s0fT7: Revealing the Secrets of the Siemens S7 PLCs

Sara Bitan | Alon Dankner

Joint work with **Professor Eli Biham**, **Maxim Barsky** and **Idan Raz**

Faculty of Computer Science, Technion – Israel Institute of Technology

Who are we?

Sara Bitan

Founder and CEO of CyCloak: Secure system design and audit

Senior researcher at the Technion Hiroshi Fujiwara Cyber Security Research Center

Alon Dankner

Security researcher
M.Sc. graduate from the Technion

Advisors: **Prof. Eli Biham, Dr. Sara Bitan**

PLCs turn rogue…

Stuxnet
(Anonymous author)

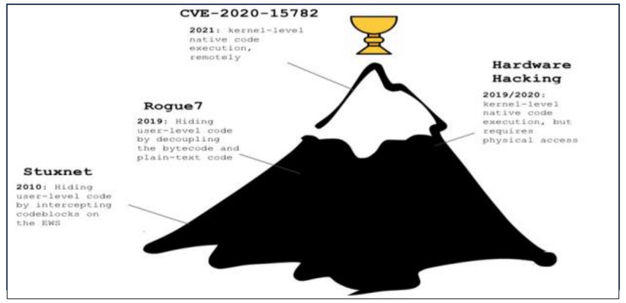
- Exploit a vulnerable Siemens Step7 engineering station/ WinCC HMI client
- Inject a rogue control program, and tamper with HMI outputs

Rogue7: Rogue Engineering-Station attacks on S7 Simatic PLCs
(Biham, et al)

- A phyton script impersonating an engineering WS
- All S7 PLCs from the same model and firmware version share the same key

Doors of Durin: The Veiled Gate to Siemens S7 Silicon
(Abbasi, et al)

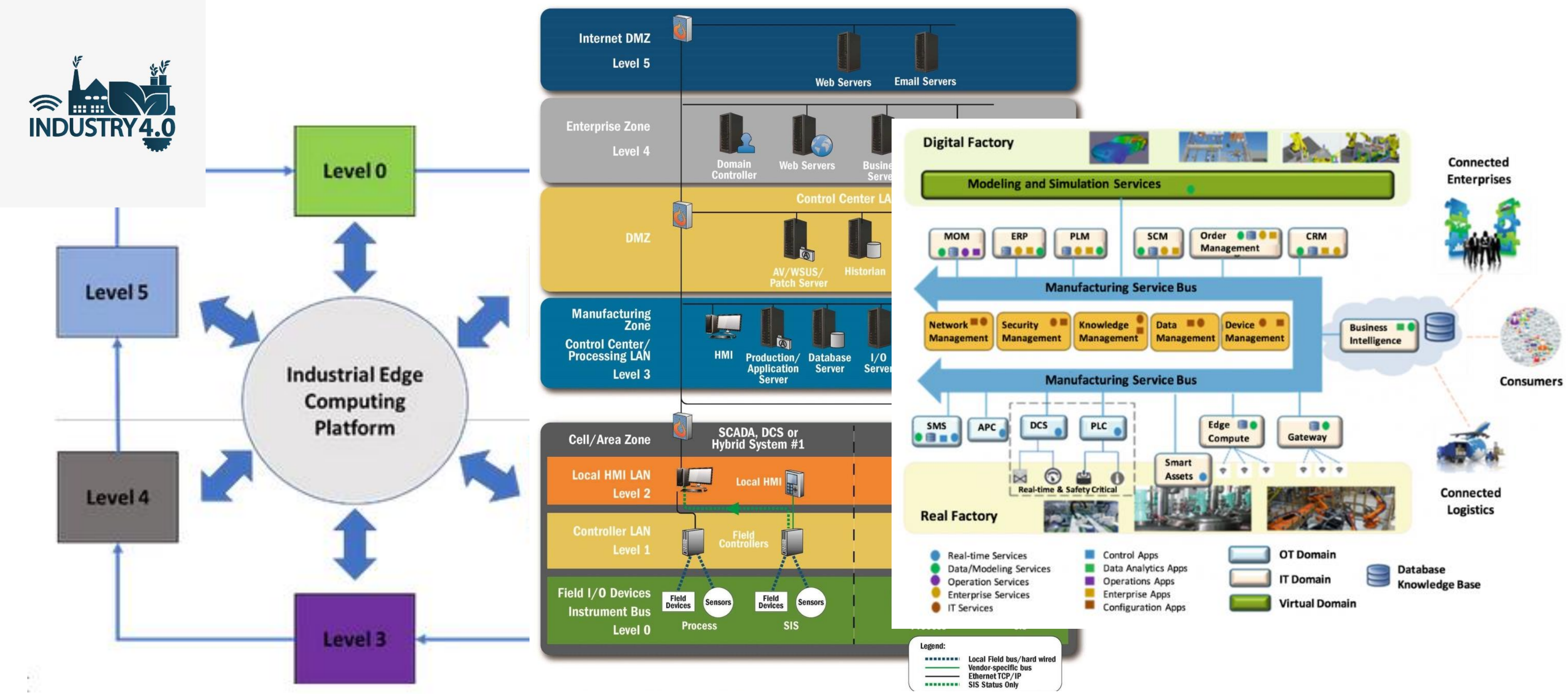
- Siemens S7-1200 PLC Bootloader Arbitrary Code Execution
- Siemens S7 firmware is using Adonis kernel


CVE-2020-15782
2021: kernel-level native code execution, remotely
Hardware Hacking
2019/2020: kernel-level native code execution, but requires physical access
Rogue7
2019: Hiding user-level code by decoupling the bytecode and plain-text code
Stuxnet
2010: Hiding user-level code by intercepting codeblocks on the EWS

The Race to Native Code Execution in PLCs
(Keren)

- Remote arbitrary code execution on Siemens S7-1500
- Exploiting memory protection vulnerability to escape the control program sandbox

ICS architectures are evolving



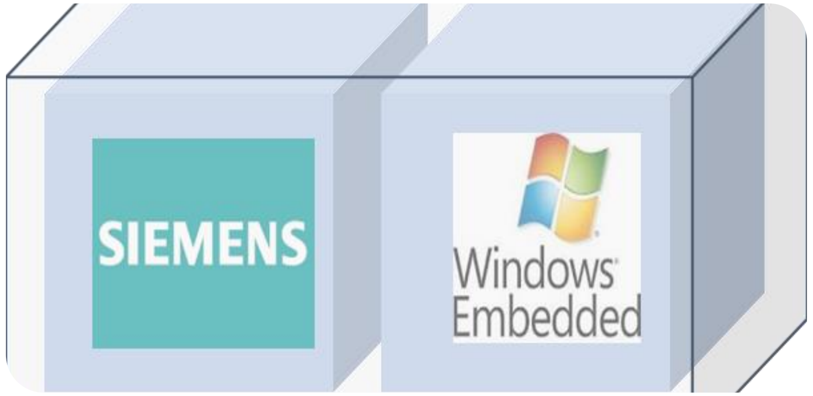
Software PLCs

Smart Manufacturing

- New requirements from PLC vendors
- New features: IDEs, new protocols, extensive cloud communication

Vendor Requirements

- Agility and flexibility
- Preserve existing IP and technology
- The solution: software PLCs



New PLC architecture

- Generic functions: GP OS – updatable, flexible→ Standard hardware
- Legacy functions : proprietary OS - closed and hardened
- Virtualization: isolation and separation

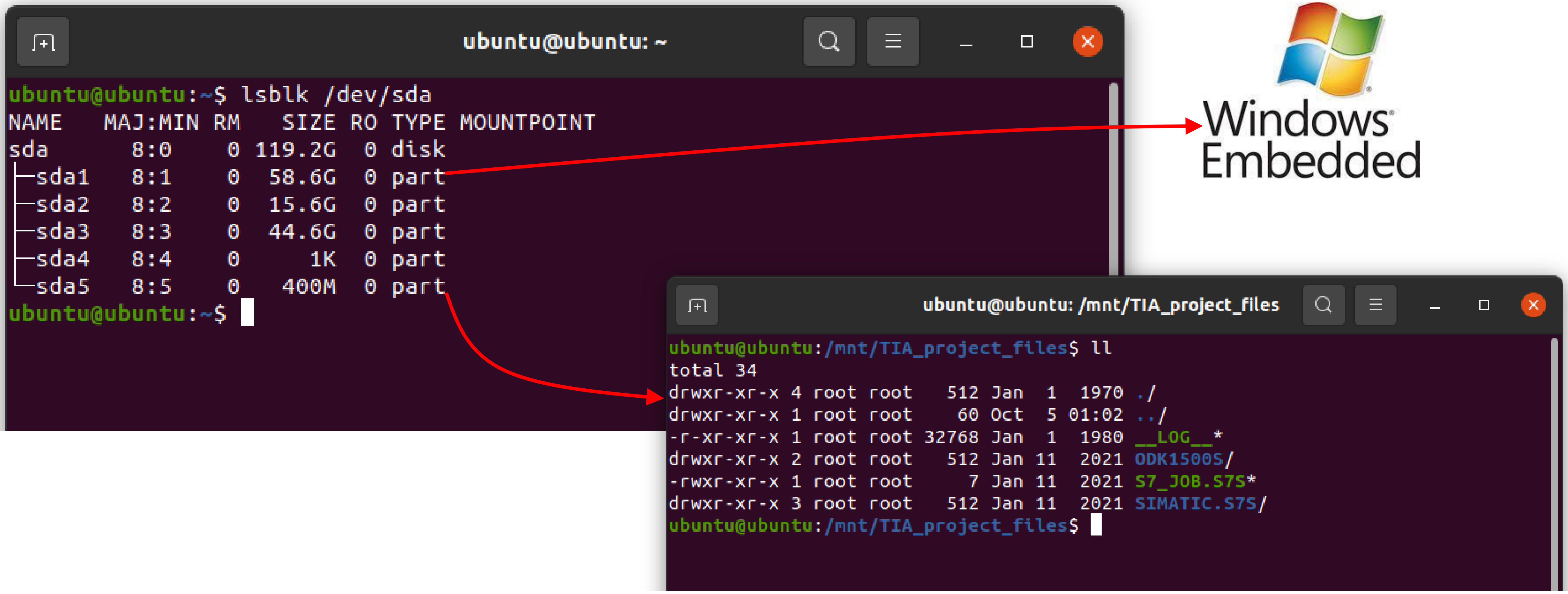
Siemens ET 200SP open controller

- The PC-based version of SIMATIC S7-1500
  - Introduced in January 2015
- Combines PLC functionality with a PC-based platform using virtualization
- Isolation between Windows and control logic
  - Supports Windows updates and reboot without interruption to the control logic
  - The controller continues to work even if Windows crashes
- DUT: CPU 1515SP PC2

The quest...

Ubuntu
SIEMENS
SIEMENS
SIMATIC
S7 - 1500
RUN
Overview
CPU 1515-2PM
X1 uy g 5.1
X2 uy g 7.1
OK

The boot process

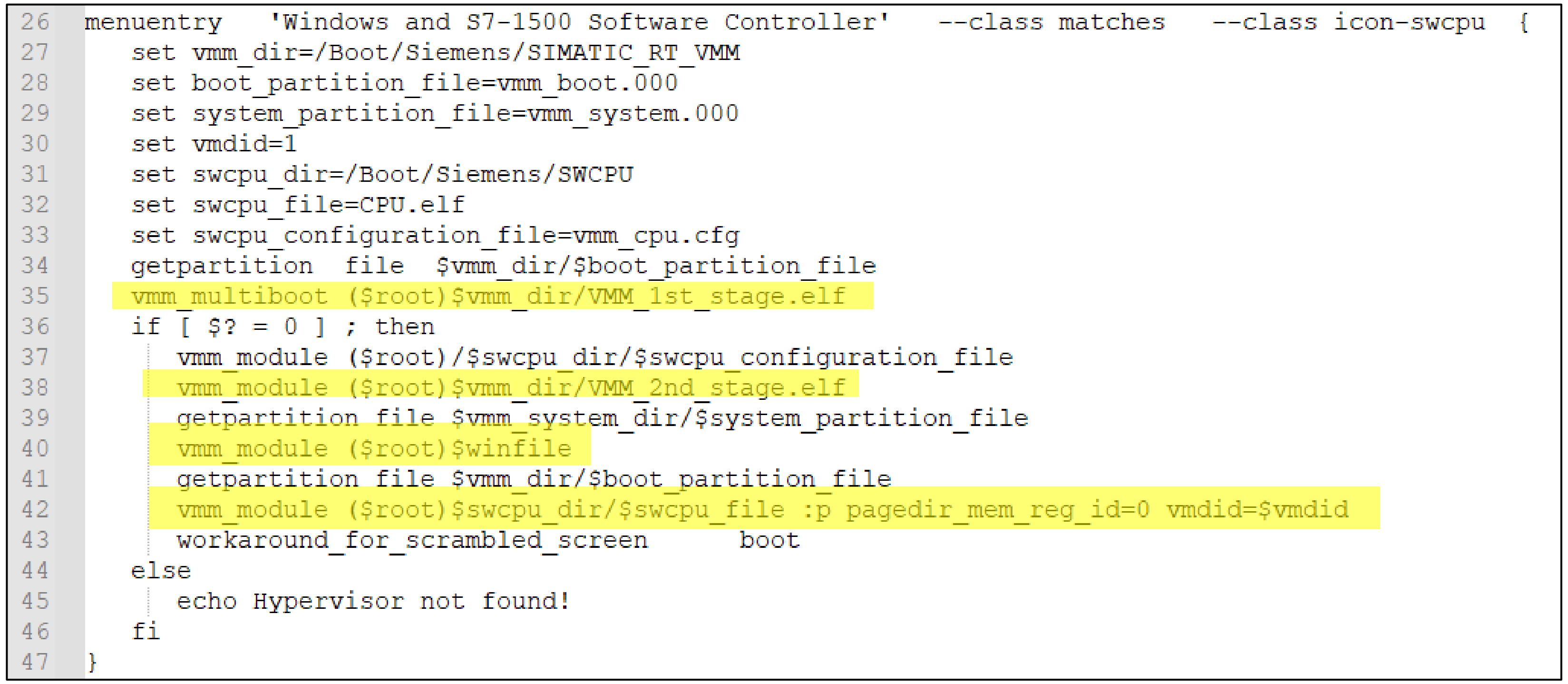


```
ubuntu@ubuntu:~$ lsblk /dev/sda
NAME    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda       8:0    0 119.2G  0 disk
├─sda1    8:1    0  58.6G  0 part
├─sda2    8:2    0  15.6G  0 part
├─sda3    8:3    0  44.6G  0 part
├─sda4    8:4    0     1K  0 part
└─sda5    8:5    0   400M  0 part
ubuntu@ubuntu:~$
```

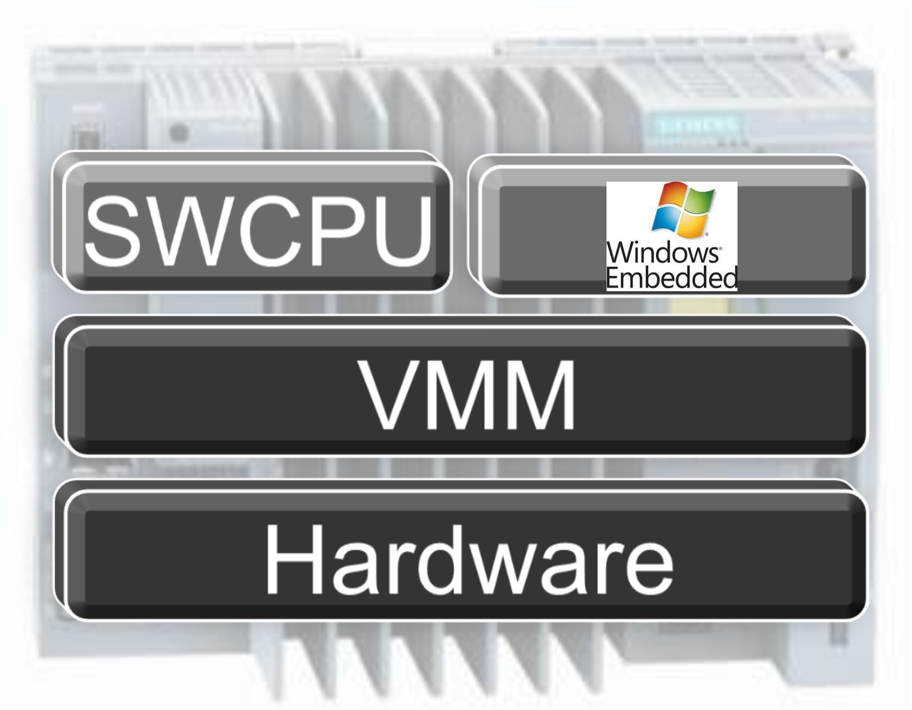
```
ubuntu@ubuntu:/mnt/TIA_project_files$ ll
total 34
drwxr-xr-x 4 root root   512 Jan  1  1970 ./
drwxr-xr-x 1 root root    60 Oct  5 01:02 ../
-r-xr-xr-x 1 root root 32768 Jan  1  1980 __LOG__*
drwxr-xr-x 2 root root   512 Jan 11  2021 ODK1500S/
-rwxr-xr-x 1 root root     7 Jan 11  2021 S7_JOB.S7S*
drwxr-xr-x 3 root root   512 Jan 11  2021 SIMATIC.S7S/
ubuntu@ubuntu:/mnt/TIA_project_files$
```

The boot process

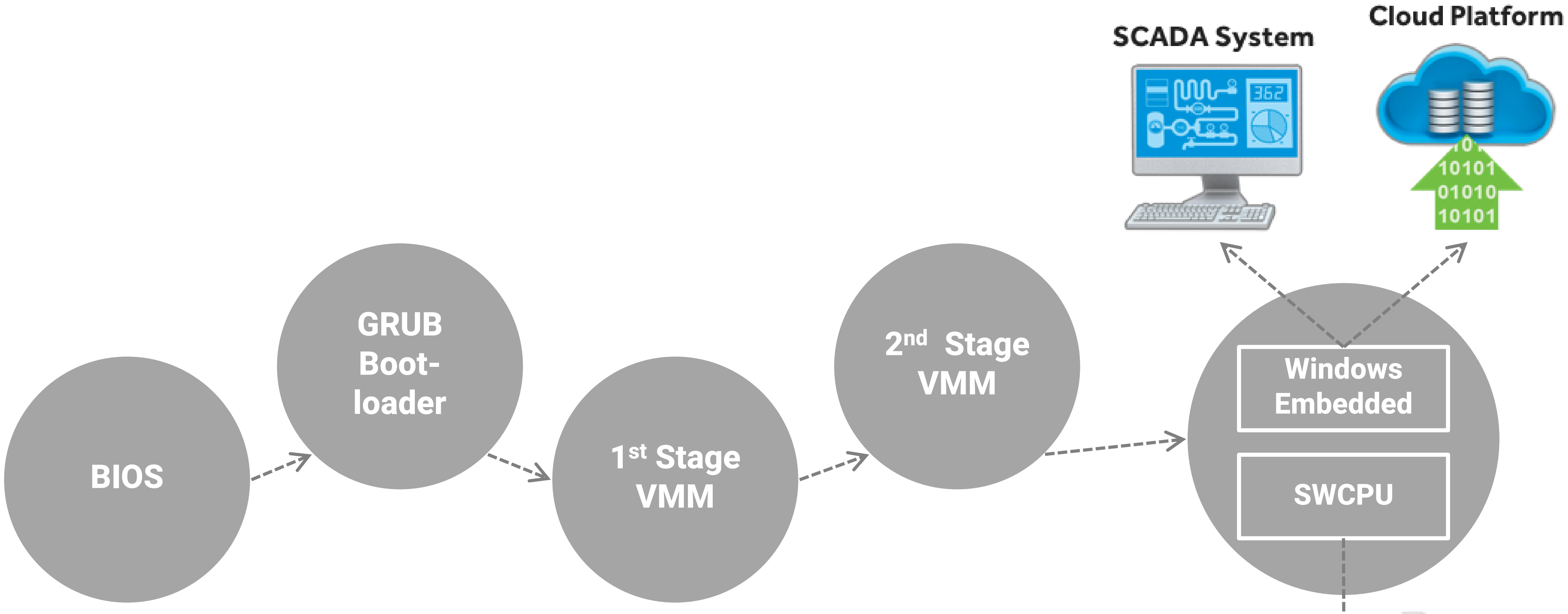
- The GRUB configuration file



```
menuentry   'Windows and S7-1500 Software Controller'   --class matches   --class icon-swcpu  {
    set vmm_dir=/Boot/Siemens/SIMATIC_RT_VMM
    set boot_partition_file=vmm_boot.000
    set system_partition_file=vmm_system.000
    set vmdid=1
    set swcpu_dir=/Boot/Siemens/SWCPU
    set swcpu_file=CPU.elf
    set swcpu_configuration_file=vmm_cpu.cfg
    getpartition  file  $vmm_dir/$boot_partition_file
    vmm_multiboot ($root)$vmm_dir/VMM_1st_stage.elf
    if [ $? = 0 ] ; then
        vmm_module ($root)/$swcpu_dir/$swcpu_configuration_file
        vmm_module ($root)$vmm_dir/VMM_2nd_stage.elf
        getpartition file $vmm_system_dir/$system_partition_file
        vmm_module ($root)$winfile
        getpartition file $vmm_dir/$boot_partition_file
        vmm_module ($root)$swcpu_dir/$swcpu_file :p pagedir_mem_reg_id=0 vmdid=$vmdid
        workaround_for_scrambled_screen      boot
    else
        echo Hypervisor not found!
    fi
}
```



Open controller boot sequence



BIOS

GRUB Boot-loader

1st Stage VMM

2nd Stage VMM

Windows Embedded

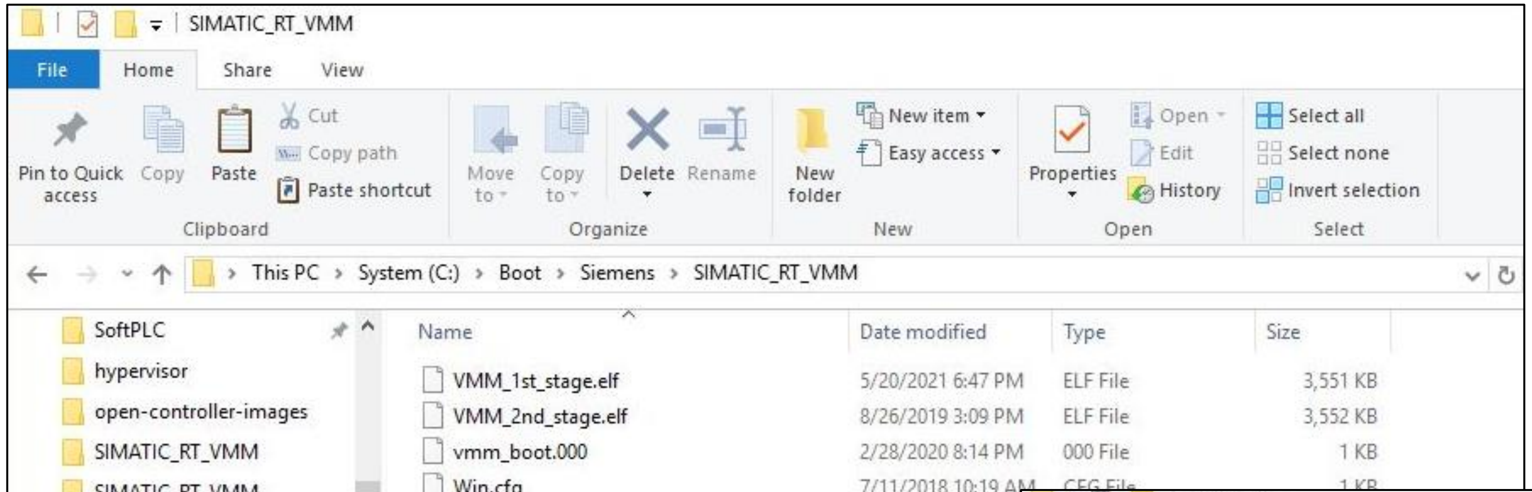
SWCPU

SCADA System

Cloud Platform

The boot process

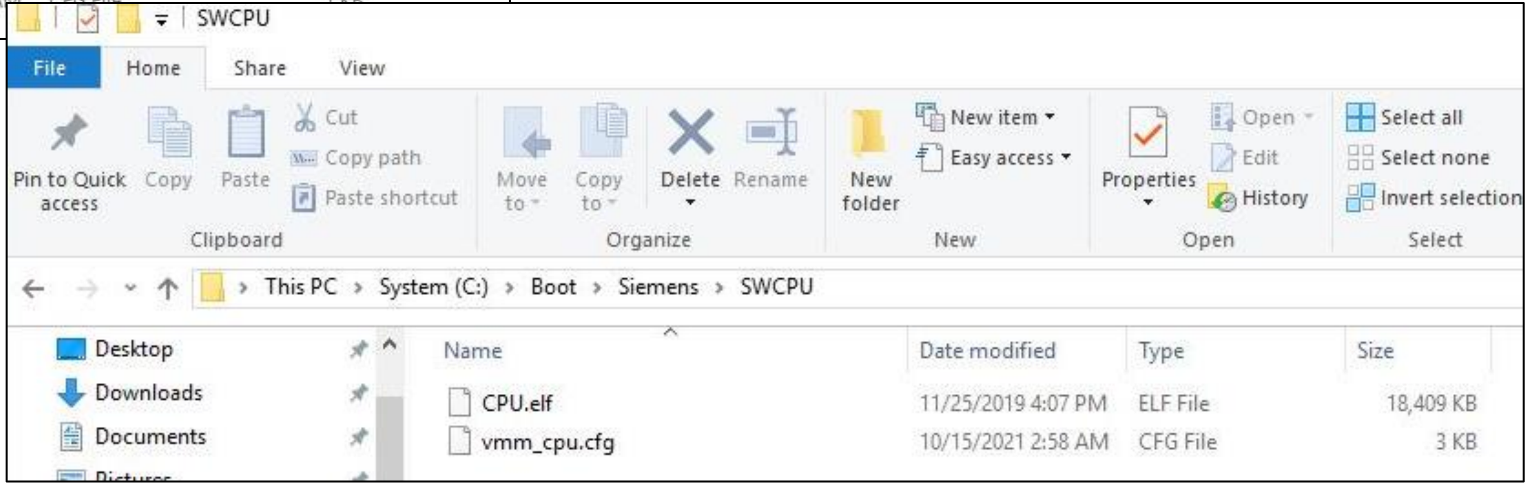
- Surprisingly, the VMM binary, grub configuration and CPU.elf files are also accessible from Windows



This PC › System (C:) › Boot › Siemens › SIMATIC_RT_VMM

Name	Date modified	Type	Size
VMM_1st_stage.elf	5/20/2021 6:47 PM	ELF File	3,551 KB
VMM_2nd_stage.elf	8/26/2019 3:09 PM	ELF File	3,552 KB
vmm_boot.000	2/28/2020 8:14 PM	000 File	1 KB
Win.cfg	7/11/2018 10:19 AM	CFG File	1 KB

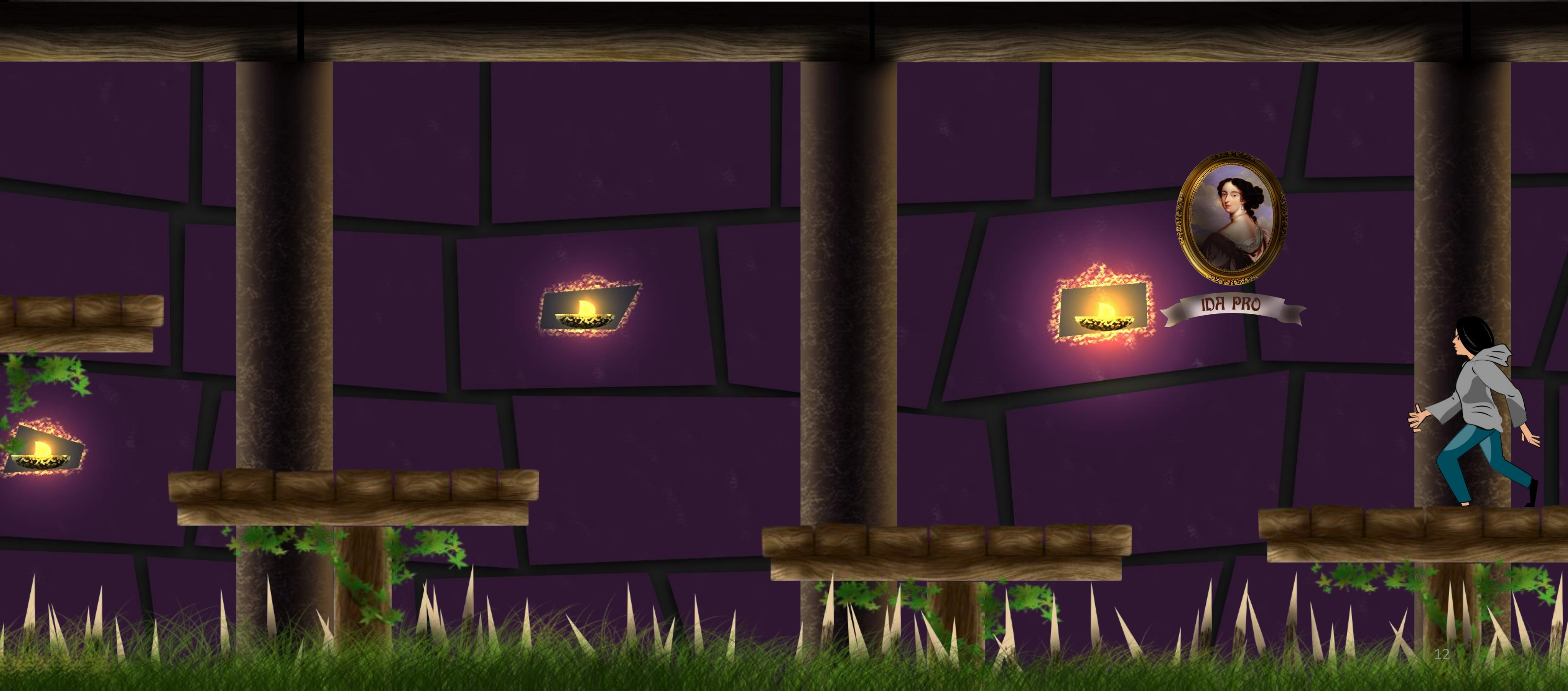
- RW by admin



This PC › System (C:) › Boot › Siemens › SWCPU

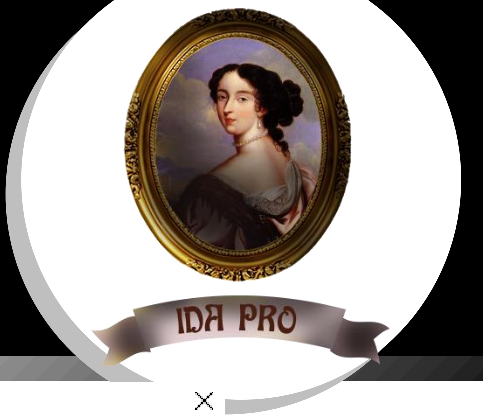
Name	Date modified	Type	Size
CPU.elf	11/25/2019 4:07 PM	ELF File	18,409 KB
vmm_cpu.cfg	10/15/2021 2:58 AM	CFG File	3 KB

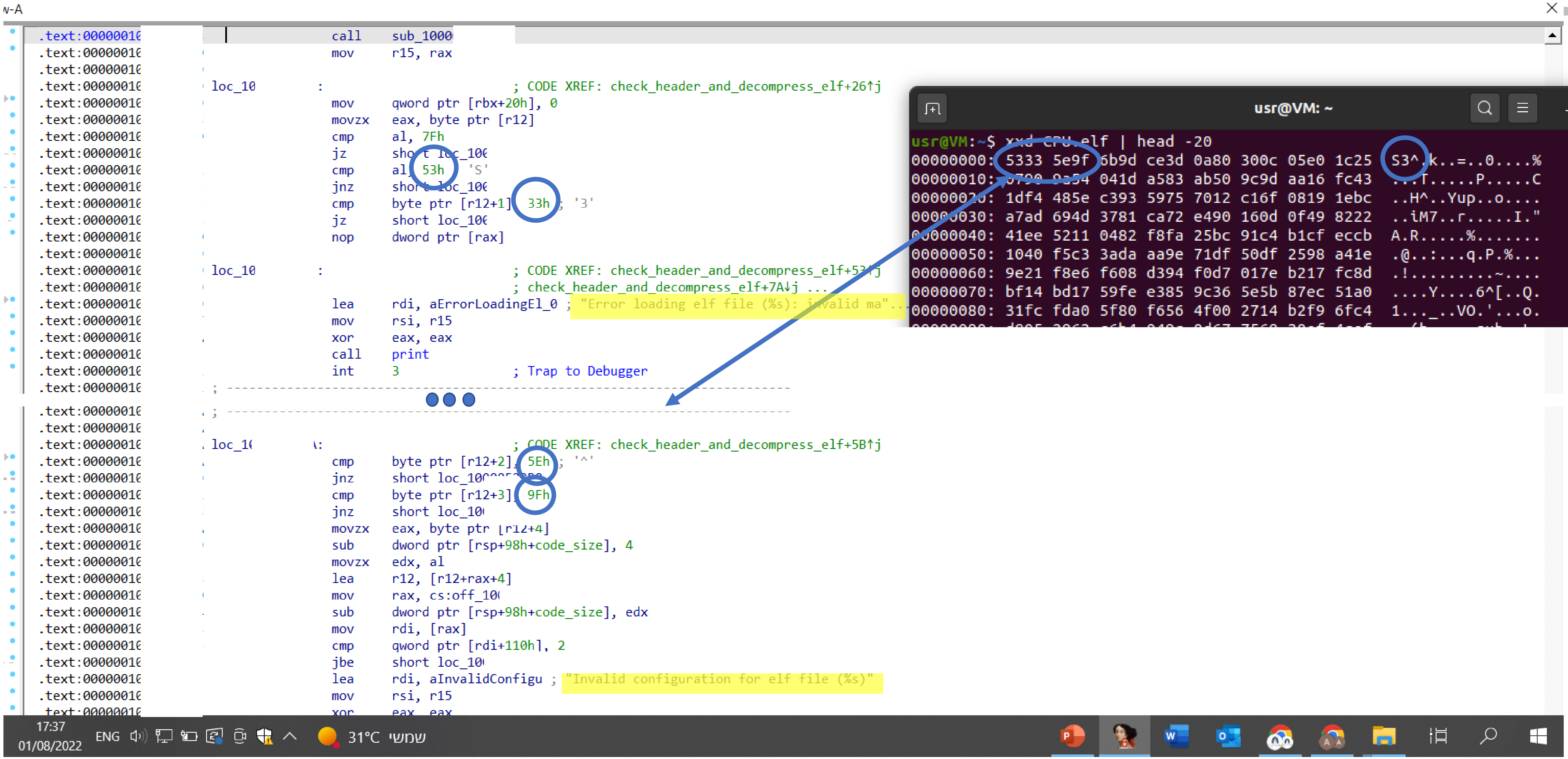
The quest…



black hat USA 2022

Finding the ELF decryption code





```
.text:0000001(                  call    sub_1000(
.text:0000001(                  mov     r15, rax
.text:0000001(
.text:0000001( loc_10       :                           ; CODE XREF: check_header_and_decompress_elf+26↑j
.text:0000001(                  mov     qword ptr [rbx+20h], 0
.text:0000001(                  movzx   eax, byte ptr [r12]
.text:0000001(                  cmp     al, 7Fh
.text:0000001(                  jz      short loc_10(
.text:0000001(                  cmp     al, 53h ; 'S'
.text:0000001(                  jnz     short loc_10(
.text:0000001(                  cmp     byte ptr [r12+1], 33h ; '3'
.text:0000001(                  jz      short loc_10(
.text:0000001(                  nop     dword ptr [rax]
.text:0000001(
.text:0000001( loc_10       :                           ; CODE XREF: check_header_and_decompress_elf+53↑j
.text:0000001(                                          ; check_header_and_decompress_elf+7A↓j ...
.text:0000001(                  lea     rdi, aErrorLoadingEl_0 ; "Error loading elf file (%s): invalid ma"...
.text:0000001(                  mov     rsi, r15
.text:0000001(                  xor     eax, eax
.text:0000001(                  call    print
.text:0000001(                  int     3               ; Trap to Debugger
.text:0000001( ; ---------------------------------------------------------------------------

.text:0000001( ; ---------------------------------------------------------------------------
.text:0000001(
.text:0000001( loc_1(       :                           ; CODE XREF: check_header_and_decompress_elf+5B↑j
.text:0000001(                  cmp     byte ptr [r12+2], 5Eh ; '^'
.text:0000001(                  jnz     short loc_10
.text:0000001(                  cmp     byte ptr [r12+3], 9Fh
.text:0000001(                  jnz     short loc_10
.text:0000001(                  movzx   eax, byte ptr [r12+4]
.text:0000001(                  sub     dword ptr [rsp+98h+code_size], 4
.text:0000001(                  movzx   edx, al
.text:0000001(                  lea     r12, [r12+rax+4]
.text:0000001(                  mov     rax, cs:off_10(
.text:0000001(                  sub     dword ptr [rsp+98h+code_size], edx
.text:0000001(                  mov     rdi, [rax]
.text:0000001(                  cmp     qword ptr [rdi+110h], 2
.text:0000001(                  jbe     short loc_10
.text:0000001(                  lea     rdi, aInvalidConfigu ; "Invalid configuration for elf file (%s)"
.text:0000001(                  mov     rsi, r15
.text:0000001(                  xor     eax, eax
```

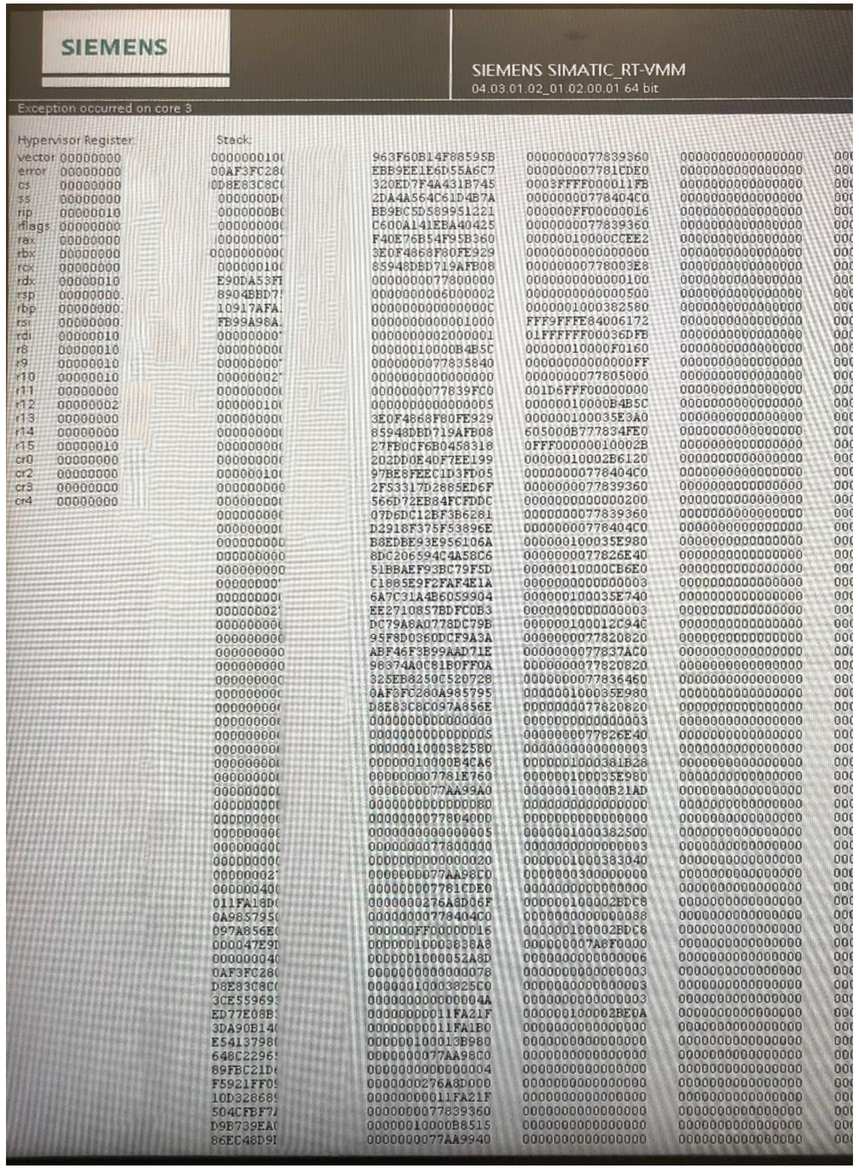
```
usr@VM:~$ xxd CPU.elf | head -20
00000000: 5333 5e9f 6b9d ce3d 0a80 300c 05e0 1c25  S3^.k..=..0....%
00000010: 0790 9a54 041d a583 ab50 9c9d aa16 fc43  ...T.....P.....C
00000020: 1df4 485e c393 5975 7012 c16f 0819 1ebc  ..H^..Yup..o....
00000030: a7ad 694d 3781 ca72 e490 160d 0f49 8222  ..iM7..r.....I."
00000040: 41ee 5211 0482 f8fa 25bc 91c4 b1cf eccb  A.R.....%.......
00000050: 1040 f5c3 3ada aa9e 71df 50df 2598 a41e  .@..:...q.P.%...
00000060: 9e21 f8e6 f608 d394 f0d7 017e b217 fc8d  .!.........~....
00000070: bf14 bd17 59fe e385 9c36 5e5b 87ec 51a0  ....Y....6^[..Q.
00000080: 31fc fda0 5f80 f656 4f00 2714 b2f9 6fc4  1..._..VO.'...o.
```

The quest...

The quest...

The “Int 3” debugger

<INT3>

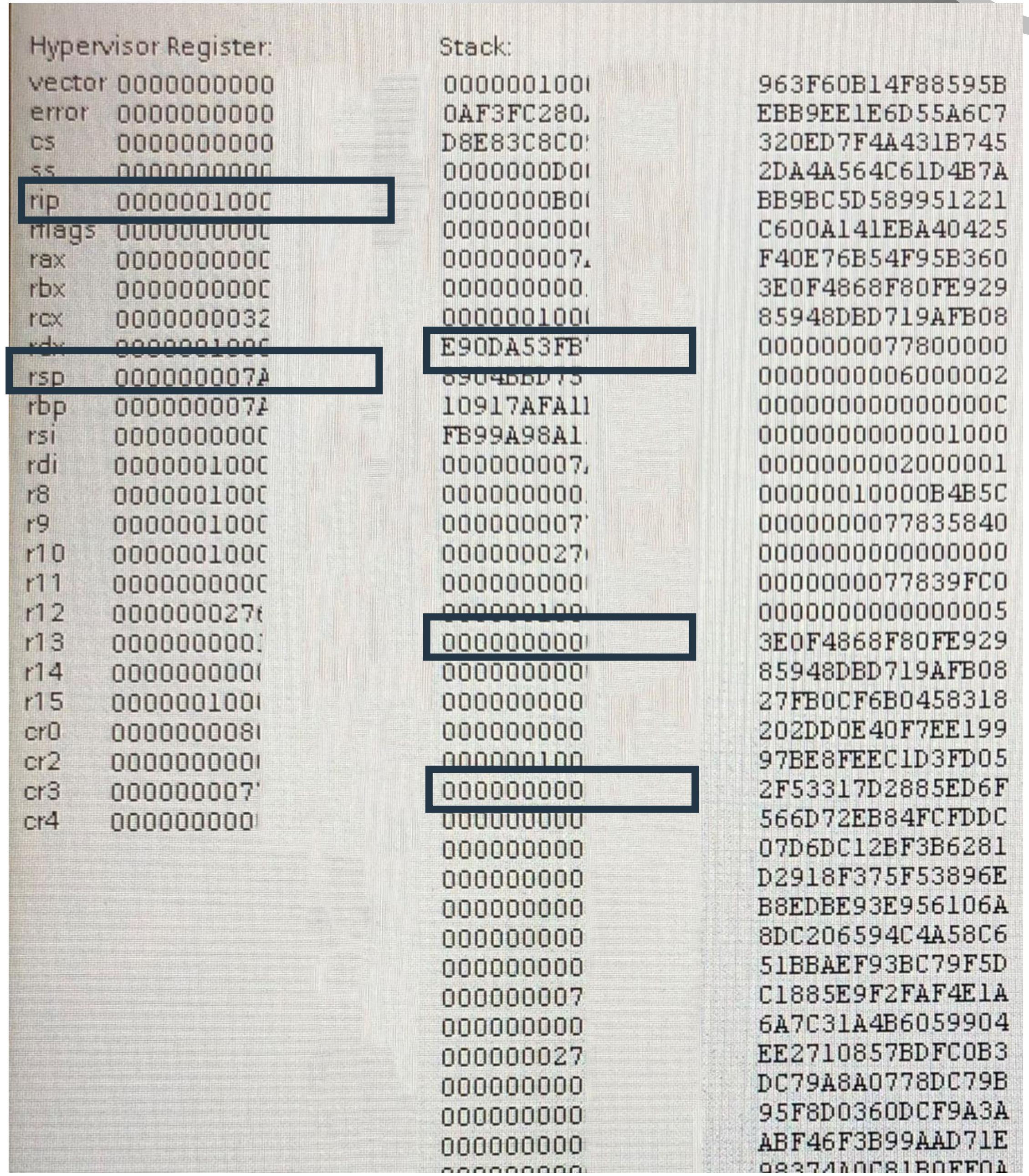


SIEMENS

SIEMENS SIMATIC_RT-VMM
04.03.01.02_01.02.00.01 64 bit

Exception occurred on core 3

```
Hypervisor Register:      Stack:
vector 00000000           00000001   963F60B14F88595B  0000000077839360  0000000000000000  000
error  00000000           00AF3FC28  EBB9EE1E6D55A6C7  000000007781CDE0  0000000000000000  000
cs     00000000           0D8E83C8C  320ED7F4A431B745  0003FFFF000011FB  0000000000000000  000
ss     00000000            0000000D  2DA4A564C61D4B7A  00000000778404C0  0000000000000000  000
rip    00000010            0000000B  BB9BC5D589951221  000000FF00000016  0000000000000000  000
rflags 00000000            00000000  C600A141EBA40425  0000000077839360  0000000000000000  000
rax    00000000            00000000  F40E76B54F95B360  00000010000CCEE2  0000000000000000  000
rbx    00000000           000000000  3E0F4868F80FE929  0000000000000000  0000000000000000  000
rcx    00000000            00000010  85948DBD719AFB08  00000000778003E8  0000000000000000  000
rdx    00000010            E90DA53F  0000000077800000  0000000000000100  0000000000000000  000
rsp    00000000            8904BBD7  0000000006000002  0000000000000500  0000000000000000  000
rbp    00000000            10917AFA  000000000000000C  0000001000382580  0000000000000000  000
rsi    00000000            FB99A98A  0000000000001000  FFF9FFFE84006172  0000000000000000  000
rdi    00000010            00000000  0000000002000001  01FFFFFF00036DFB  0000000000000000  000
r8     00000010            00000000  00000010000B4B5C  00000010000F0160  0000000000000000  000
r9     00000010            00000000  0000000077835840  00000000000000FF  0000000000000000  000
r10    00000010            00000002  0000000000000000  0000000077805000  0000000000000000  000
r11    00000000            00000000  0000000077839FC0  001D6FFF00000000  0000000000000000  000
r12    00000002            00000010  0000000000000005  00000010000B4B5C  0000000000000000  000
r13    00000000            00000000  3E0F4868F80FE929  000000100035E3A0  0000000000000000  000
r14    00000000            00000000  85948DBD719AFB08  605000B777834FE0  0000000000000000  000
r15    00000010            00000000  27FB0CF6B0458318  0FFF00000010002B  0000000000000000  000
cr0    00000000            00000000  202DD0E40F7EE199  00000010002B6120  0000000000000000  000
cr2    00000000            00000010  97BE8FEEC1D3FD05  00000000778404C0  0000000000000000  000
cr3    00000000            000000000 2F53317D2885ED6F  0000000077839360  0000000000000000  000
cr4    00000000            00000000  566D72EB84FCFDDC  0000000000000200  0000000000000000  000
                           00000000  07D6DC12BF3B6281  0000000077839360  0000000000000000  000
                           00000000  D2918F375F53896E  00000000778404C0  0000000000000000  000
                           000000000 B8EDBE93E956106A  000000100035E980  0000000000000000  000
                           000000000 8DC206594C4A58C6  0000000077826E40  0000000000000000  000
                           000000000 51BBAEF93BC79F5D  00000010000CB6E0  0000000000000000  000
                           00000000  C1885E9F2FAF4E1A  0000000000000003  0000000000000000  000
                           00000000  6A7C31A4B6059904  000000100035E740  0000000000000000  000
                           00000002  EE2710857BDFC0B3  0000000000000003  0000000000000000  000
                           00000000  DC79A8A0778DC79B  000000100012C94C  0000000000000000  000
                           000000000 95F8D0360DCF9A3A  0000000077820820  0000000000000000  000
                           000000000 ABF46F3B99AAD71E  0000000077837AC0  0000000000000000  000
                           000000000 98374A0C81B0FF0A  0000000077820820  0000000000000000  000
                           000000000 325EB8250C520728  0000000077836460  0000000000000000  000
                           00000000  0AF3FC280A985795  000000100035E980  0000000000000000  000
                           00000000  D8E83C8C097A856E  0000000077820820  0000000000000000  000
                           00000000  0000000000000000  0000000000000003  0000000000000000  000
                           00000000  0000000000000005  0000000077826E40  0000000000000000  000
                           00000000  0000001000382580  0000000000000003  0000000000000000  000
                           00000000  00000010000B4CA6  0000001000381B28  0000000000000000  000
                           00000000  000000007781E760  000000100035E980  0000000000000000  000
                           00000000  0000000077AA99A0  00000010000B21AD  0000000000000000  000
                           00000000  0000000000000080  0000000000000000  0000000000000000  000
                           00000000  0000000077804000  0000000000000000  0000000000000000  000
                           00000000  0000000000000005  0000001000382500  0000000000000000  000
                           00000000  0000000077800000  0000000000000003  0000000000000000  000
                           00000000  0000000000000020  0000001000383040  0000000000000000  000
                           00000002  0000000077AA98C0  0000000300000000  0000000000000000  000
                           00000040  000000007781CDE0  0000000000000000  0000000000000000  000
                           011FA18D  0000000276A8D06F  000000100002BDC8  0000000000000000  000
                           0A985795  00000000778404C0  0000000000000088  0000000000000000  000
                           097A856E  000000FF00000016  000000100002BDC8  0000000000000000  000
                           000047E9  00000010003838A8  000000007A8F0000  0000000000000000  000
                           00000004  0000001000052A8D  0000000000000006  0000000000000000  000
                           0AF3FC28  0000000000000078  0000000000000003  0000000000000000  000
                           D8E83C8C  00000010003825C0  0000000000000003  0000000000000000  000
                           3CE55969  000000000000004A  0000000000000003  0000000000000000  000
                           ED77E08B  00000000011FA21F  000000100002BE0A  0000000000000000  000
                           3DA90B14  00000000011FA1B0  0000000000000000  0000000000000000  000
                           E5413798  000000100013B980  0000000000000000  0000000000000000  000
                           648C2296  0000000077AA98C0  0000000000000000  0000000000000000  000
                           89FBC21D  0000000000000004  0000000000000000  0000000000000000  000
                           F5921FF0  0000000276A8D000  0000000000000000  0000000000000000  000
                           10D32868  00000000011FA21F  0000000000000000  0000000000000000  000
                           504CFBF7  0000000077839360  0000000000000000  0000000000000000  000
                           D9B739EA  00000010000B8515  0000000000000000  0000000000000000  000
                           86EC48D9  0000000077AA9940  0000000000000000  0000000000000000  000
```



```
Hypervisor Register:      Stack:
vector 0000000000         000000100    963F60B14F88595B
error  0000000000         0AF3FC280    EBB9EE1E6D55A6C7
cs     0000000000         D8E83C8C0    320ED7F4A431B745
ss     0000000000         0000000D0    2DA4A564C61D4B7A
rip    000000100C         0000000B0    BB9BC5D589951221
rflags 000000000C         000000000    C600A141EBA40425
rax    000000000C         000000007    F40E76B54F95B360
rbx    000000000C         000000000    3E0F4868F80FE929
rcx    0000000032         000000100    85948DBD719AFB08
rdx    000000100C         E90DA53FB    0000000077800000
rsp    000000007A         8904BBD75    0000000006000002
rbp    000000007A         10917AFA1    000000000000000C
rsi    000000000C         FB99A98A1    0000000000001000
rdi    000000100C         000000007    0000000002000001
r8     000000100C         000000000    00000010000B4B5C
r9     000000100C         000000007    0000000077835840
r10    000000100C         000000027    0000000000000000
r11    000000000C         000000000    0000000077839FC0
r12    0000000027         000000100    0000000000000005
r13    000000000          000000000    3E0F4868F80FE929
r14    000000000          000000000    85948DBD719AFB08
r15    000000100          000000000    27FB0CF6B0458318
cr0    000000008          000000000    202DD0E40F7EE199
cr2    000000000          000000100    97BE8FEEC1D3FD05
cr3    000000007          000000000    2F53317D2885ED6F
cr4    000000000          000000000    566D72EB84FCFDDC
                          000000000    07D6DC12BF3B6281
                          000000000    D2918F375F53896E
                          000000000    B8EDBE93E956106A
                          000000000    8DC206594C4A58C6
                          000000000    51BBAEF93BC79F5D
                          000000007    C1885E9F2FAF4E1A
                          000000000    6A7C31A4B6059904
                          000000027    EE2710857BDFC0B3
                          000000000    DC79A8A0778DC79B
                          000000000    95F8D0360DCF9A3A
                          000000000    ABF46F3B99AAD71E
```

Intermediary status check

- Found the decompression/decryption function
- Static analysis – very complex
- Debugging
  - Using Int3 debugger
    - We have the decrypted swcpu in memory
    - But cannot export it from the PLC, for analysis

Stealing the vault

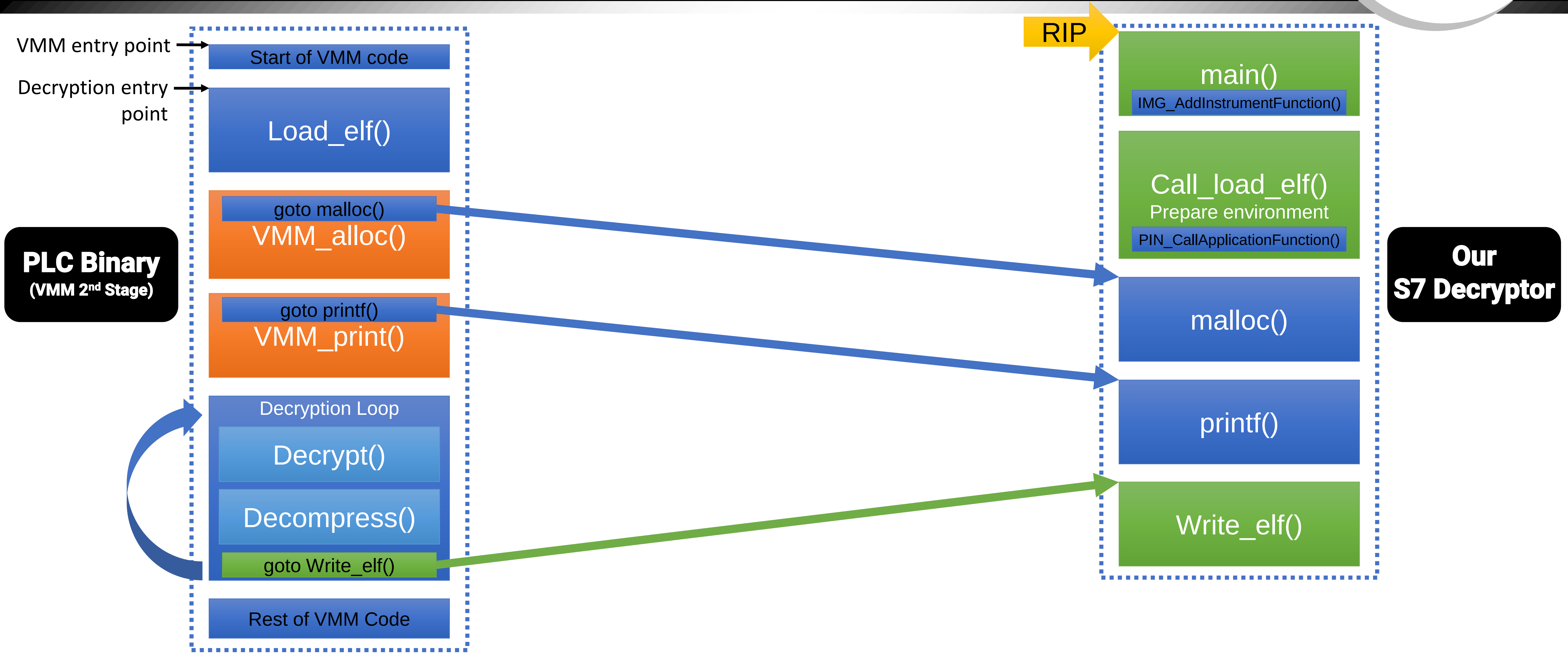
SUCCESS!

We moved the vault to our own powerful attack environment

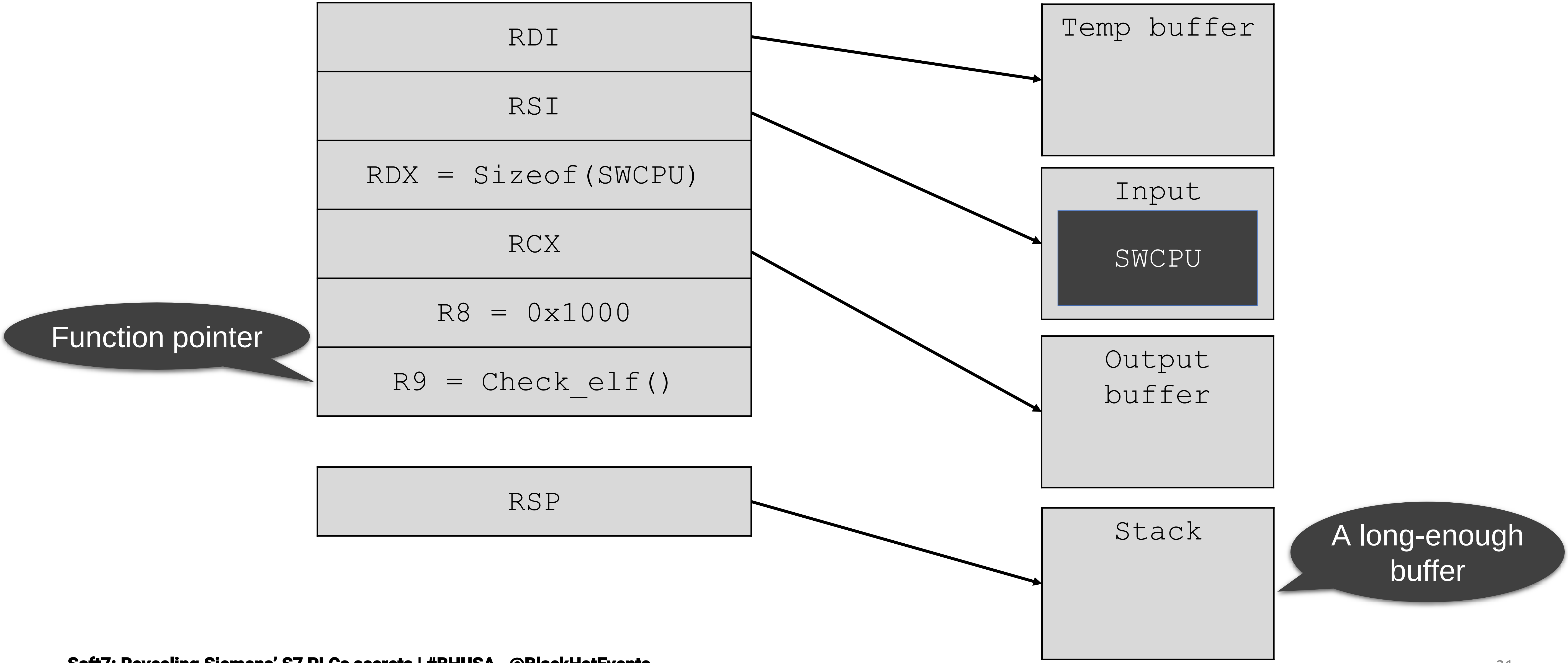
Unlocking the vault

- The VMM is an x86 binary → We run it on standard Linux (Ubuntu)
- Challenge: different execution environment
  - VMM runs in hypervisor mode, we run it in user mode
  - Siemens proprietary VMM run time library vs. standard CRT
- Solution: dynamic binary instrumentation
  - Start from a specific instruction
  - Replace VMM functions
  - Add our code
- We used Intel Pin to run the VMM decryption

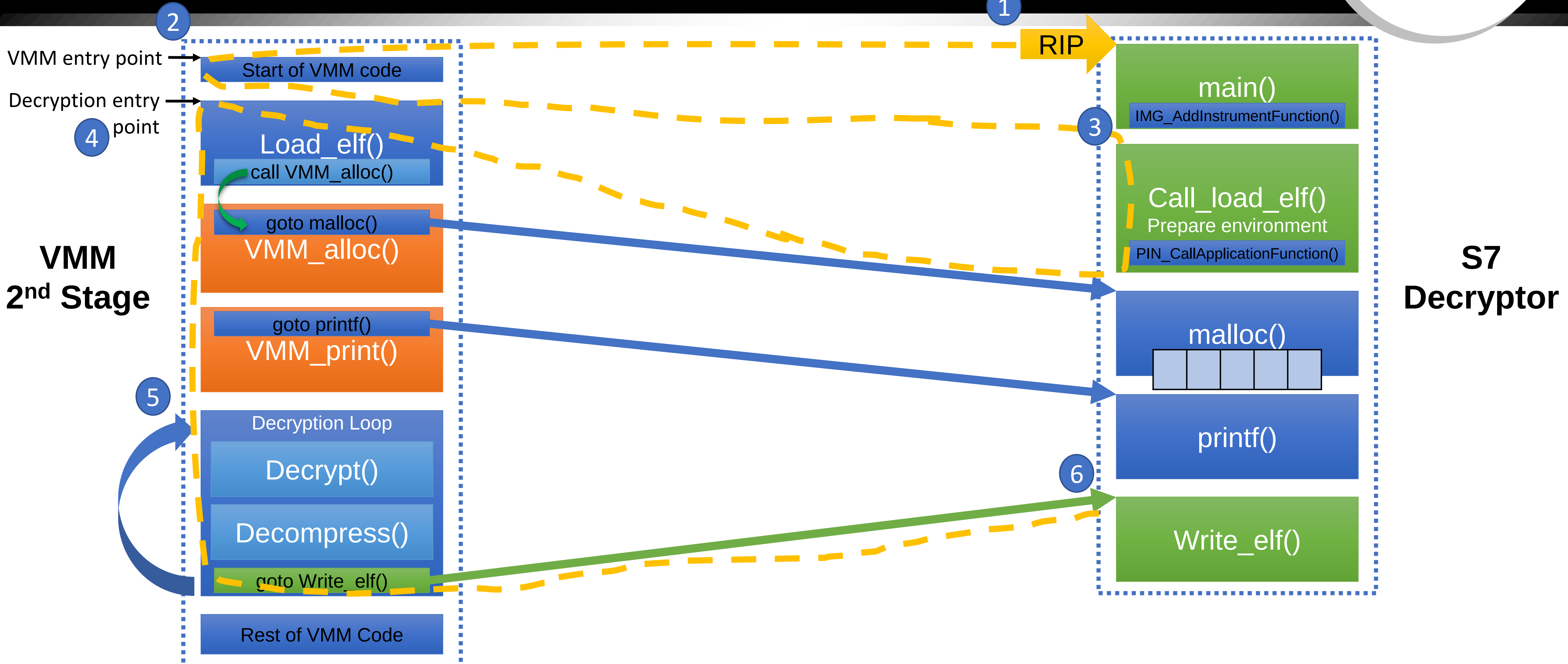
PLC binary instrumentation



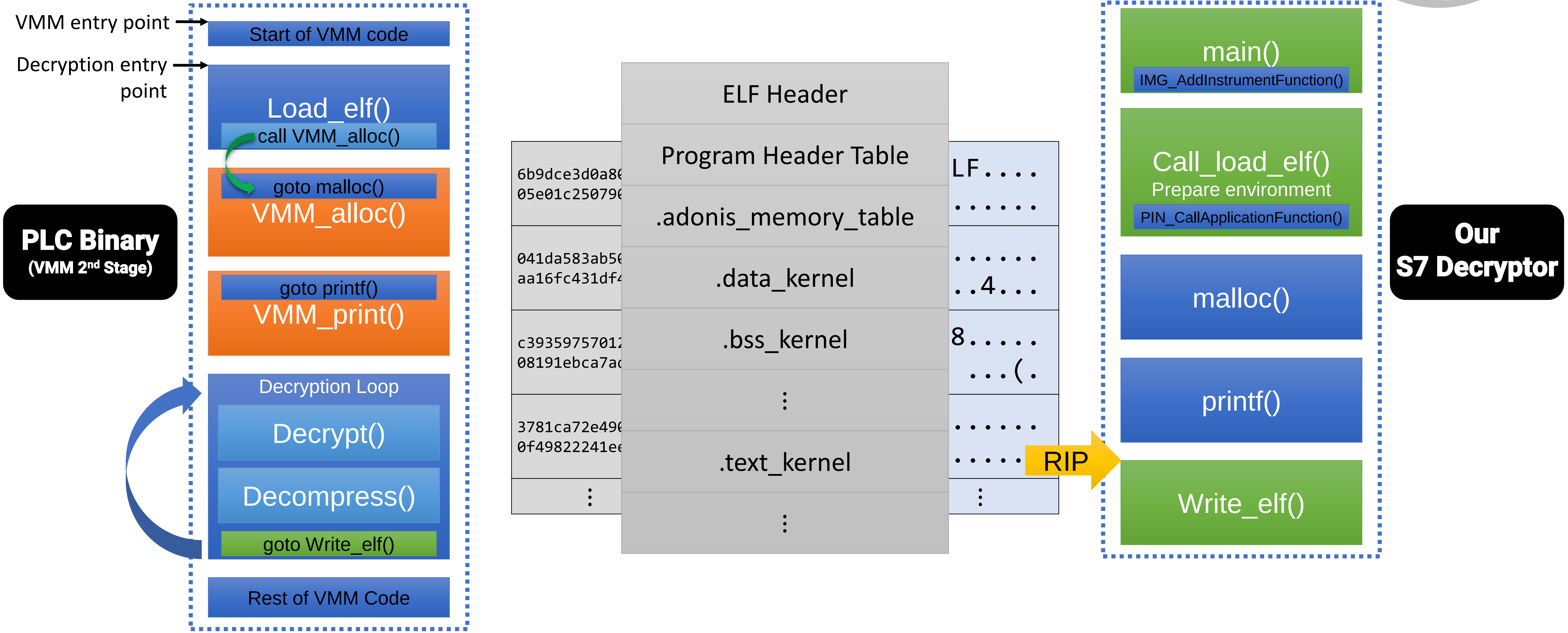
Environment



VMM binary instrumentation



PLC binary instrumentation



VMM entry point

Decryption entry point

PLC Binary
(VMM 2nd Stage)

Start of VMM code

Load_elf()
call VMM_alloc()

goto malloc()
VMM_alloc()

goto printf()
VMM_print()

Decryption Loop
Decrypt()
Decompress()
goto Write_elf()

Rest of VMM Code

```
6b9dce3d0a8
05e01c25079
041da583ab5
aa16fc431df
c3935975701
08191ebca7a
3781ca72e49
0f49822241e
⋮
```

ELF Header
Program Header Table
.adonis_memory_table
.data_kernel
.bss_kernel
⋮
.text_kernel
⋮

```
LF....
......
......
..4...
8.....
...(.
......
.....
⋮
```

RIP

main()
IMG_AddInstrumentFunction()

Call_load_elf()
Prepare environment
PIN_CallApplicationFunction()

malloc()

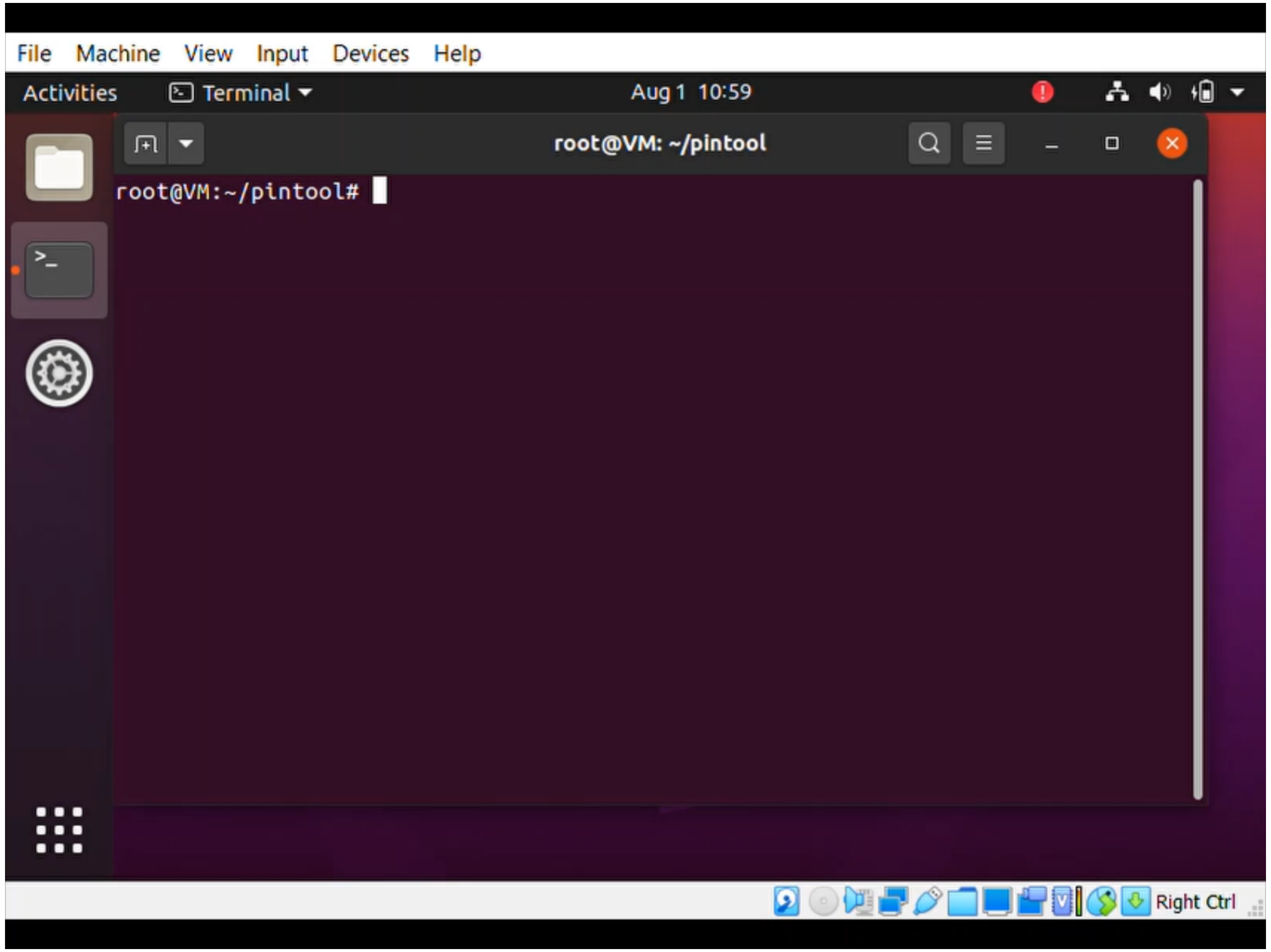
printf()

Write_elf()

Our S7 Decryptor

Demo

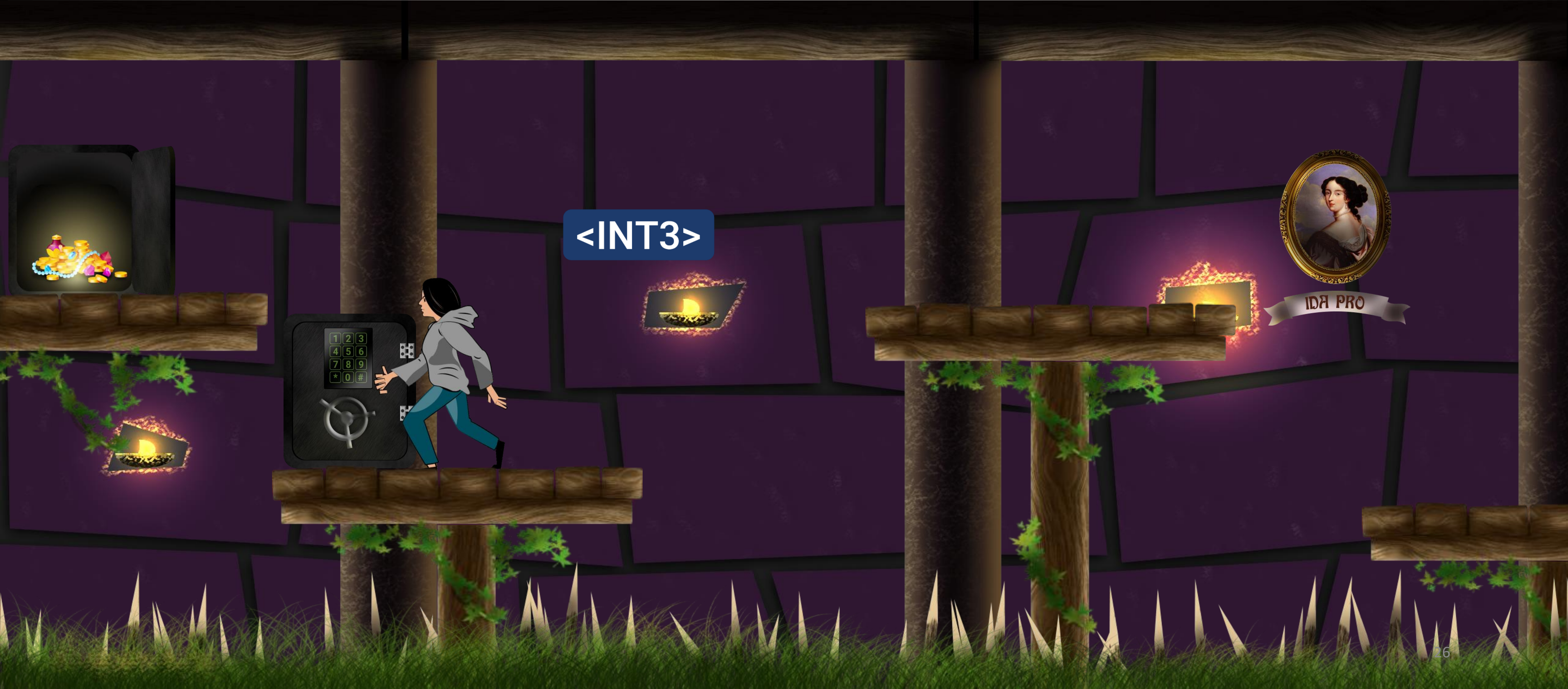
- Running PLC binary (VMM 2nd stage) on our Ubuntu machine



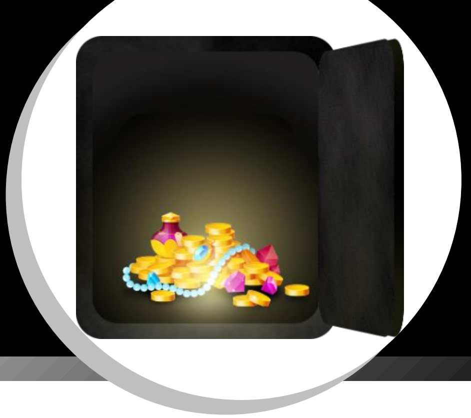
```
root@VM:~/pintool#
```

The quest...

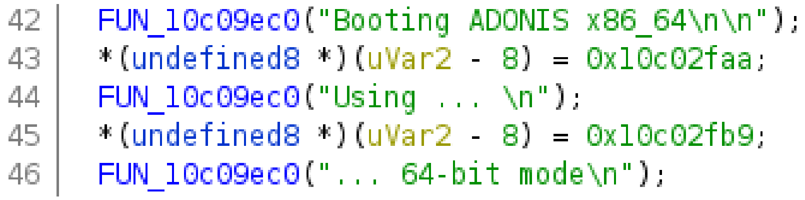
The quest...



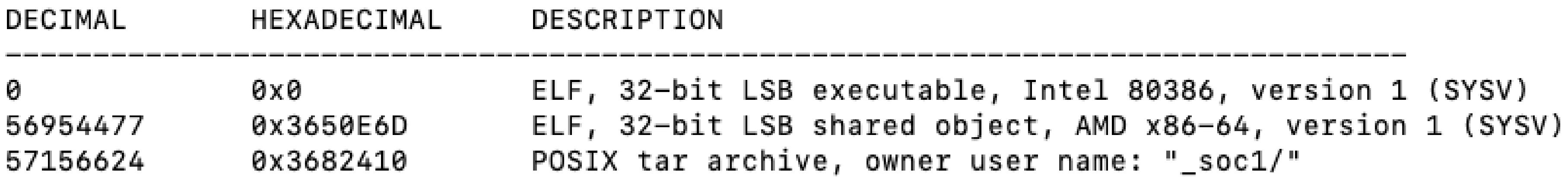
The Decrypted firmware



- Our initial research shows that SWCPU is based on the Adonis Linux
- Contains far more than the basic kernel + PLC code:
  - Stand-alone libc.so
  - openSSL
  - tar archive called “winac_bb_soc1” with a MIPS ELF inside
  - Strings from other S7 Simatic PLCs



```
FUN_10c09ec0("Booting ADONIS x86_64\n\n");
*(undefined8 *)(uVar2 - 8) = 0x10c02faa;
FUN_10c09ec0("Using ... \n");
*(undefined8 *)(uVar2 - 8) = 0x10c02fb9;
FUN_10c09ec0("... 64-bit mode\n");
```



```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             ELF, 32-bit LSB executable, Intel 80386, version 1 (SYSV)
56954477      0x3650E6D       ELF, 32-bit LSB shared object, AMD x86-64, version 1 (SYSV)
57156624      0x3682410       POSIX tar archive, owner user name: "_soc1/"
```

The quest...

The quest...

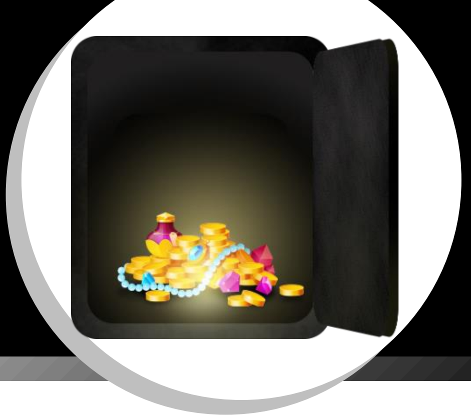
The quest...

SECURE BOOT

TPM

DM CRYPT

Mitigations



TPM

Separating the key from the code:
prevents decryption with PIN

SECURE BOOT

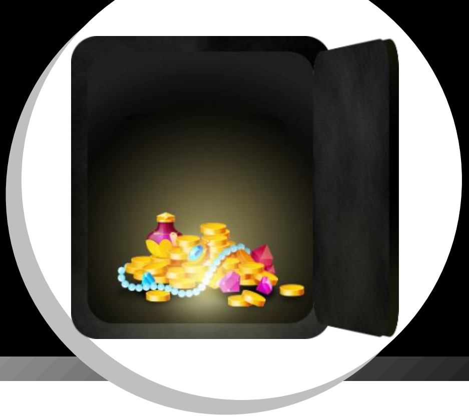
Prevents INT3 debugging

Prevents Ubuntu booting

DM CRYPT

Prevents static code reversing

Customer impact



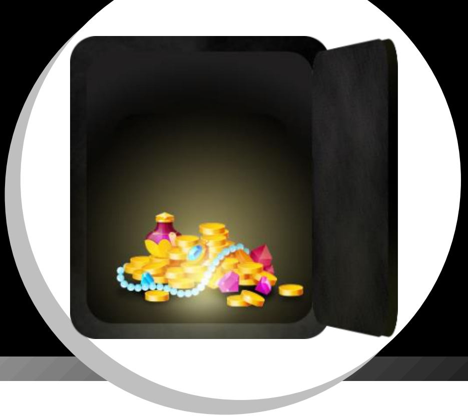
! PLC firmware leakage exposes the full Simatic S7 product line

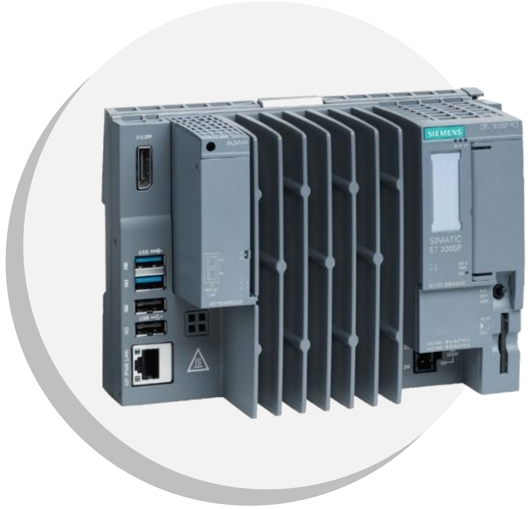
- Via exploitation of known vulnerabilities
- The horses may have already left the stable…

! Recent finding (future publication)

- An attacker who gains admin rights on the Windows VM can replace the PLC firmware with his own crafted rogue PLC firmware
- We shared the full details with Siemens

Soft7 Summary





31%

Siemens PLC market share (2019)

Deployment

Power plants, water facilities, transportation systems, nuclear reactors

Firmware leakage

↓

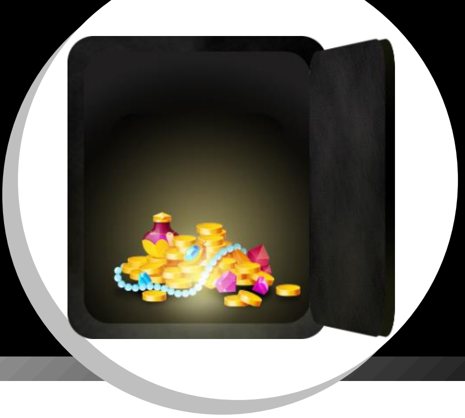
Exposure to known unpatched vulnerabilities

A design flaw

↓

No easy solution

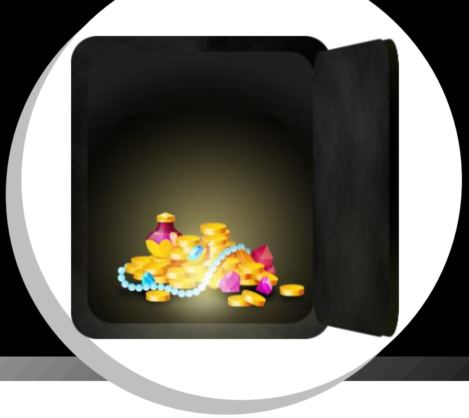
Message to the community



Message to the security & research community

- Secure binding to hardware and large-scale key management are tough operational problems
- This is challenge to the security & research community
  - Especially important since ICS architecture currently shifting from walled garden to open and cloud-oriented environments
- A solution is crucial!!!

Message to customers



Message to the customers of all ICS vendors

- You are the assets owners!
- You will suffer from the impact!
- Demand the security you need from the ICS vendors!
  - Otherwise, you get “generic” security features that do not fit your full requirements

Thank you!

Sara Bitan | Alon Dankner

Sarab@cycloak.com | Alon.Dankner@cs.technion.ac.il