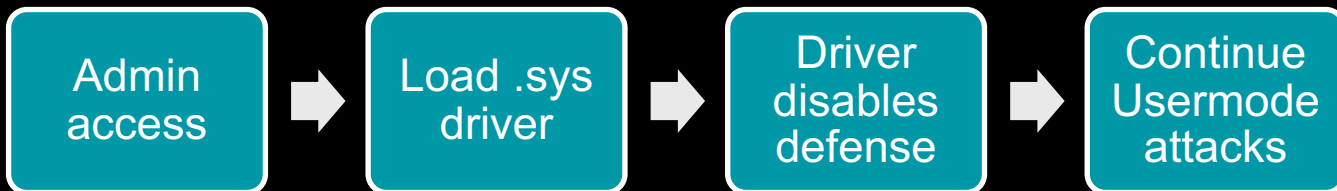
Challenge

- We want kernel code execution
- Windows Hypervisor blocks this
- BYOVD attacks only work for one driver
- No free public framework for BYOVD yet!
(sort of)
- HVCI blocks all (HV code integrity)
- Shadow Stacks block code execution

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

What is BYOVD?

- Bring your own vulnerable driver!
- Used for persistence
- EDR killing/muting
- Password dumping
- General mayhem



Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Solution

- Framework for BYOVD attacks
- You provide the driver and the library, I provide a large framework
- Many built in attacks!
- With HVCI, the question is:

How much of Windows can we change without editing executable code?

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

What is HVCI?

- Hypervisor Code Integrity
- Part of VBS (virtualization based security)
- Blocks almost all kernel code execution
- Prevents RWX memory in the kernel
- Ensures that we cannot execute our own code within the kernel

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Background

- Juan Sacco has already created a similar framework
- Then took it all off Github and is selling it
- Fundamentally different framework
- Luckily scammers put back some libraries, with virus

ntoskrnlwalker (Private)

this is a virus do not build!

• C++ Updated 4 minutes ago

NTKernelWalkerLib (Private)

clone of ntkernelwalkerlib from jsacco

• C++ Updated 7 minutes ago

DataOnlyGadget (Private)

clone of dog from jsacco

• C++ Updated 17 minutes ago

ropkit (Private)

General rootkit for a virtual kernel r/w prim

• C++ MIT License Updated 20 hours ago

April 17, 2026



jsacco 4/17/26, 12:41 AM

Funny how he copied my project and gave no credits
I have everything on my PC. But not planning to share it on Github



nathan 4/17/26, 12:46 AM



jsacco 4/23/26, 1:18 AM

That's great! Yes the r/w shouldn't matter
The physical r/w is translated into VA in any case
If you are interested in learning more check out my training!



Kernel Pack

€1.200 EUR

Kernel Pack is the game-over tool to obtain ring-0 access without requiring you to design, build, deploy, and use a full-featured graphical C2 interface under VBS, HVCI, and KCET.

Important: Access Subject to V
Due to the nature of our tools, access is provided after successful completion of a security audit.

Single-user, single machine. Each license is the email used at purchase; one license per email.

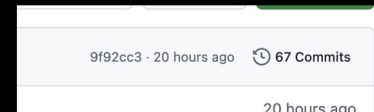
Regularly Special pricing to our

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Differences

- Framework for virtual r/w prim, not just physical
- Not just data only attacks!
- I include different code execution methods
- Free & open source

ropkit was born



Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

How To use:

- You provide the virtual r/w primitive
- The framework handles the attacks and offsets
- You possibly write your own attack
- If you cannot leak kernel memory in your driver, this must run as admin. If you can, launch at medium integrity!
- Two methods of automatic offset determination working on 25H2

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Features

- Hiding a process completely
- Privilege escalation to system
- Protected Process Light (PPL) removal
- Keylogging
- Code Execution!!!
- Dynamic Offsets!

```
KDTARGET: Refreshing KD connection

*** Fatal System Error: 0x0000003b
(0x00000000C0000005,0xFFFFF807E2621C97,0xFFFFFC609EB65FCE0,0x0000000000000000)

Break instruction exception - code 80000003 (first chance)

A fatal system error has occurred.
Debugger entered on first try; Bugcheck callbacks have not been invoked.

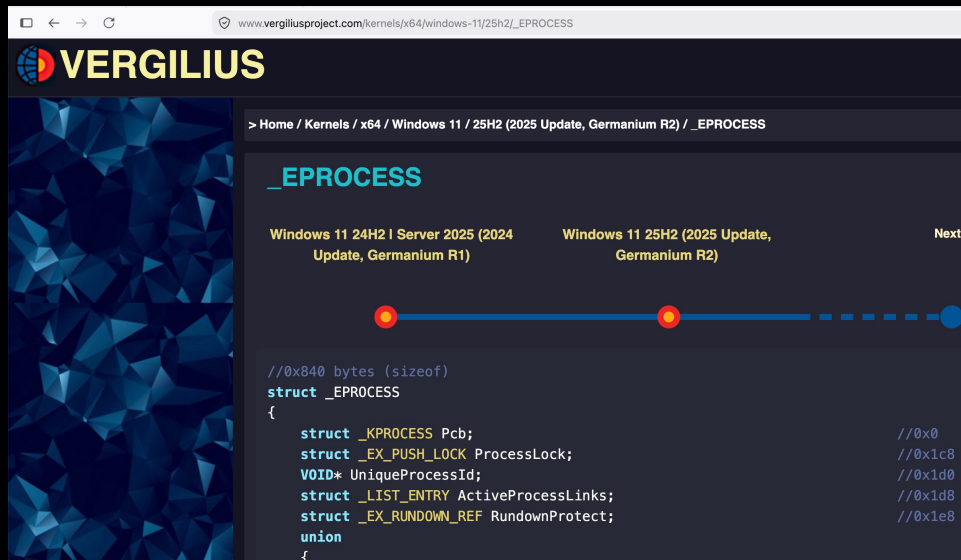
A fatal system error has occurred.

nt!DbgBreakPointWithStatus:
fffff804'de98b1b0 cc          int     3
```

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

How is this possible? Data first

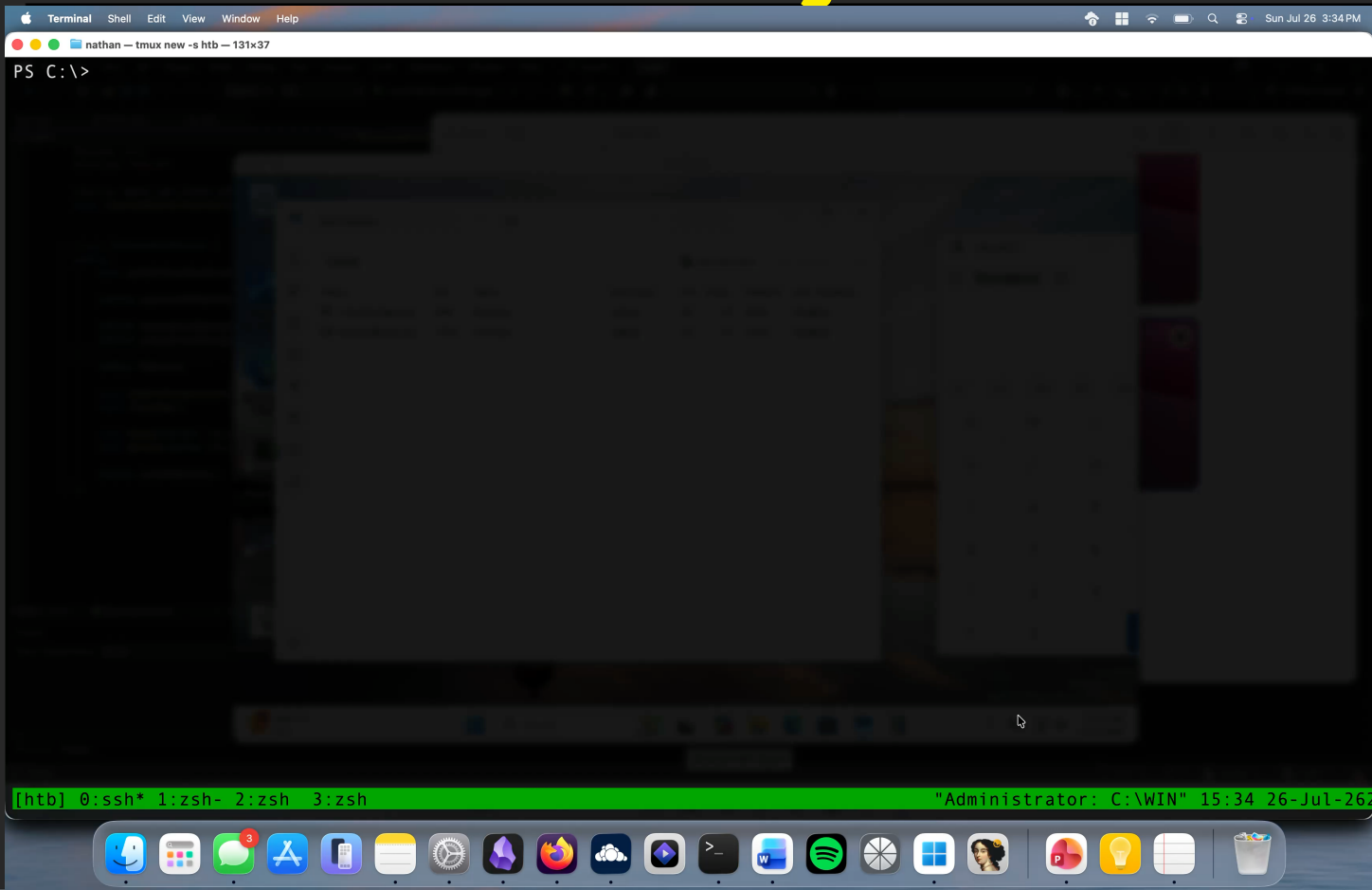
- Find and edit EPROCESS structure
- Holds process information , like protected process light (PPL) status
- Not novel



The screenshot shows the Vergilius website interface. The browser address bar displays the URL: `www.vergiliusproject.com/kernels/x64/windows-11/25h2/_EPROCESS`. The page title is "VERGIILIUS". The breadcrumb navigation path is: `> Home / Kernels / x64 / Windows 11 / 25H2 (2025 Update, Germanium R2) / _EPROCESS`. The main content area is titled `_EPROCESS` and features a timeline diagram with three points: "Windows 11 24H2 | Server 2025 (2024 Update, Germanium R1)", "Windows 11 25H2 (2025 Update, Germanium R2)", and "Next". Below the timeline, the C structure definition for `_EPROCESS` is shown, starting with `//0x840 bytes (sizeof)` and `struct _EPROCESS`. The structure includes fields for `_KPROCESS Pcb`, `_EX_PUSH_LOCK ProcessLock`, `VOID* UniqueProcessId`, `_LIST_ENTRY ActiveProcessLinks`, `_EX_RUNDOWN_REF RundownProtect`, and a `union` block.

```
//0x840 bytes (sizeof)
struct _EPROCESS
{
    struct _KPROCESS Pcb; //0x0
    struct _EX_PUSH_LOCK ProcessLock; //0x1c8
    VOID* UniqueProcessId; //0x1d0
    struct _LIST_ENTRY ActiveProcessLinks; //0x1d8
    struct _EX_RUNDOWN_REF RundownProtect; //0x1e8
    union
    {
```

Privesc and Hiding Demo!



Protected Process Light Demo

The screenshot shows a QEMU virtual machine running Windows 11. The System Informer tool is open, displaying the list of processes. The `lsass.exe` process is highlighted, showing it is a Protected Process (Lsa). The process details are as follows:

Name	PID	CPU	I/O total r...	Private b...	User name	Description	Protection
lsass.exe	996	0.01		9.95 MB	NT AUTHORITY\SYSTEM	Local Security Authority Process	Light (Lsa)

The background shows the C++ source code for the `VulnerableDriver` in the `ropkit` project. The code includes headers for `kernel.h` and `ioctl.h`, and defines a `VulnerableDriver` class with methods for handling I/O requests and enabling/disabling primitives.

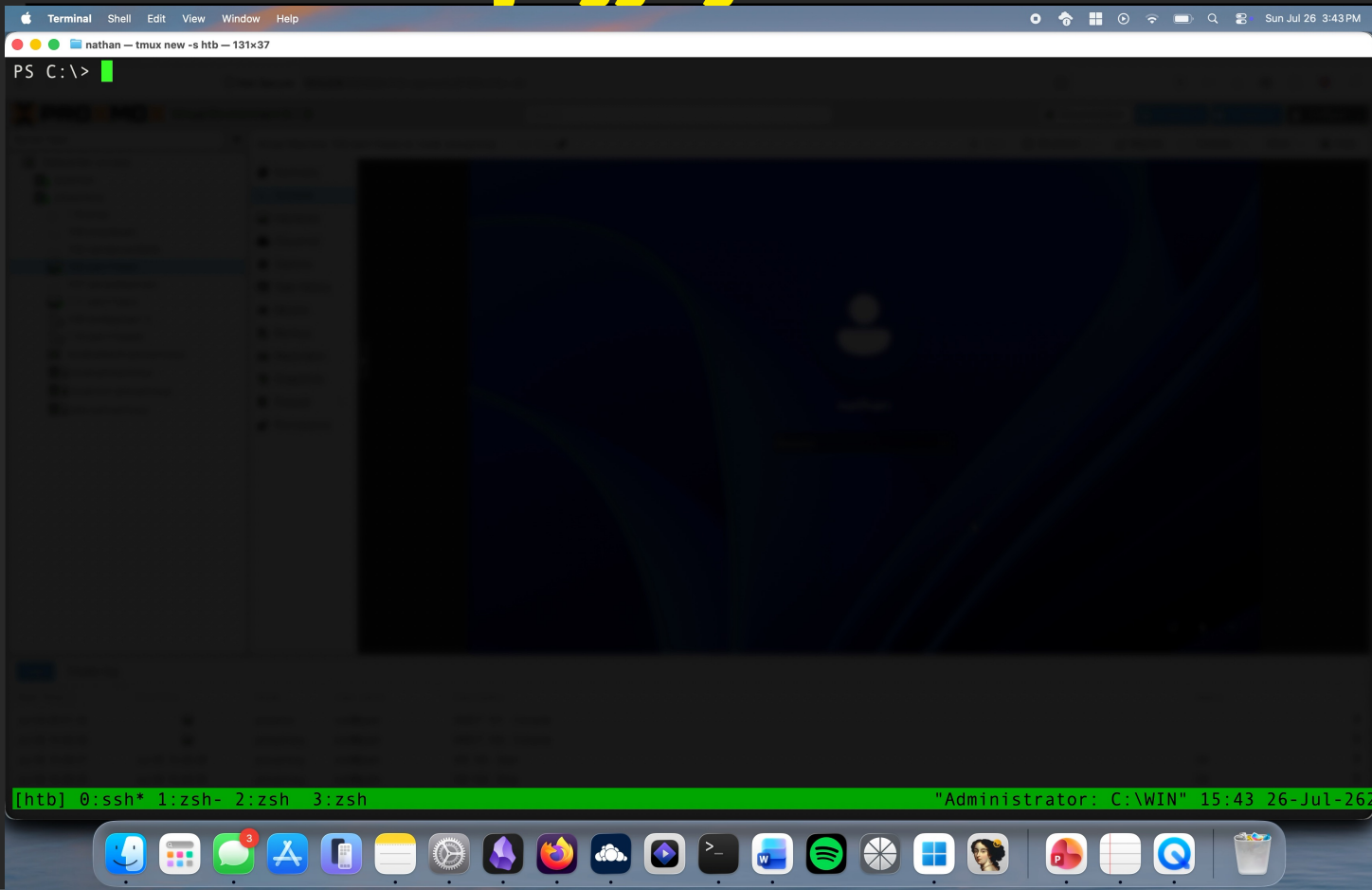
The bottom of the screen shows the Windows taskbar with various applications and the system tray displaying the date and time as 12:50 PM on 7/26/2026.

Keylogging!

- We have access to reading kernel memory, so let's view driver memory
- We can view keystrokes as a result!

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Keylogging Demo



Code Execution Methods

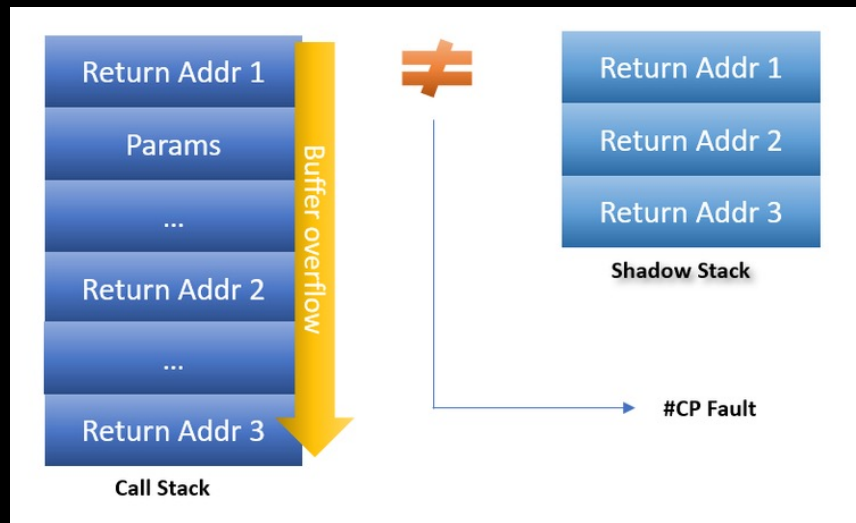
Method	Blocked by	Requires
Rop chain	Shadow stacks	N/A
Callback hijack	N/A with DKOM, otherwise HLAT/Intel VT-RP	N/A in my implementation, otherwise Driver without CFG
COP/JOP chain	N/A	Driver without CFG
SSDT hijack	HLAT/Intel VT-RP	N/A
Trap Frame Edit	N/A	N/A

- **Implemented ROP chain: this is Connor McGarr's strategy**
 - **Callback Hijack: Amazing, completed by Worawit, but not implemented**
 - **JOP: Implemented JOP chain**
 - **SSDT hijack: Blocked by KDP, bypassed by remap**
 - **Trap Frame Edit: FUN!!**
- Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)

What failed: VulnerableDriver.sys

Shadow Stack

- Compares Stack to a saved copy, crashes if they do not match
- Compares on a return
- Return addresses are pushed on a call instruction
- Not commonly enabled in computers still. Intel 11th gen or AMD Zen 3



JOP chain

- Shadow stacks block overwriting stack, KDP and HVCI block forward attacks, so how can you run shellcode?
- Start a JOP chain!

General plan:

- Use IRP hooking to overwrite Dispatch Table of a driver without Control Flow Guard to jump to a separate driver with gadgets!

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

JOP chain

- Every instruction must end in `JMP qword [rax+offset]`
- Load a second Nvidia driver for more gadgets
- Utilized gadgets to suspend a process
- Can easily change this!

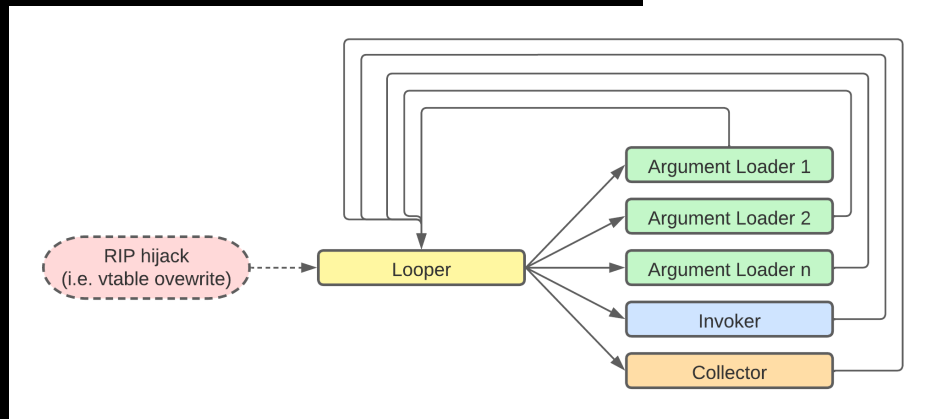


Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

JOP Gadgets

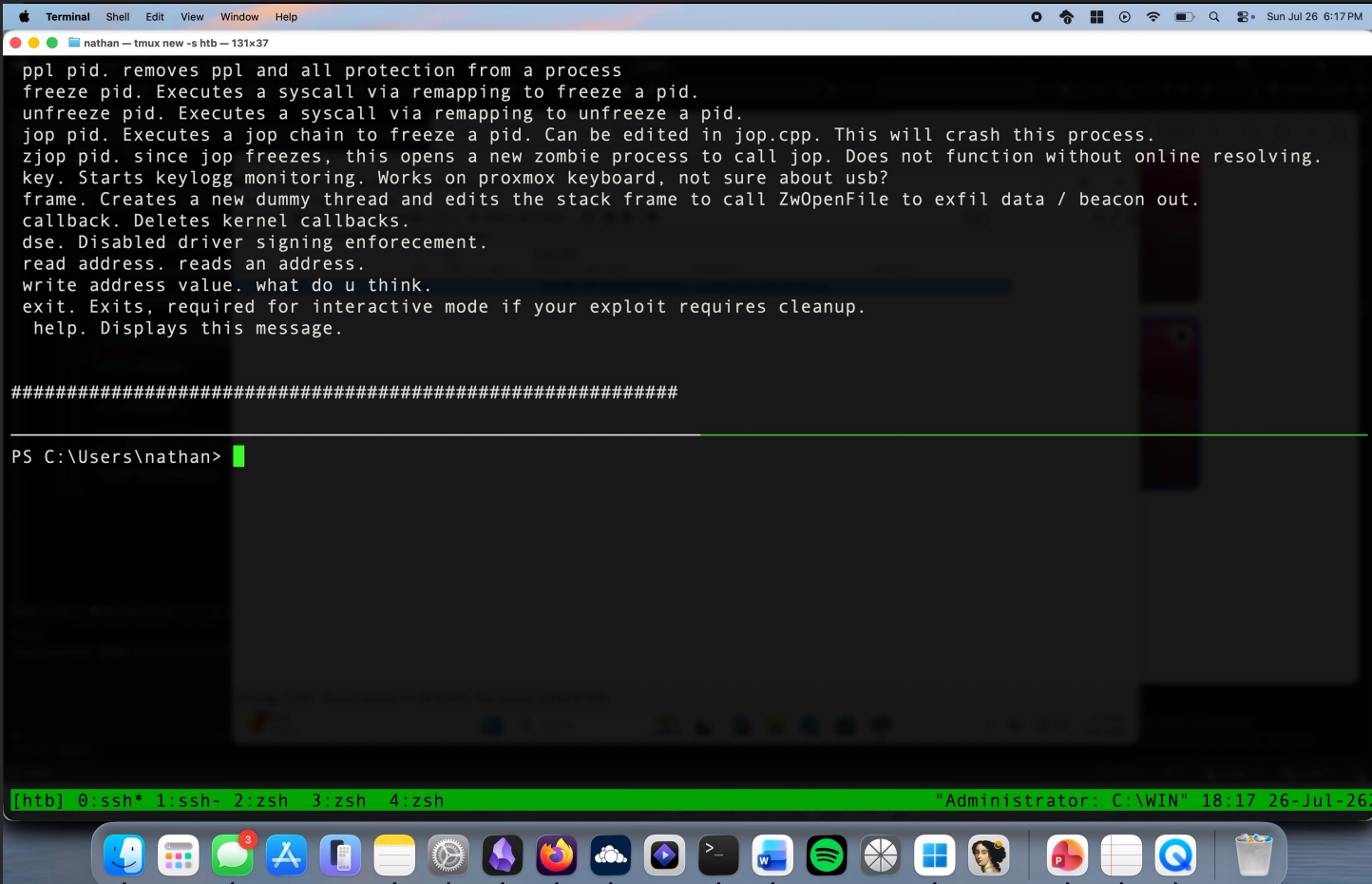
- We can issue a driver request to start chain
- Use primitive to overwrite Device_Object with gadgets
- Gadgets end with `jmp reg + offset`

```
jmp qword [rcx+0x20]
```



Picture from Offsec – Exploiting Control flow enforcement Technology

JOP Demo



```
Terminal Shell Edit View Window Help
nathan — tmux new -s htb — 131x37

ppl pid. removes ppl and all protection from a process
freeze pid. Executes a syscall via remapping to freeze a pid.
unfreeze pid. Executes a syscall via remapping to unfreeze a pid.
jop pid. Executes a jop chain to freeze a pid. Can be edited in jop.cpp. This will crash this process.
zjop pid. since jop freezes, this opens a new zombie process to call jop. Does not function without online resolving.
key. Starts keylogg monitoring. Works on proxmox keyboard, not sure about usb?
frame. Creates a new dummy thread and edits the stack frame to call ZwOpenFile to exfil data / beacon out.
callback. Deletes kernel callbacks.
dse. Disabled driver signing enforcement.
read address. reads an address.
write address value. what do u think.
exit. Exits, required for interactive mode if your exploit requires cleanup.
help. Displays this message.

#####

PS C:\Users\nathan> [
```

[htb] 0:ssh* 1:ssh- 2:zsh 3:zsh 4:zsh "Administrator: C:\WIN" 18:17 26-Jul-262

JOP Drawbacks

- Leaves a zombie process
- IRP request is never completed
- Requires two extra drivers
- Does work under Shadow Stacks!
- Working Kernel Code Execution!

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

SSDT hijack

- System service descriptor table
- Holds locations of syscalls
- Can change the offsets, so one syscall = another syscall

Why is this important?

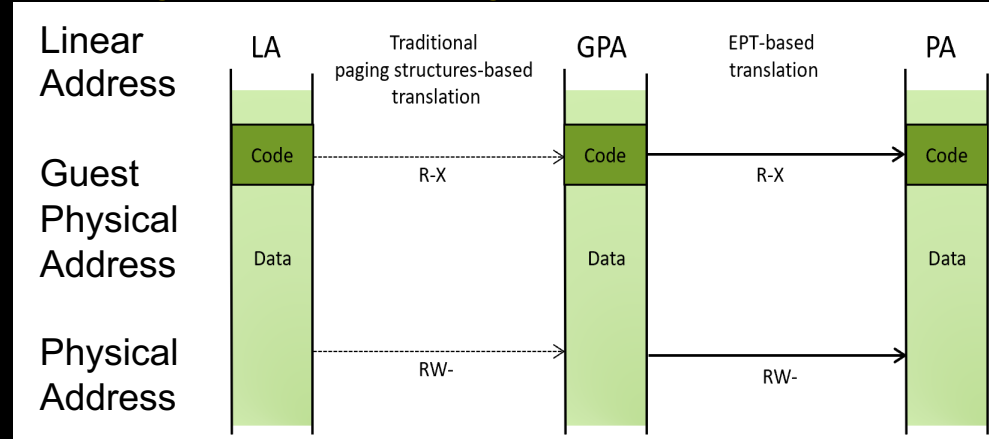
- Previously privileged action is now unprivileged
- Hiding actions
- Blocked by Kernel Data Protection

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Understanding Paging

- Linear Address is translated to a Guest Physical Address, done with CR3 register
- Guest Physical Address is translated to a true Physical Address, using Extended Page Tables

Remap
attack
source and
pictures:
Satoshi
Tanda



Kernel Data Protection and Remap

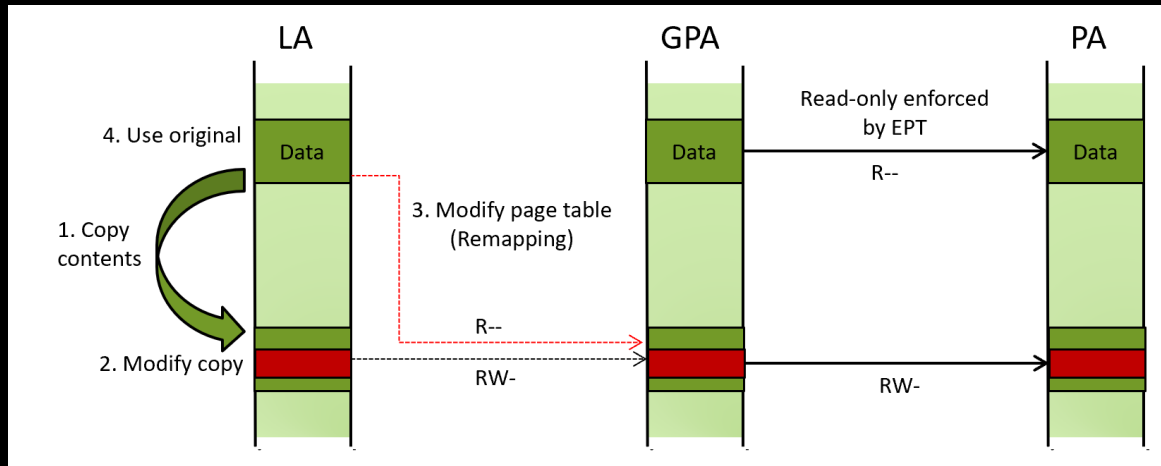
- Enforces specific sensitive memory is read only
- Kernel Data Protection enforces memory permissions between Guest Physical Address → Physical Address
- Remap the Linear Address!
- Edit the the Linear Address → Guest Physical Address translation
- Edit the page tables (not EPT!) and point GPA to different LA

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

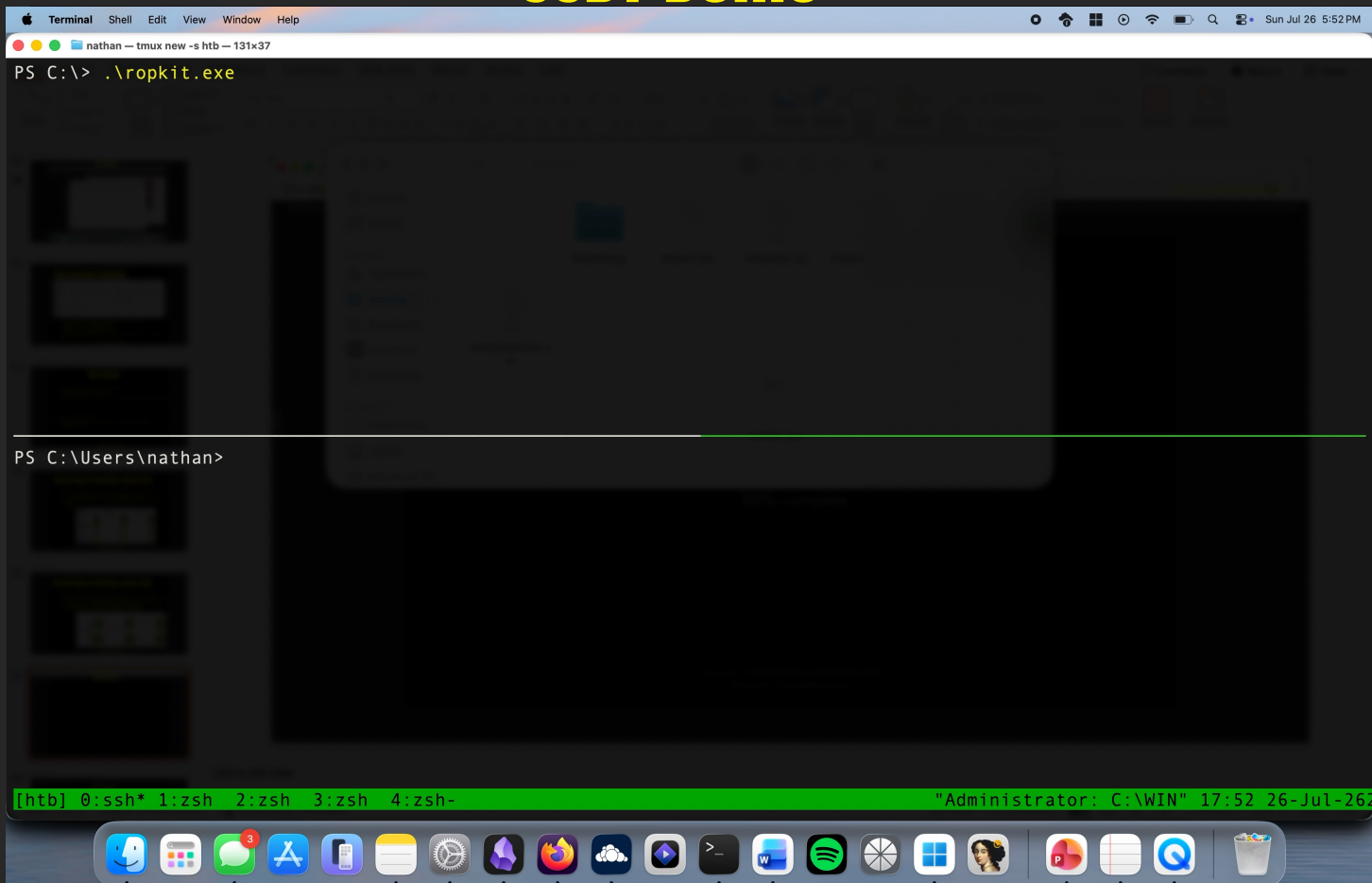
SSDT hijack

- With a read/write primitive can copy the contents, modify them, and update the page structure to point to our fake copy. This bypasses Kernel Data Protection!
- Blocked by Intel VT-RP. No AMD equivalent.

Remap
attack
source and
pictures:
Satoshi
Tanda



SSDT Demo



SSDT Drawbacks

- Blocked by Intel VT-RP in 12th gen CPUs
- No AMD Equivalent
- Detecting this would kill performance

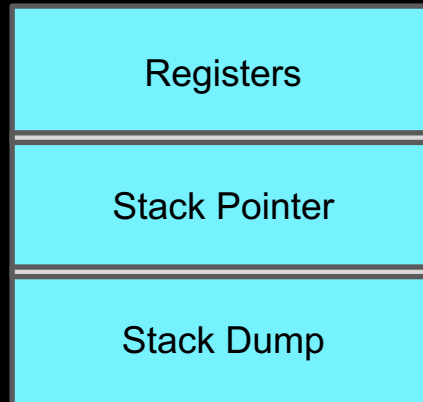
How does a CPU stop this?

- Intel VT-RP adds a separate Hypervisor Linear Address Translation Register (HLAT) field to stop this
- Translation is now fully done through hypervisor

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Trap Frame Editing

- Looks a little like this:
- Not protected!
- We can freeze a thread and edit



Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Trap Frame Editing

- What if we edited a frozen Trap Frame?
- This bypasses Shadow Stacks – for the launch
- We can directly change the Instruction Pointer
- Manually edit stack for complex calls

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Trap Frame Editing

- The stack can be edited for complex function calls!
- Let's issue a call to ZwOpenFile to ping an IP address from the kernel
- Can exfiltrate data over UDP requests to LAN using an address that does not exist

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Trap Frame Demo

The screenshot displays a noVNC interface for a Windows 11 virtual machine. The main window is a terminal titled "Terminal" with a menu bar (Shell, Edit, View, Window, Help). The terminal shows a series of commands and outputs from a user named "nathan" on a system named "DESKTOP-6P1BD47". The commands include several "Z:\>" prompts, followed by ".\ropkit.exe", and finally "[htb] 0:zsh 1:zsh 2:ssh* 3:sudo-". A green status bar at the bottom of the terminal window displays the text "Administrator: C:\WIN" 12:48 13-May-26".

Below the terminal window is a Wi-Fi packet capture window titled "Capturing from Wi-Fi: en0". It shows a list of captured packets. The first packet is highlighted, showing details for IP address 10.5.5.227 and TCP port 22. The packet details include:

No.	Time	Source	Destination	Protocol	Length	Info
3708	67.287226	10.5.5.227	10.5.5.255	BROWS...	243	Host Announcement DESKTOP-6P1BD47, Workstation, Server, NT Workstation

The bottom of the screen shows a macOS-style dock with various application icons, including Safari, Messages, Photos, Music, App Store, Mail, Calendar, Reminders, Notes, System Preferences, Firefox, VS Code, Docker Desktop, and others.

Trap Frame Downsides

- After payload is sent, this will crash with Shadow Stacks
- Likely recoverable!
- Fully recoverable (through editing return addresses)
- Can likely be explored/combined to become stable
- No zombie process or second driver needed!

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Wrap up!

- HVCI does a great job!
- Shadow Stacks and remap protection kill most attacks
- Many of these great protections are not enabled
- Shadow Stacks is not commonly enabled still

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys

Future Work

- Full network stack in Shadow Stacks
- More stable trap frame hijack
- No zombie processes
- Implement crypto mining
- Act as a kernel C2

nathan@nathan2.com

nathan2.com/posts/ropkit

github.com/nasawyer/ropkit

Thank you!

- Connor McGarr: Initial Rop chain and HVCI breakdown
- Satoshi Tanda: Remap explanation and attack
- Worawit Wangwarunyoo: KDP remap attack
- Jordan Higgins: Initial driver code, trap frame idea
- Juan Sacco: Offset resolution

nathan@nathan2.com

nathan2.com/posts/ropkit

github.com/nasawyer/ropkit

References

Remapping method: <https://tandasat.github.io/blog/2023/07/05/intel-vt-rp-part-1.html>

SSDT hijack method: <https://www.exploitpack.com/blogs/news/bypassing-kernel-code-execution-a-data-only-ssdt-hijack-under-hvci-but-how>

Vergilius project for initial offsets:

https://www.vergiliusproject.com/kernels/x64/windows-11/25h2/_MI_VISIBLE_STATE

Connor mgcar rop kit setup: <https://connormcgarr.github.io/hvci/>

jop ideas: <https://www.offsec.com/blog/bypassing-intel-cet-with-counterfeit-objects/>

kernel callback method: <https://github.com/worawit/malk>

kernel callbacks: <https://datafarm-cybersecurity.medium.com/code-execution-against-windows-hvci-f617570e9df0>

This was built using Jodan's <https://github.com/Kalagious/VulnerableDriver> and <https://github.com/Kalagious/VulnerableDriverReadWrite> library.

Stop code: SYSTEM_SERVICE_EXCEPTION (0x3B)
What failed: VulnerableDriver.sys