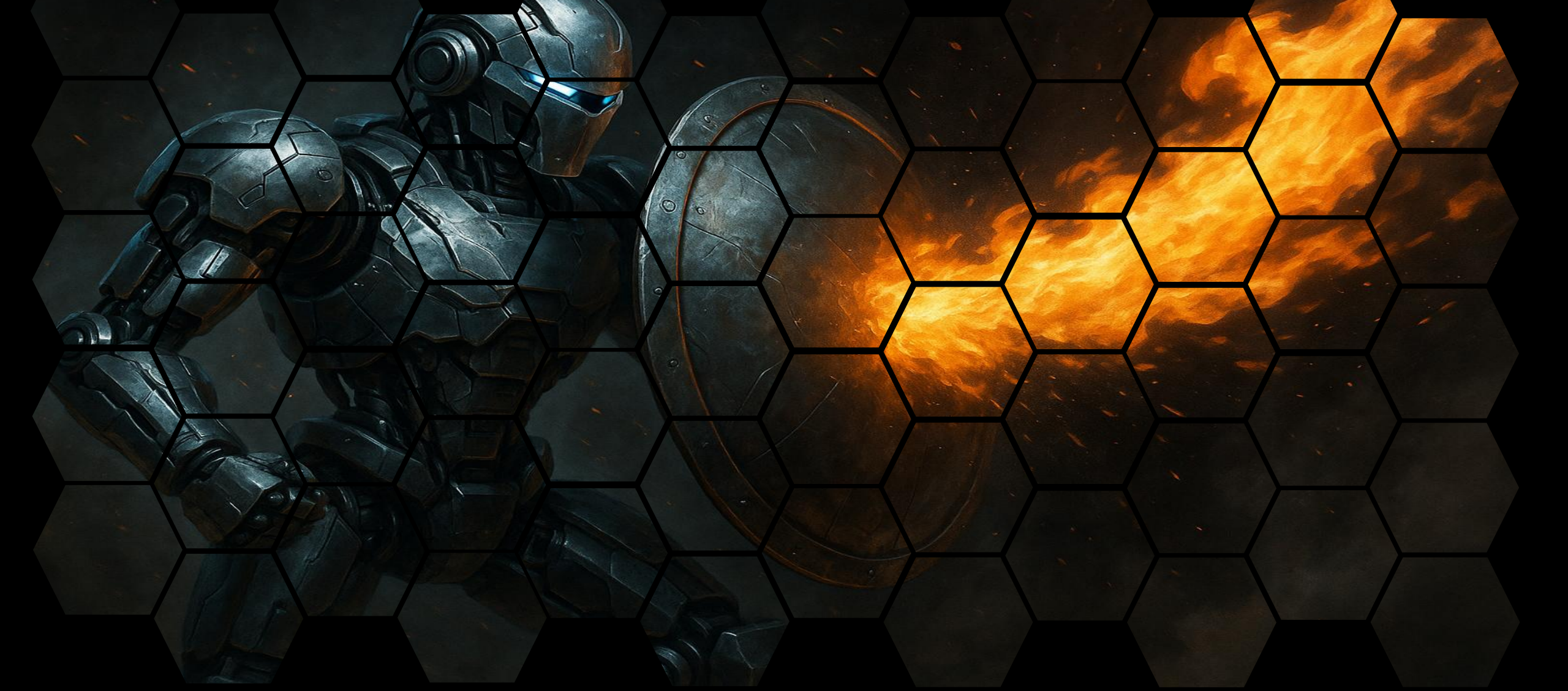


Malware detection… with type-1 hypervisors

… by André Lima (0x4ndr3) @ Sikkerhetsfestivalen 2025

> whoami

C:\WINDOWS\system32\cmd.exe

```
C:\Users\Andre Lima\Documents>type whoami.txt

 - Red Team Lead @ Telenor CyberDefence
 - Speaker / Researcher
     Bsides Oslo 2022, 2023
     Sikkerhetsfestivalen 2023, 2024, 2025
     Bsides Lisbon 2022
 - Pentester, Red Teamer, Researcher since 2011
 - Worked in Portugal and Australia
 - Blogger
     [new] https://medium.com/@0x4ndr3
     [old] https://pentesterslife.blog/
 - Youtube:
     https://www.youtube.com/@0x4ndr3
 - OSED, eCRE, SLAE64, eWPTX, OSCP, etc
```

Table of Contents

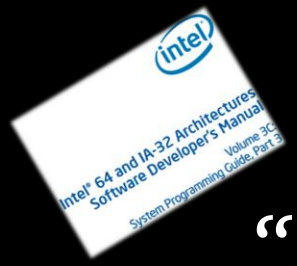
⚠️ Disclaimer ⚠️

- Type-1 (bare-metal: ESXi, Xen, Hyper-v) not type-2 (VMware Workstation, VMware Fusion, Oracle VirtualBox, Parallels Desktop, KVM)
- Intel VT-x != AMD-V (initially SVM)
- This presentation is all about VT-x
- This presentation, by its very nature, has some gross over-simplifications!

⚠️ Disclaimer ⚠️

- Type-1 (bare-metal: ESXi, Xen, Hyper-v) not type-2 (VMware Workstation, VMware Fusion, Oracle VirtualBox, Parallels Desktop, KVM)
- Intel VT-x != AMD-V (initially SVM)
- This presentation is all about VT-x
- This presentation, by its very nature, has some gross over-simplifications!

“Intel VT-x provides hardware-assisted virtualization, enabling isolation between virtual machines with minimal performance overhead.”

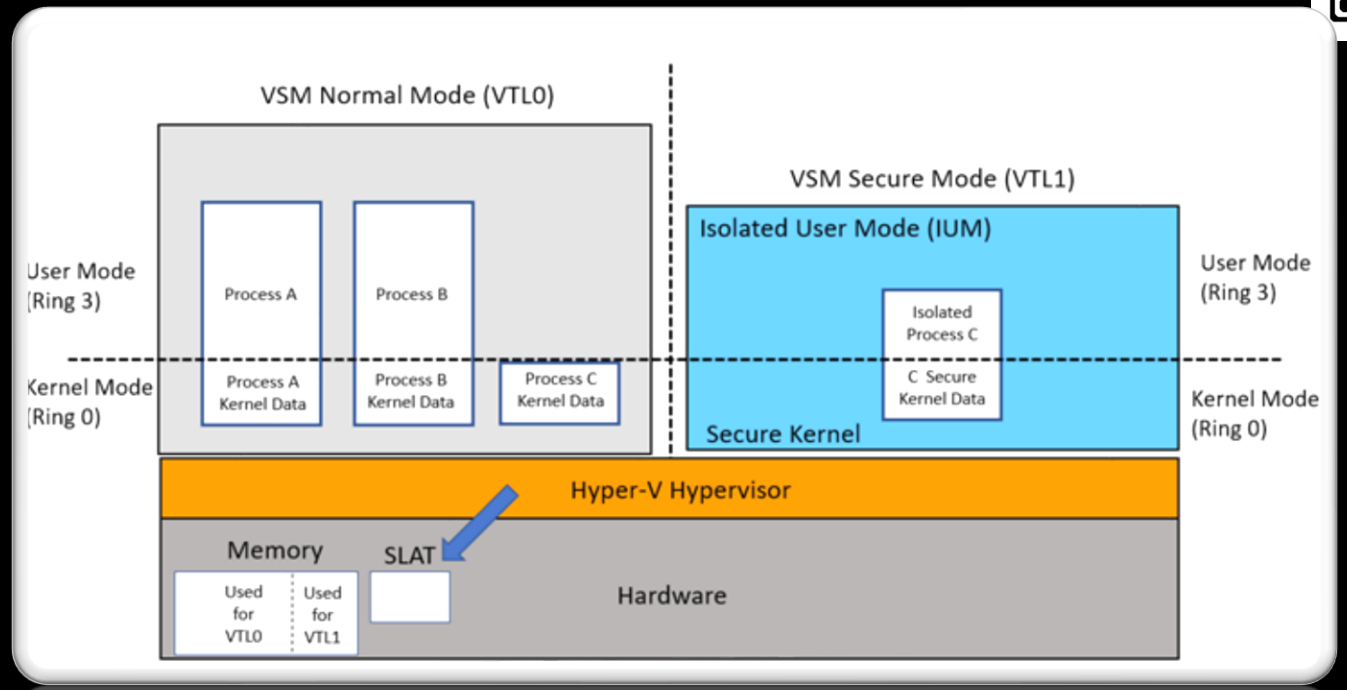


“Intel VT-x provides hardware-assisted virtualization, enabling isolation between virtual machines with minimal performance overhead.”

- Runs multiple OSes on same hardware
- Introduces two CPU modes: VMX Root & VMX Non-Root
- Controlled by the VMM (hypervisor)

But wait! why over-complicate things ?? 🤔

- Historical lack of trust in the KM



Intel VT-x

- VMM & Guest
  - VMM (Virtual Machine Monitor): Manages and controls VMs, runs in VMX Root
  - Guest: The OS running inside a VM, in VMX Non-Root

Intel VT-x

- VMX Root vs. VMX Non-Root
  - Root Operation: Hypervisor control, full hardware access
  - Non-Root Operation: Guest OS execution, trapped when needed

Intel VT-x

- VMX Transitions
  - VM Entry: Switch from VMM → Guest (Root → Non-Root)
  - VM Exit: Switch from Guest → VMM (Non-Root → Root)

Intel VT-x

- Extended Page Tables (EPT)
  - Allows for SLAT
  - VA > GPA > SPA

Select Administrator: Windows PowerShell

```
PS C:\WINDOWS\system32> coreinfo -v

Coreinfo v3.31 - Dump information on system CPU and memory topology
Copyright (C) 2008-2014 Mark Russinovich
Sysinternals - www.sysinternals.com

Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
Intel64 Family 6 Model 158 Stepping 10, GenuineIntel
Microcode signature: 000000B4
HYPERVISOR      -       Hypervisor is present
VMX             *       Supports Intel hardware-assisted virtualization
EPT             *       Supports Intel extended page tables (SLAT)
PS C:\WINDOWS\system32>
```

Intel VT-x

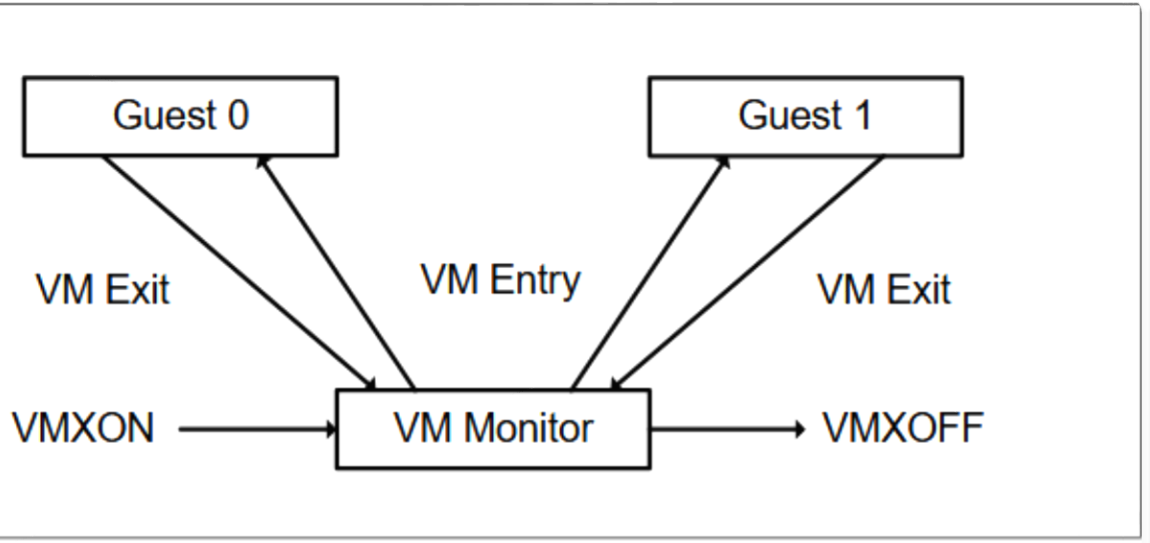


Figure 23-1. Interaction of a Virtual-Machine Monitor and Guests

APPENDIX C
VMX BASIC EXIT REASONS

Every VM exit writes a 32-bit exit reason to the VMCS (see Section 26.9.1). Certain VM-entry failures also do this (see Section 28.8). The low 16 bits of the exit-reason field form the basic exit reason which provides basic information about the cause of the VM exit or VM-entry failure.

Table C-1 lists values for basic exit reasons and explains their meaning. Entries apply to VM exits, unless otherwise noted.

Table C-1. Basic Exit Reasons

Basic Exit Reason	Description
0	**Exception or non-maskable interrupt (NMI).** Either: 1: Guest software caused an exception and the bit in the exception bitmap associated with exception's vector was 1. This case includes executions of BOUND that cause #BR, executions of INT1 (they cause #DB), executions of INT3 (they cause #BP), executions of INTO that cause #OF, and executions of UD0, UD1, and UD2 (they cause #UD). 2: An NMI was delivered to the logical processor and the "NMI exiting" VM-execution control was 1.
1	**External interrupt.** An external interrupt arrived and the "external-interrupt exiting" VM-execution control was 1.
2	**Triple fault.** The logical processor encountered an exception while attempting to call the double-fault handler and that exception did not itself cause a VM exit due to the exception bitmap.
3	**INIT signal.** An INIT signal arrived
4	**Start-up IPI (SIPI).** A SIPI arrived while the logical processor was in the "wait-for-SIPI" state.
5	**I/O system-management interrupt (SMI).** An SMI arrived immediately after retirement of an I/O instruction and caused an SMM VM exit (see Section 33.15.2).
6	**Other SMI.** An SMI arrived and caused an SMM VM exit (see Section 33.15.2) but not immediately after retirement of an I/O instruction.
7	**Interrupt window.** At the beginning of an instruction, RFLAGS.IF was 1; events were not blocked by STI or by MOV SS; and the "interrupt-window exiting" VM-execution control was 1.
8	**NMI window.** At the beginning of an instruction, there was no virtual-NMI blocking; events were not blocked by MOV SS; and the "NMI-window exiting" VM-execution control was 1.
9	**Task switch.** Guest software attempted a task switch.
10	**CPUID.** Guest software attempted to execute CPUID.
11	**GETSEC.** Guest software attempted to execute GETSEC.
12	**HLT.** Guest software attempted to execute HLT and the "HLT exiting" VM-execution control was 1.
13	**INVD.** Guest software attempted to execute INVD.
14	**INVLPG.** Guest software attempted to execute INVLPG and the "INVLPG exiting" VM-execution control was 1.
15	**RDPMC.** Guest software attempted to execute RDPMC and the "RDPMC exiting" VM-execution control was 1.
16	**RDTSC.** Guest software attempted to execute RDTSC and the "RDTSC exiting" VM-execution control was 1.
17	**RSM.** Guest software attempted to execute RSM in SMM.
18	**VMCALL.** VMCALL was executed either by guest software (causing an ordinary VM exit) or by the executive monitor

60	**ENCLS.** Guest software attempted to execute ENCLS, "enable ENCLS exiting" VM-execution control was 1, and either (1) EAX < 63 and the corresponding bit in the ENCLS-exiting bitmap is 1; or (2) EAX ? 63 and bit 63 in the ENCLS-exiting bitmap is 1.
61	**RDSEED.** Guest software attempted to execute RDSEED and the "RDSEED exiting" VM-execution control was 1.
62	**Page-modification log full.** The processor attempted to create a page-modification log entry and the value of the PML index was not in the range 0–511.
63	**XSAVES.** Guest software attempted to execute XSAVES, the "enable XSAVES/XRSTORS" was 1, and a bit was set in the logical-AND of the following three values: EDX:EAX, the IA32_XSS MSR, and the XSS-exiting bitmap.
64	**XRSTORS.** Guest software attempted to execute XRSTORS, the "enable XSAVES/XRSTORS" was 1, and a bit was set in the logical-AND of the following three values: EDX:EAX, the IA32_XSS MSR, and the XSS-exiting bitmap.
65	**PCONFIG.** Guest software attempted to execute PCONFIG, "enable PCONFIG" VM-execution control was 1, and either (1) EAX < 63 and the corresponding bit in the PCONFIG-exiting bitmap is 1; or (2) EAX ? 63 and bit 63 in the PCONFIG-exiting bitmap is 1.
66	**SPP-related event.** The processor attempted to determine an access's sub-page write permission and encountered an SPP miss or an SPP misconfiguration. See Section 30.3.4.2.
67	**UMWAIT.** Guest software attempted to execute UMWAIT and the "enable user wait and pause" and "RDTSC exiting" VM-execution controls were both 1.
68	**TPAUSE.** Guest software attempted to execute TPAUSE and the "enable user wait and pause" and "RDTSC exiting" VM-execution controls were both 1.
69	**LOADIWKEY.** Guest software attempted to execute LOADIWKEY and the "LOADIWKEY exiting" VM-execution control was 1.
72	**ENQCMD PASID translation failure.** A VM exit occurred during PASID translation because the present bit was clear in a PASID-directory entry, the valid bit was clear in a PASID-table entry, or one of the entries set a reserved bit.
73	**ENQCMDS PASID translation failure.** A VM exit occurred during PASID translation because the present bit was clear in a PASID-directory entry, the valid bit was clear in a PASID-table entry, or one of the entries set a reserved bit.
74	**Bus lock.** The processor asserted a bus lock while the "bus-lock detection" VM-execution control was 1. (Such VM exits will also set bit 26 of the exit-reason field.)
75	**Instruction timeout.** The "instruction timeout" VM-execution control was 1 and certain operations prevented the processor from reaching an instruction boundary within the amount of time specified by the instruction-timeout control.
76	**SEAMCALL.** Guest software attempted to execute SEAMCALL.[1]
77	**TDCALL.** Guest software attempted to execute TDCALL.[1]
78	**RDMSRLIST.** Guest software attempted to execute RDMSRLIST and either the "use MSR bitmaps" VM-execution control was 0 or any of the following holds for the index an MSR being accessed: ▪ The index is neither in the range 00000000H – 00001FFFH nor in the range C0000000H – C0001FFFH. ▪ The index is in the range 00000000H – 00001FFFH and the nth bit in read bitmap for low MSRs is 1, where *n* is the index. ▪ The index is in the range C0000000H – C0001FFFH and the nth bit in read bitmap for high MSRs is 1, where *n* is the logical AND of the index and the value 00001FFFH.
79	**WRMSRLIST.** Guest software attempted to execute WRMSRLIST and either the "use MSR bitmaps" VM-execution control was 0 or any of the following holds for the index an MSR being accessed: ▪ The index is neither in the range 00000000H – 00001FFFH nor in the range C0000000H – C0001FFFH. ▪ The index is in the range 00000000H – 00001FFFH and the nth bit in write bitmap for low MSRs is 1, where *n* is the index. ▪ The index is in the range C0000000H – C0001FFFH and the nth bit in write bitmap for high MSRs is 1, where *n* is the logical AND of the index and the value 00001FFFH.

36	**MWAIT.** Guest software attempted to execute MWAIT and the "MWAIT exiting" VM-execution control was 1.
37	**Monitor trap flag.** A VM exit occurred due to the 1-setting of the "monitor trap flag" VM-execution control (see Section 27.5.2) or VM entry injected a pending MTF VM exit as part of VM entry (see Section 28.6.2).
39	**MONITOR.** Guest software attempted to execute MONITOR and the "MONITOR exiting" VM-execution control was 1.
40	**PAUSE.** Either guest software attempted to execute PAUSE and the "PAUSE exiting" VM-execution control was 1 or the "PAUSE-loop exiting" VM-execution control was 1 and guest software executed a PAUSE loop with execution time exceeding PLE_Window (see Section 27.1.3).
41	**VM-entry failure due to machine-check event.** A machine-check event occurred during VM entry (see Section 28.9).
43	**TPR below threshold.** The logical processor determined that the value of bits 7:4 of the byte at offset 080H on the virtual-APIC page was below that of the TPR threshold VM-execution control field while the "use TPR shadow" VM-execution control was 1 either as part of TPR virtualization (Section 31.1.2) or VM entry (Section 28.7.7).
44	**APIC access.** Guest software attempted to access memory at a physical address on the APIC-access page and the "virtualize APIC accesses" VM-execution control was 1 (see Section 31.4).
45	**Virtualized EOI.** EOI virtualization was performed for a virtual interrupt whose vector indexed a bit set in the EOI-exit bitmap.
46	**Access to GDTR or IDTR.** Guest software attempted to execute LGDT, LIDT, SGDT, or SIDT and the "descriptor-table exiting" VM-execution control was 1.
47	**Access to LDTR or TR.** Guest software attempted to execute LLDT, LTR, SLDT, or STR and the "descriptor-table exiting" VM-execution control was 1.
48	**EPT violation.** An attempt to access memory with a guest-physical address was disallowed by the configuration of the EPT paging structures.
49	**EPT misconfiguration.** An attempt to access memory with a guest-physical address encountered a misconfigured EPT paging-structure entry.
50	**INVEPT.** Guest software attempted to execute INVEPT.
51	**RDTSCP.** Guest software attempted to execute RDTSCP and the "enable RDTSCP" and "RDTSC exiting" VM-execution controls were both 1.
52	**VMX-preemption timer expired.** The preemption timer counted down to zero.
53	**INVVPID.** Guest software attempted to execute INVVPID.
54	**WBINVD or WBNOINVD.** Guest software attempted to execute WBINVD or WBNOINVD and the "WBINVD exiting" VM-execution control was 1.
55	**XSETBV.** Guest software attempted to execute XSETBV.
56	**APIC write.** Guest software completed a write to the virtual-APIC page that must be virtualized by VMM software (see Section 31.4.3.3).
57	**RDRAND.** Guest software attempted to execute RDRAND and the "RDRAND exiting" VM-execution control was 1.
58	**INVPCID.** Guest software attempted to execute INVPCID and the "enable INVPCID" and "INVLPG exiting" VM-execution controls were both 1.
59	**VMFUNC.** Guest software invoked a VM function with the VMFUNC instruction and the VM function either was not enabled or generated a function-specific condition causing a VM exit.
60	**ENCLS.** Guest software attempted to execute ENCLS, "enable ENCLS exiting" VM-execution control was 1, and either (1) EAX < 63 and the corresponding bit in the ENCLS-exiting bitmap is 1; or (2) EAX ? 63 and bit 63 in the ENCLS-exiting bitmap is 1.
61	**RDSEED.** Guest software attempted to execute RDSEED and the "RDSEED exiting" VM-execution control was 1.
62	**Page-modification log full.** The processor attempted to create a page-modification log entry and the value of the PML index was not in the range 0–511.
63	**XSAVES.** Guest software attempted to execute XSAVES, the "enable XSAVES/XRSTORS" was 1, and a bit was set in the logical-AND of the following three values: EDX:EAX, the IA32_XSS MSR, and the XSS-exiting bitmap.
64	**XRSTORS.** Guest software attempted to execute XRSTORS, the "enable XSAVES/XRSTORS" was 1, and a bit was set

Ways to attach code to HV

- UEFI-based vs OS kernel module-based hypervisors
  - UEFI-based
    - Pros: more powerful
    - Cons: harder to dev
  - OS kernel module
    - Pros: easier to dev
    - Cons: less powerful

Ways to attach code to HV

- HV types by functionality
  - full-fledged
  - pass-through: only virtualizes existing CPUs and MMUs
    - Much (MUCH) smaller and easier to understand concepts
    - aka: “Blue-pill style”, “Hyperjack-style” and “Type-0”

Ways to attach code to HV

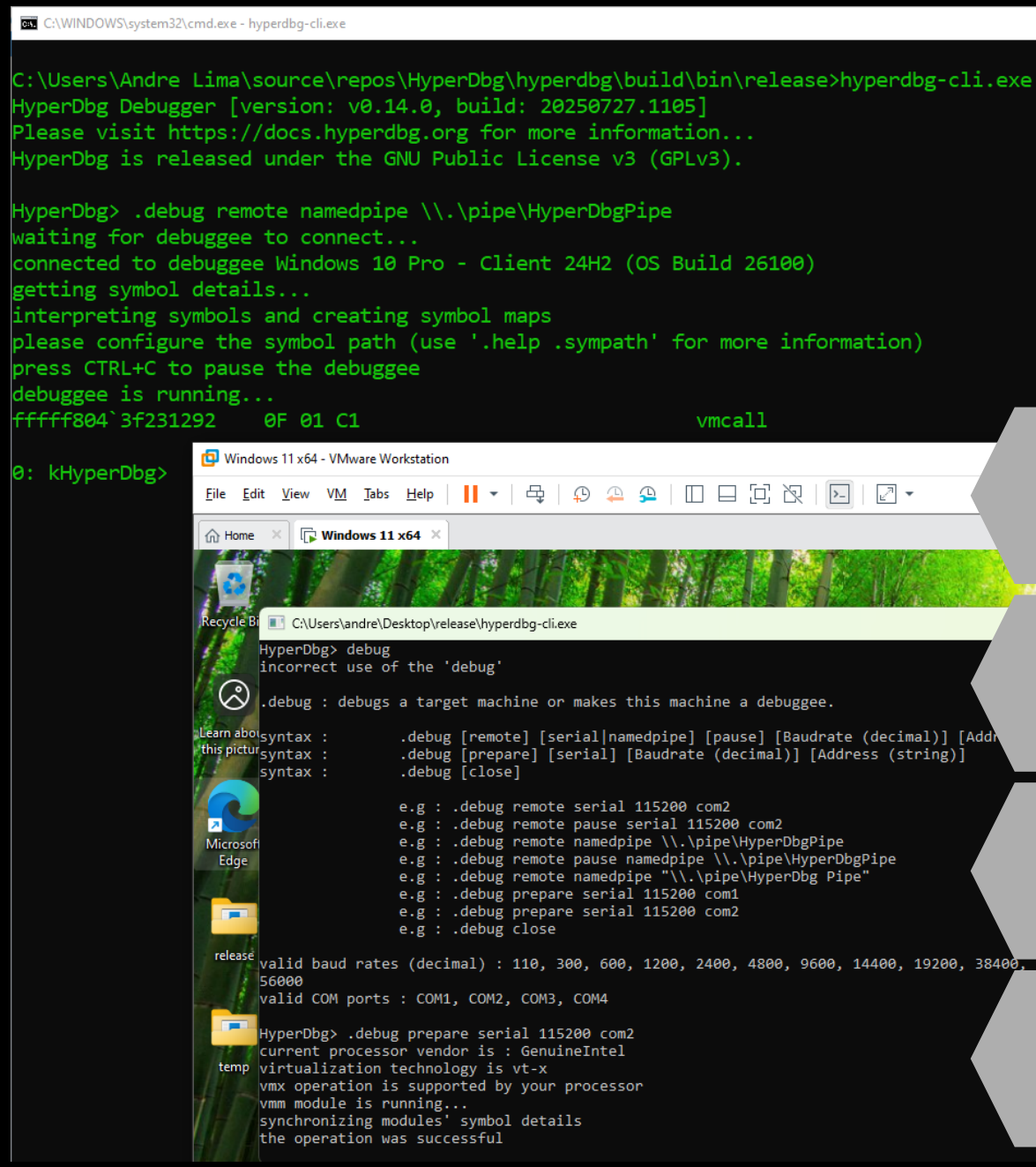
- UEFI-based pass-through hypervisor
  - Cross-platform by design
  - More stealthiness
  - Better to understand VT
  - Greater ability to take control of the system

Ways to attach code to HV

- kernel module-based pass-through hypervisor
  - Good if closer interaction with OS is desirable
  - Reboot on installation is not required
  - Can utilize expertise with familiar tools and workflow
  - Examples: EDR…

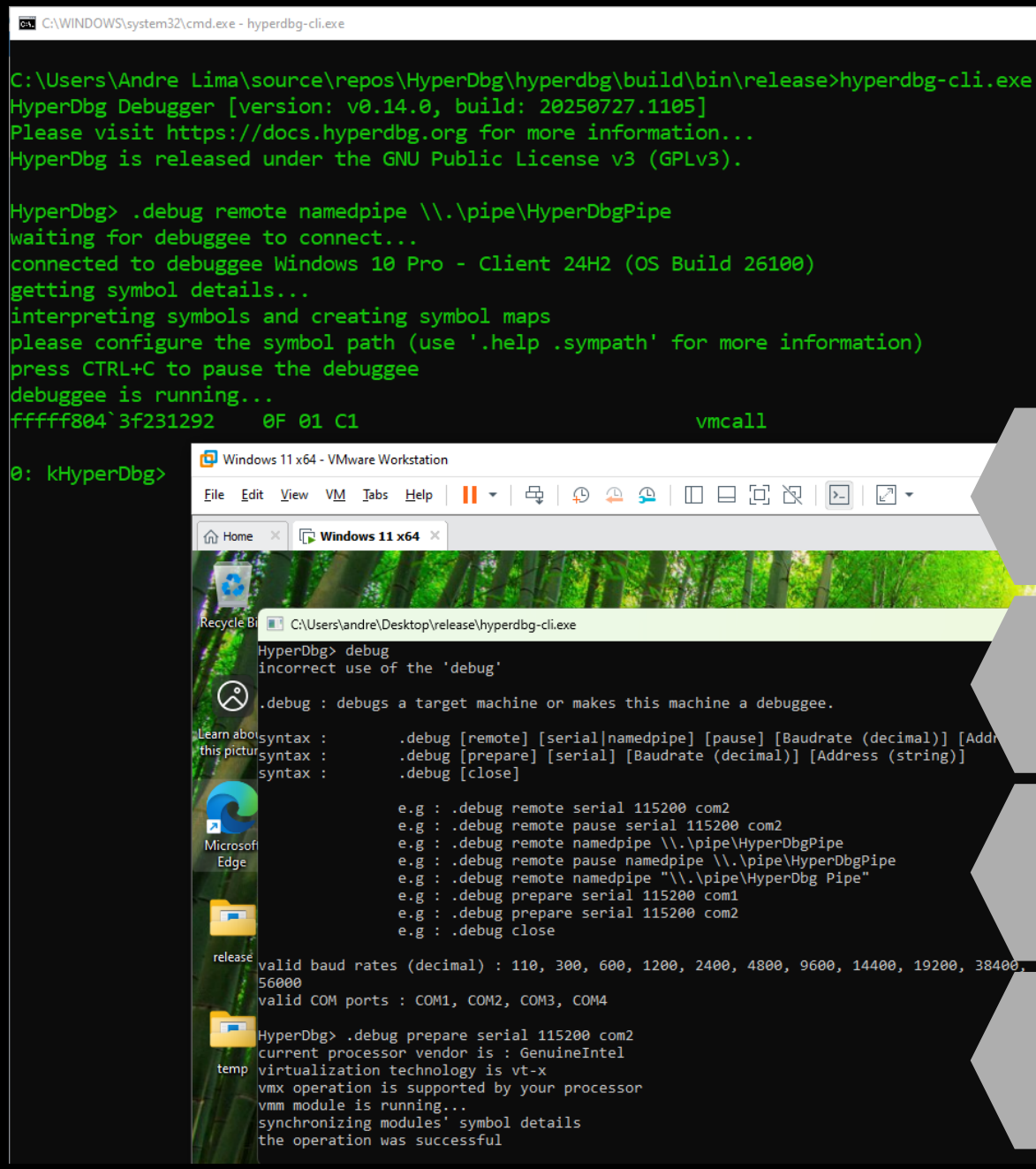
Demo – Explanation

- However, what we are doing here is:
  - Use HyperDBG
  - Hook on a VM Exit, and manipulate the response to the UM process who caused the VM Exit in the first place



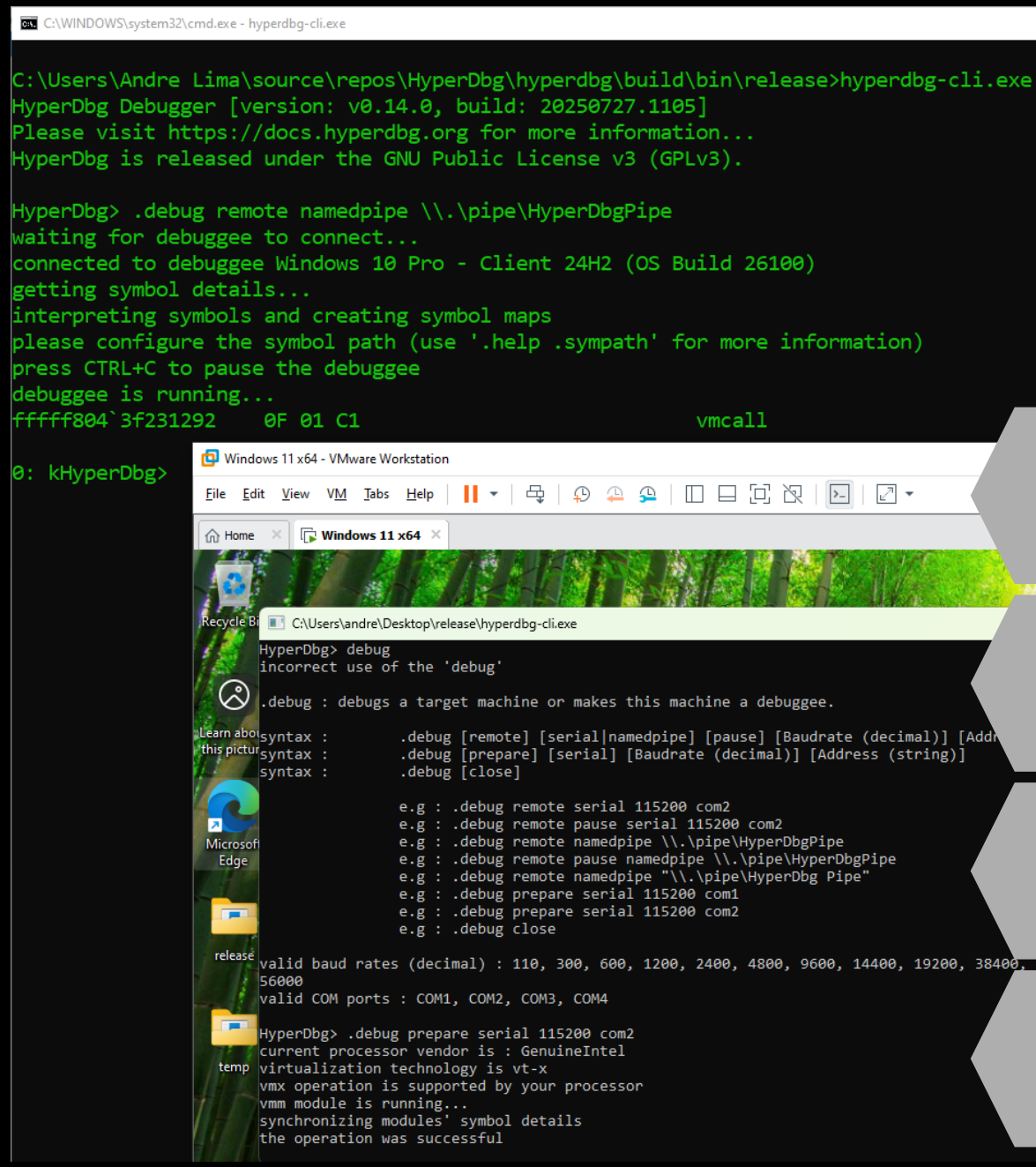
Demo – Explanation

- Typical things to watch out for:
  - EPT violations: X on non-image kernel pages; W on kernel .text; optional R/W on SSDT/IDT/GDT.
  - CR writes: CR0, CR3 (with CR3-targeting), CR4.
  - MSRs (bitmap): LSTAR/STAR/SFMASK, SYSENTER trio, EFER, DEBUGCTL, TSC-related.
  - Descriptor ops: LGDT/LIDT (and LTR/LLDT if you suspect segment tricks).
  - Selective I/O via IO bitmaps for ports of interest.
  - DRx, CPUID (anti-VM), INVLPG/INVPCID, VMCALL.



Demo – Explanation

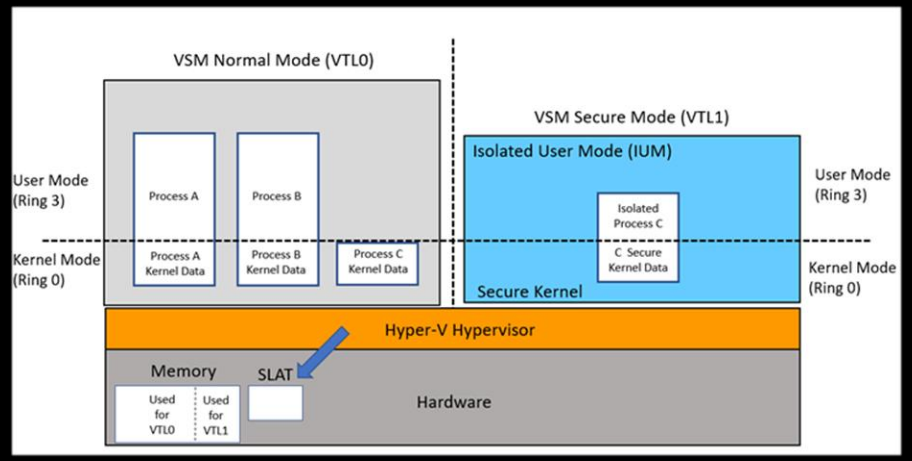
- Typical things to watch out for:
  - EPT violations: X on non-image kernel pages; W on kernel .text; optional R/W on SSDT/IDT/GDT.
  - DRx, CPUID (anti-VM), INVLPG/INVPCID, VMCALL.



Demo…

Conclusions

- Exploitation is going “down”

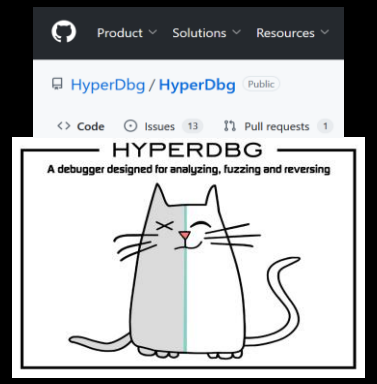


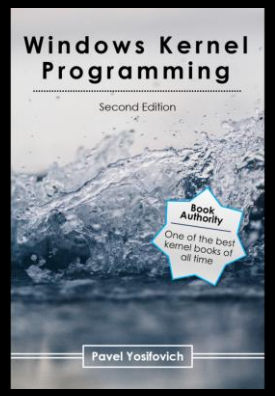
- Generalized lack of understanding of these tech in the industry
  - Makes everyone more vulnerable

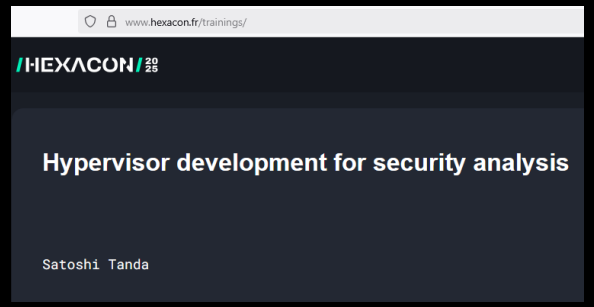
Conclusions

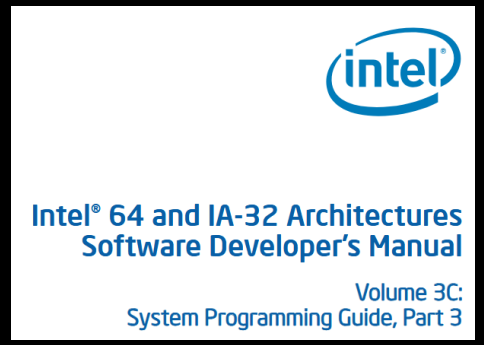
- Blue teams / Defense providers
  - Learn this tech. Start investing in it…
- If you're buying EDRs
  - Be more inquisitive about their HV capability
  - Test them through specific Red Teams / Purple Teams targeting such tech

Resources


Product
Solutions
Resources
HyperDbg / HyperDbg
Public
Code
Issues 13
Pull requests 1
HYPERDBG
A debugger designed for analyzing, fuzzing and reversing


Windows Kernel Programming
Second Edition
Book Authority
One of the best kernel books of all time
Pavel Yosifovich


www.hexacon.fr/trainings/
HEXACON 2025
Hypervisor development for security analysis
Satoshi Tanda


intel
Intel® 64 and IA-32 Architectures
Software Developer's Manual
Volume 3C:
System Programming Guide, Part 3

Thank you for coming… ☺

https://www.linkedin.com/in/aflima/

0x4ndr3.bsky.social

0x4ndr3

https://andrelima.info

Interested in working for us?

Or in the best SOC services in Norway?

TIBER Red Team tests?

Modern Purple Team tests?

telenor | cyberdefence