FORESCOUT

Automated Cybersecurity Across Your Digital Terrain

NAKATOMI SPACE

Lateral Movement as Level 1 Post-Exploitation in OT

Jos Wetzels
Security Researcher, Forescout

Who am I?

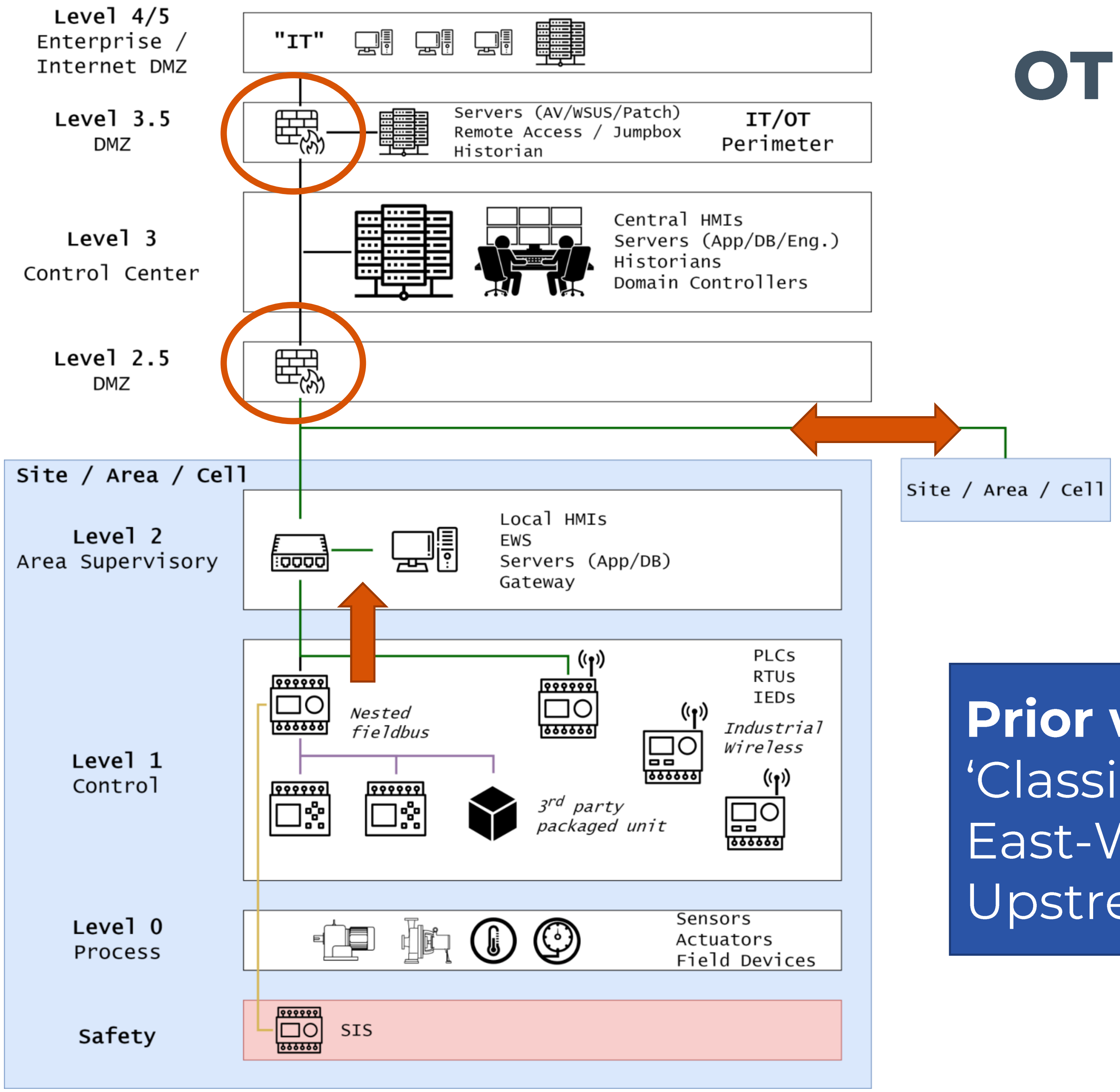
- Security Researcher @ Forescout
  - Focus on OT / IoT, embedded systems in general
- Joined Forescout in 2018 via SecurityMatters
  - OT-focused cybersecurity vendor
- Previously, researcher @ University of Twente (NL)
- Frequent speaker at security conferences, such as Black Hat, DEF CON, CCC, HITB, etc.

jos.wetzels@forescout.com

twitter: @s4mvartaka

https://www.linkedin.com/in/jos-w-61539598/

OT Lateral movement



Prior work

'Classical' perimeters at L3.5/L2.5

East-West @ L2+

Upstream to L2

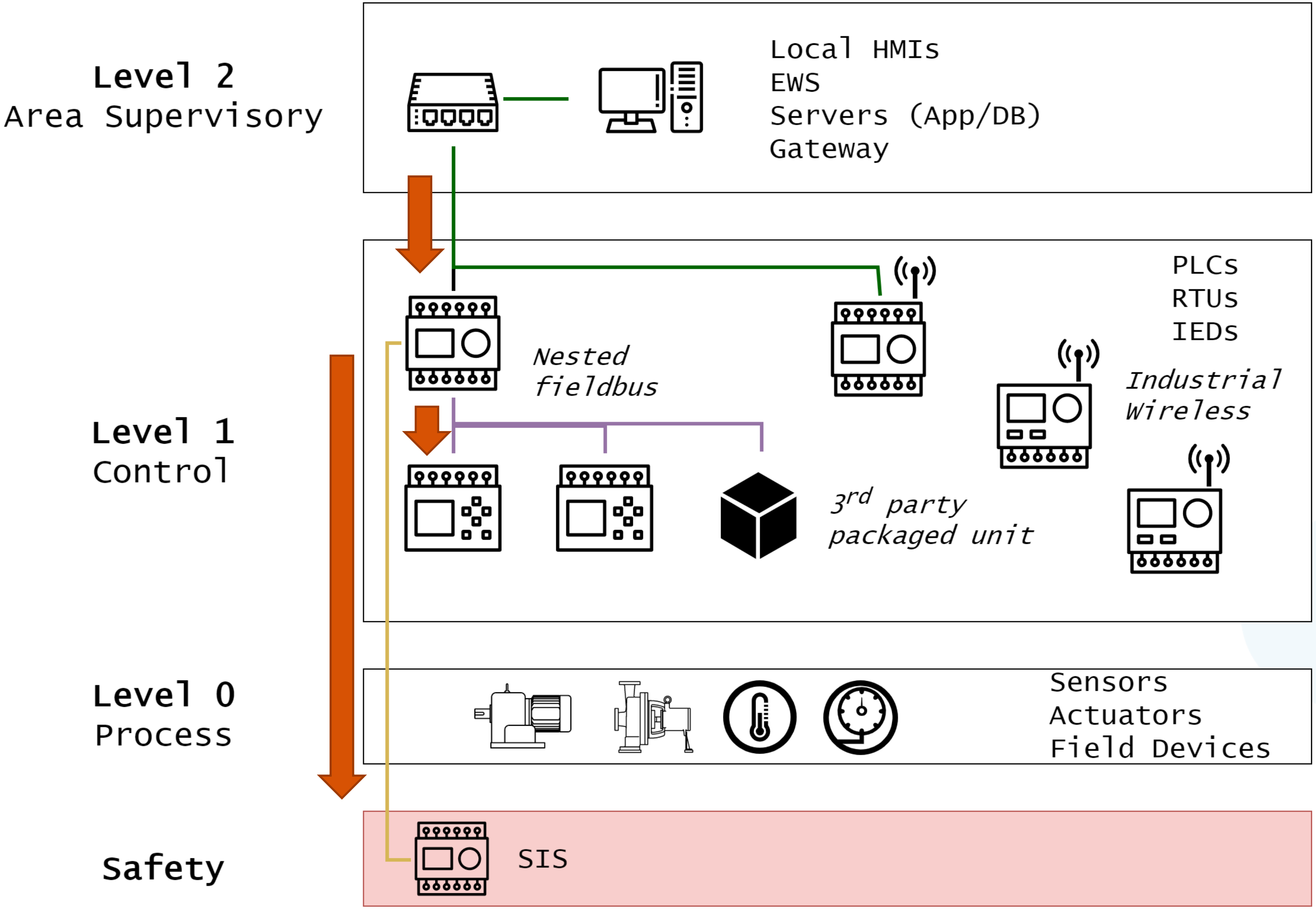
Nakatomi (Cyber)Space*

- **OT has lot of “*network crawl space*”**
  - Highly complex systems-of-systems
- **Lot of stuff beyond typical Ethernet networks**
  - Fieldbus networks (PROFIBUS/NET, CANopen, etc.)
  - RF networks (WirelessHART, 900MHz, TETRA WAN)
  - PTP links to 3rd party systems
- **Often complete lack of visibility**
  - Perimeters at this level often unacknowledged
  - Little awareness of possibility for maneuver
  - No ability to detect activity

Architectural elements with latent potential to enable traversing it in unintended and often overlooked ways

* https://bldgblog.com/2010/01/nakatomi-space/

Deep Lateral Movement



Focus
East-West @ L1
"Deep downstream"

Examples
Nested Fieldbus
Industrial Wireless
3rd Party PUs
BPCS / SIS links

Different Networks
Non-routable (PTP)
Non-IP (serial, RF)

Going *through* may require L1 RCE

- Has been demonstrated against many vendors now

- Several L1 post-exploitation TTPs have been publicly explored
  - Persistence[1,2,3,4,5]
  - Privilege escalation[2]
  - Evasion[2,6]
  - C2[7]
  - Exfiltration[8,9]
  - "*OT payloads*" (impair process control + inhibit response)[1,3,10,11,12]

- But no lateral movement at L1

[1] MITRE S0603, [2] MITRE S1009, [3] MITRE S1006
[4] INCONTROLLER: New State-Sponsored Cyber Attack Tools Target Multiple ICS - Mandiant
[5] Cyber-Security in Building Automation Systems - Forescout
[6] The Race to Native Code Execution in PLCs – T. Keren et al.
[7] Evil bubbles – M. Krotofil et al.
[8] Exfiltrating reconnaissance data from air-gapped ICS/SCADA networks - D. Atch et al.
[9] Greetings from the '90s – M. Krotofil et al.
[10] Ghost in the PLC – A. Abbasi et al.
[11] A diet of poisoned fruit – J. Wetzels et al.
[12] Hey, My Malware Knows Physics! – L. Garcia et al.

Why bother? Reason #1: Perimeter crossing

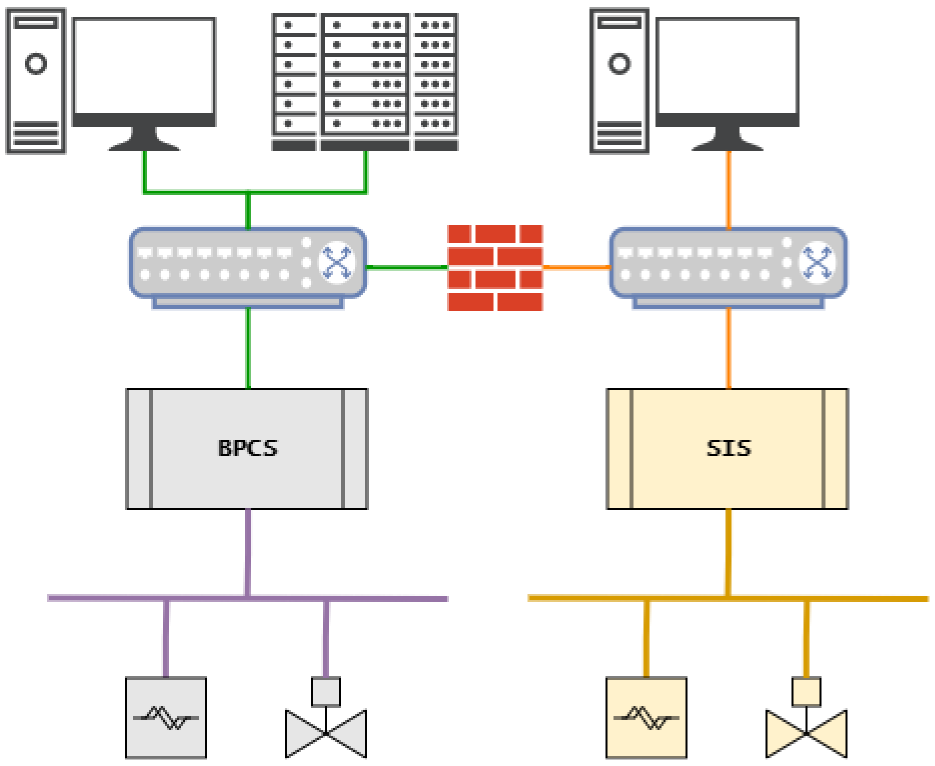
I need to move across hardened or unacknowledged perimeters

If a device is multi-homed (incl. serial/RF/etc. links) between different zones, it is a ***perimeter*** device

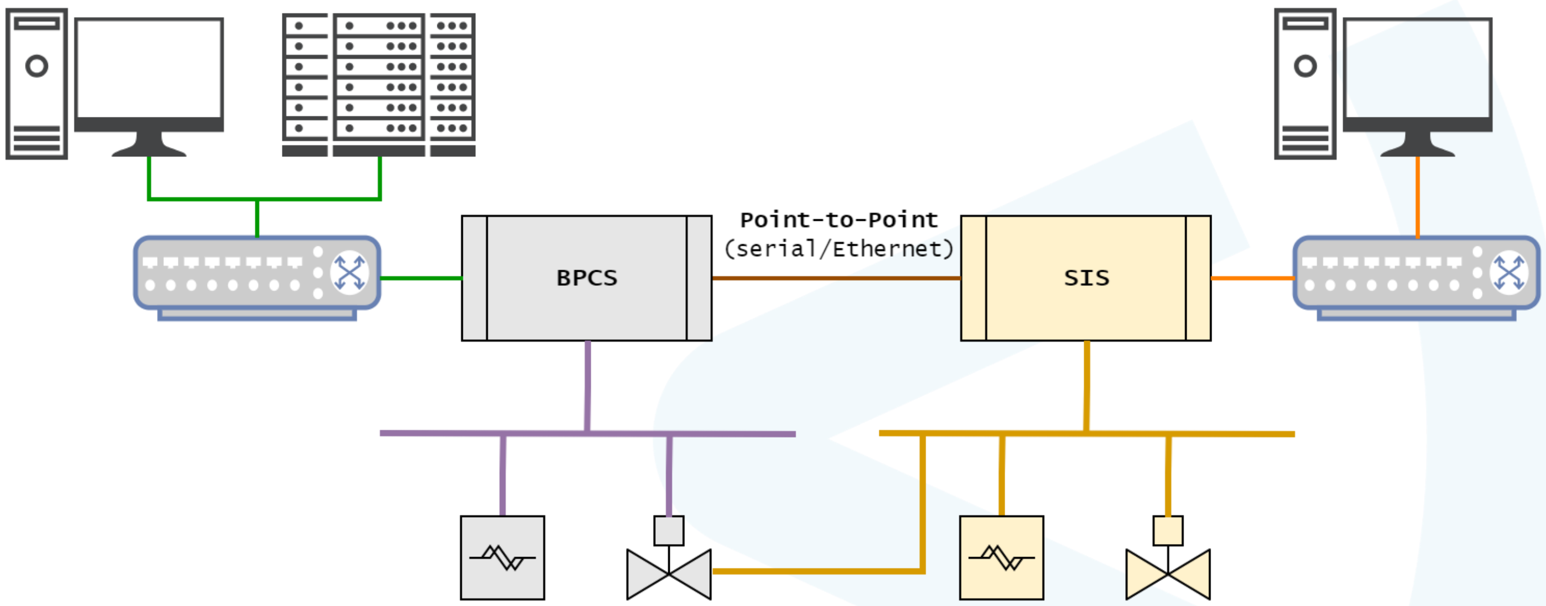
BPCS / SIS architectures

Can be generalized to any *distinct* but *interacting* control systems

Integrated



Interfaced / “Shared”



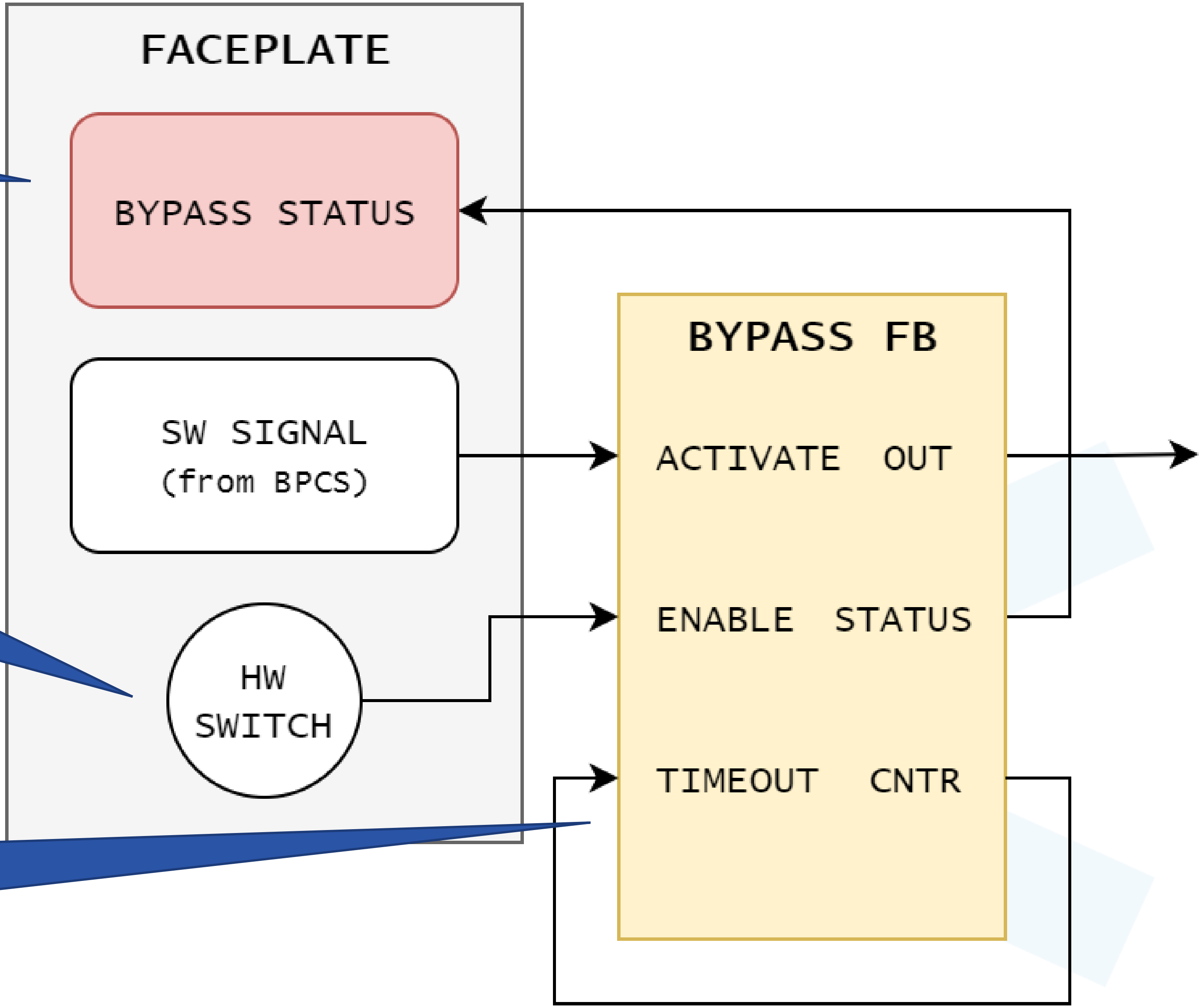
Example: SIS Bypasses

Bypass sensor, actuator, SIF

Needs to be enabled before activation

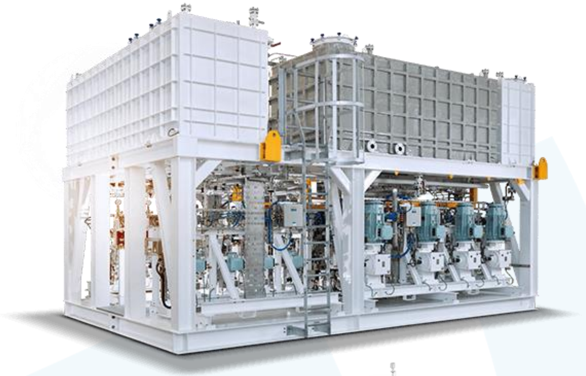
Not from BPCS

Deep SIS access to enable BPASS + disable limits



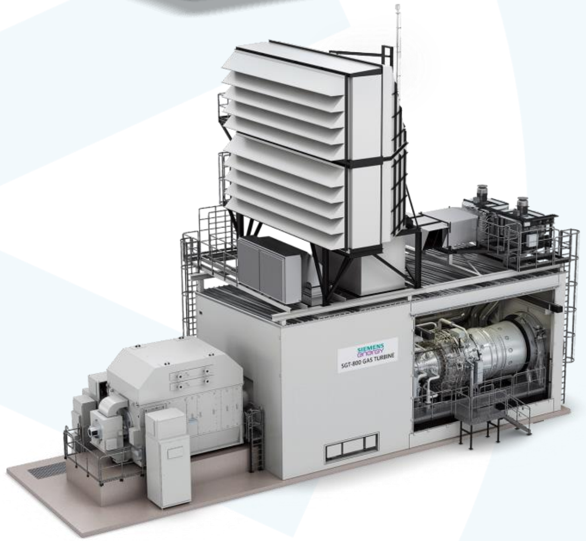
Packaged Units (PU)

- Blackbox control systems with specific function
  - HVAC, chemical injection, water treatment, gas turbine
  - Can range from subsystem to entire plant



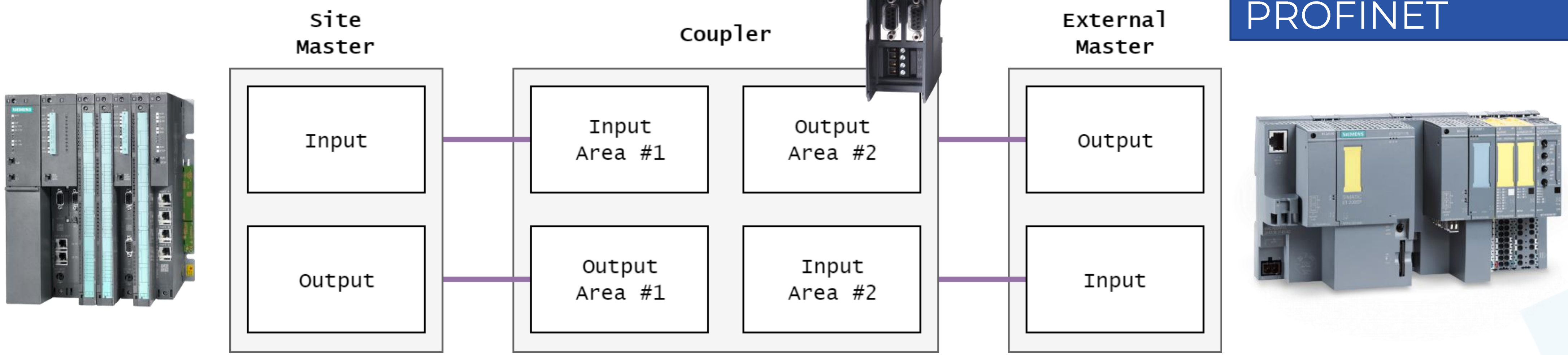
- Control/Monitoring interface to PCN/SCADA
  - Limited PVs / setpoints exposed
  - No direct control over PU internals

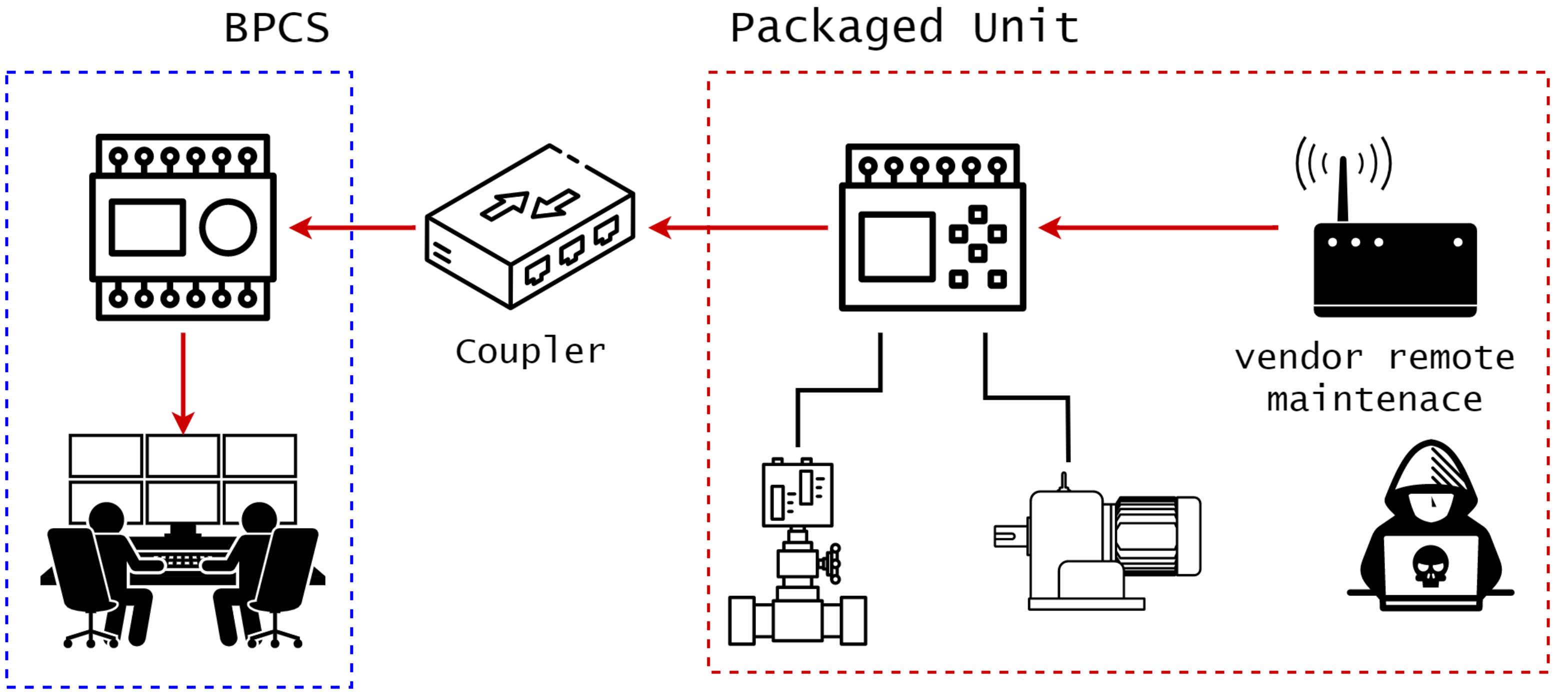
- Maintenance often done by 3rd party
  - E.g. cellular modem
  - Indirectly exposes PCN to external connectivity



Example: Fieldbus Couplers

Connect e.g. PROFIBUS DP ↔ PROFINET





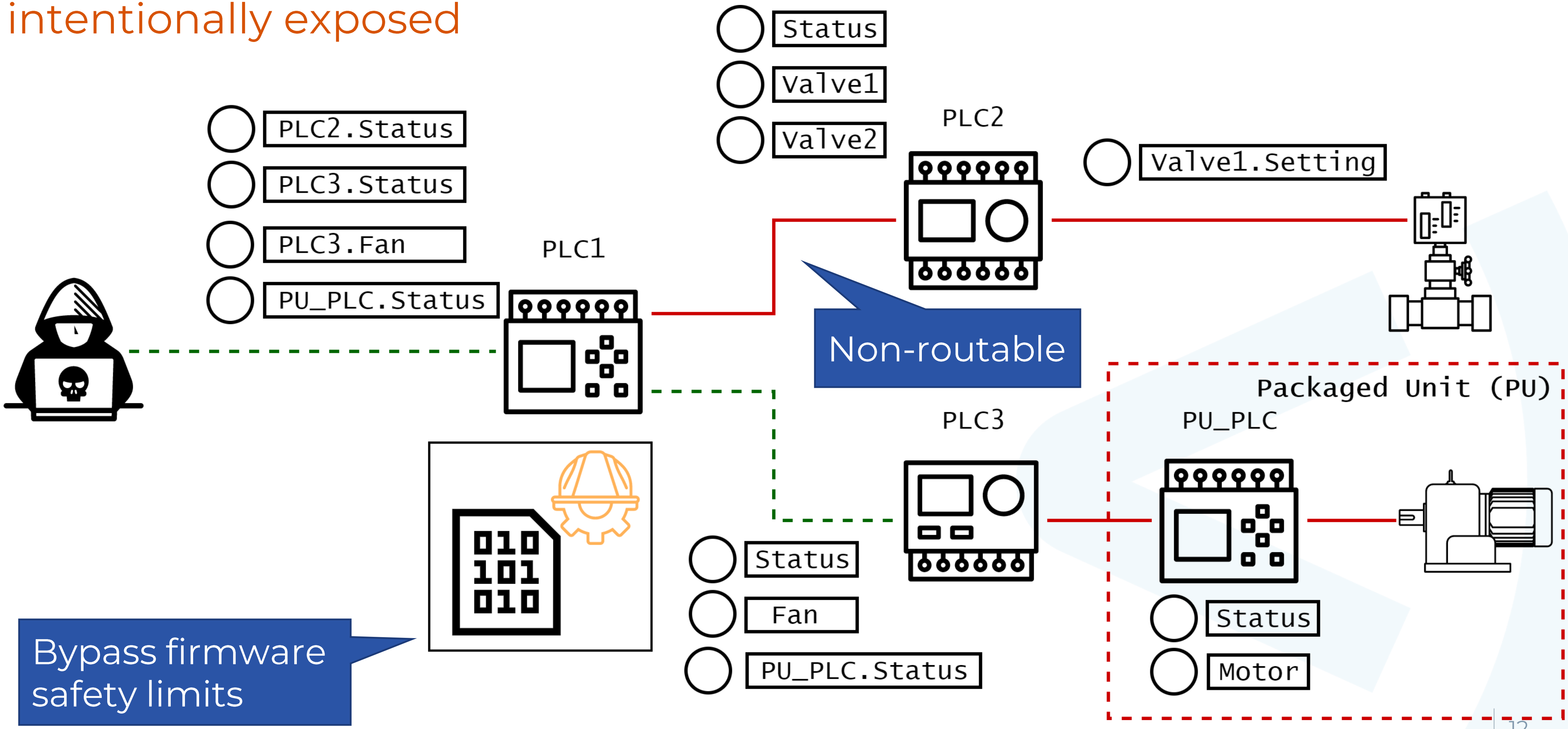
Often considered sufficient perimeter due to limited capabilities

Used to be ‘dumb’
Increasingly ‘smart’

Perimeter assumptions not evaluated for new attack surface

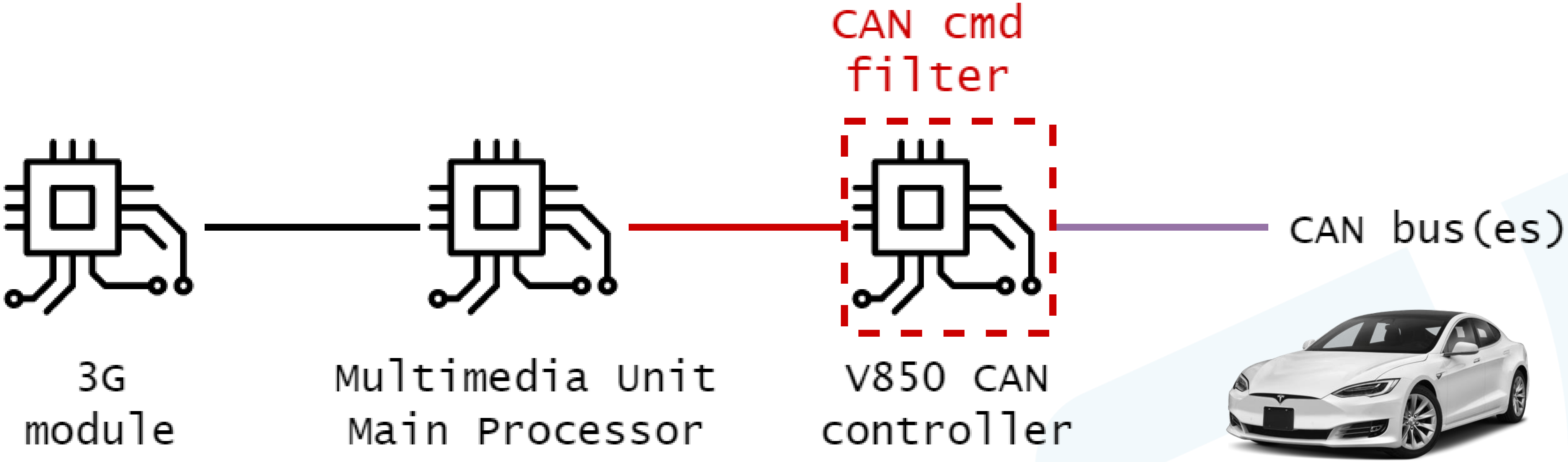
Why bother? Reason #2: Granular control

I want to talk to nested devices in a way not possible through what's intentionally exposed



Very common in automotive exploitation

RCE on CAN controller / GW to bypass filter → unrestricted CAN access



* C. Miller et al. (2015), ** Tencent Keenlab (2016), *** Computest (2018)

What do vendors & standards say?

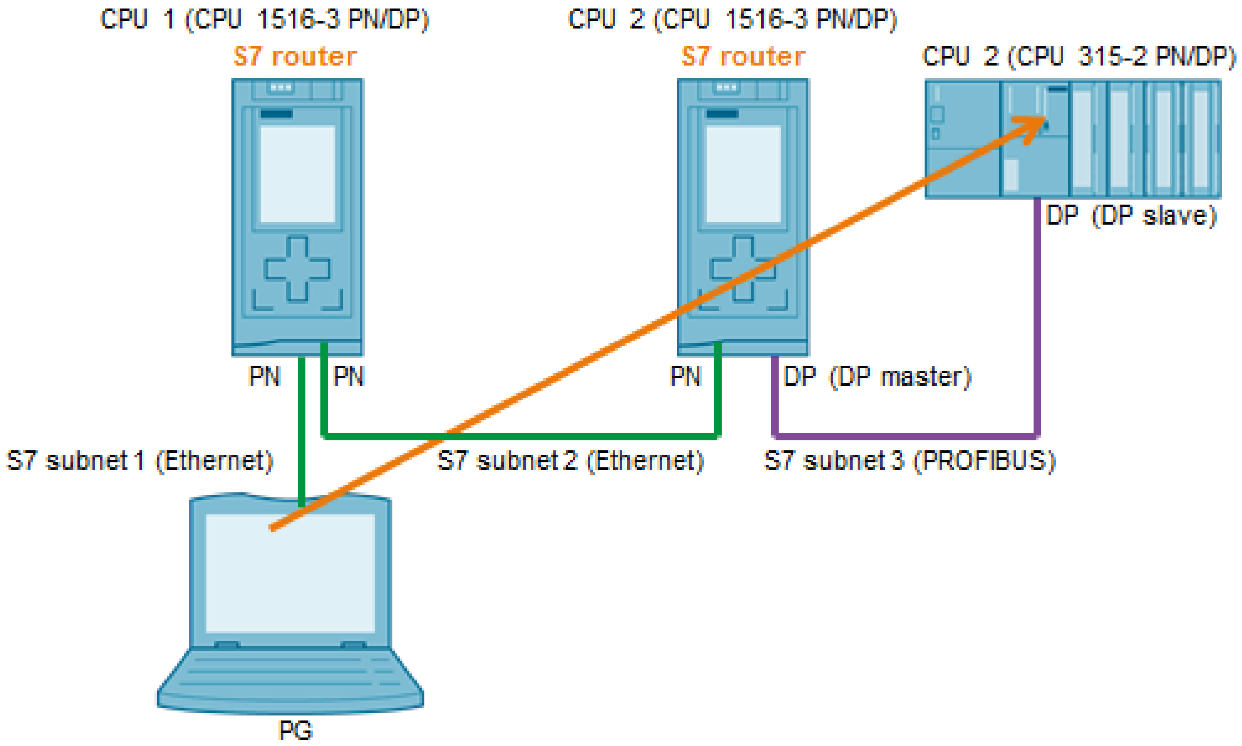
- General acceptance of integrated, interfaced and common architectures
- Usual segmentation advice
- Non-routable or serial PTP links are seen as sufficiently segmented
- Little attention to backplane security in multi-zone devices

> There is a conduit between the BPCS zone and the SIS zone, presumably to provide read only data from the SIS to the BPCS. In this case segregation has been achieved by using a dedicated point-to-point serial connection. Note that the discrete I/O also shown

* IEC 61508 intro, ** HSE OG-0086 ed2, *** NERC CIP-005-6, **** NIST SP 800-82r2

Example L1 lateral movement TTPs

1. Routing & Encapsulation / Tunneling
   - Many OT protocols deeply routable across media (e.g. CIP, S7comm, EtherCAT, Modbus MEI, HART pass-through)



2. In-band code downloads
   - Especially dangerous if ‘hot’

3. Direct memory manipulation
   - Lack of ACLs or bounds checks
   - Write to code or control-flow data → RCE

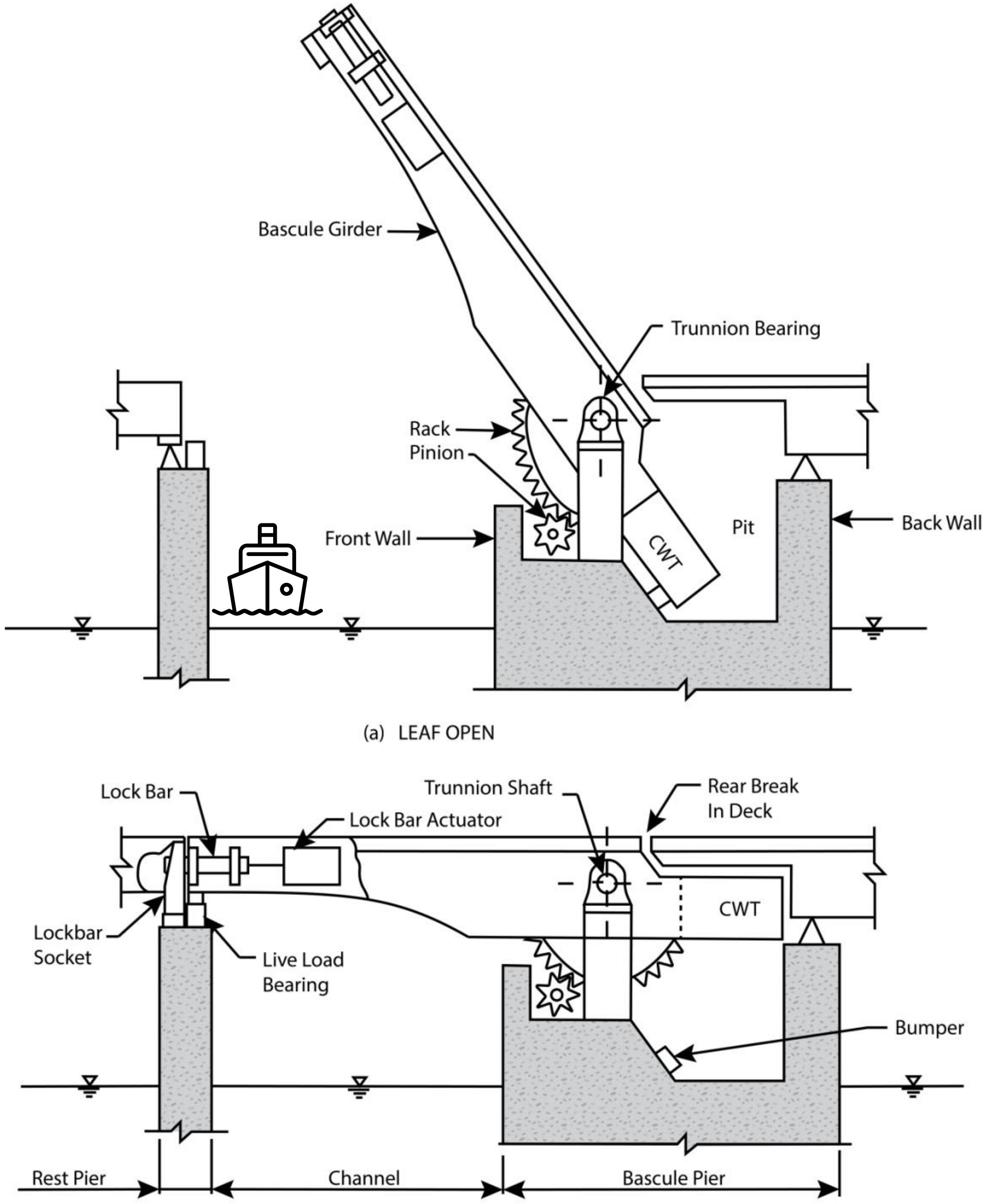
4. Protocol stack vulnerabilities
   - A serial fieldbus parsers written in C is... still a parser written in C
   - Sometimes occur deep down in system: e.g. during protocol conversion[1]

Image sources: Siemens

[1] Lost in Translation – M. Balduzzi et al.

Proof-of-Concept Scenario

Scenario: Movable Bridge

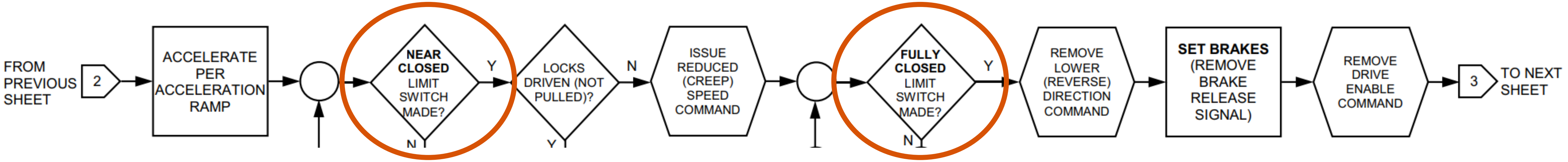


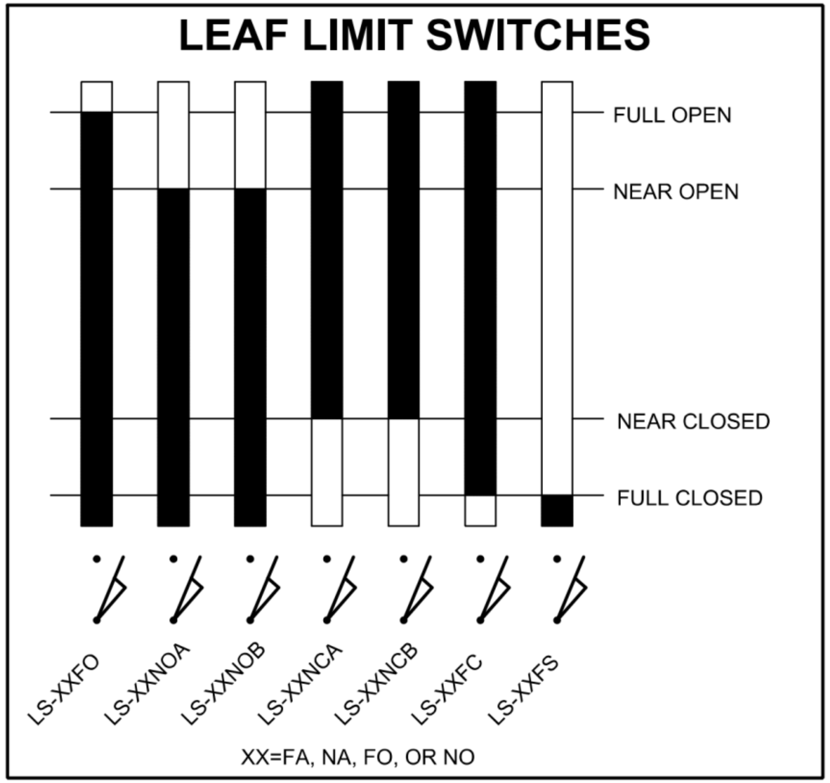
(a) LEAF OPEN

(b) LEAF CLOSED

* Bridge Maintenance Reference Manual – FDOT, structurae.net

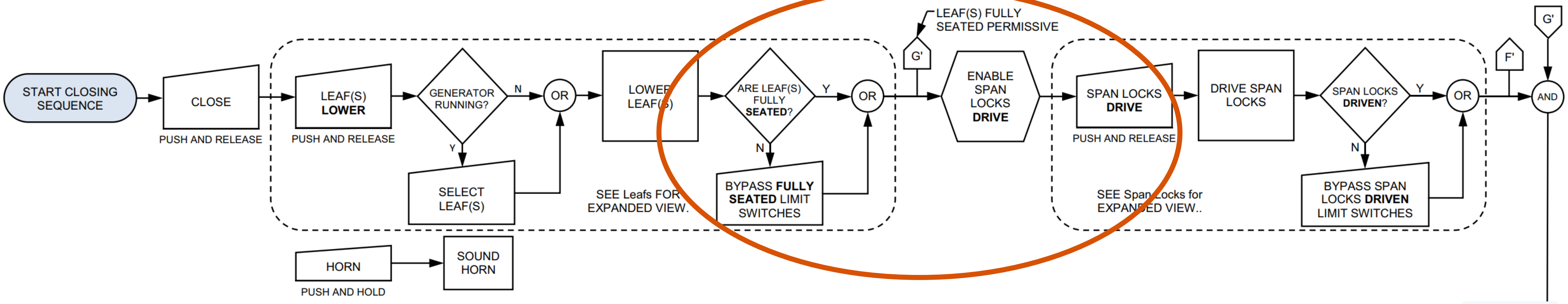
Bridge closing sequence – Limit Switches





* Bridge Maintenance Reference Manual - FDOT

Bridge closing sequence – Lock Bar



Attack Scenarios

▶ Scenario 1 : Close at full speed, hit bearings

– *Without* decel. to creep speed
– Lock bar driven before closing

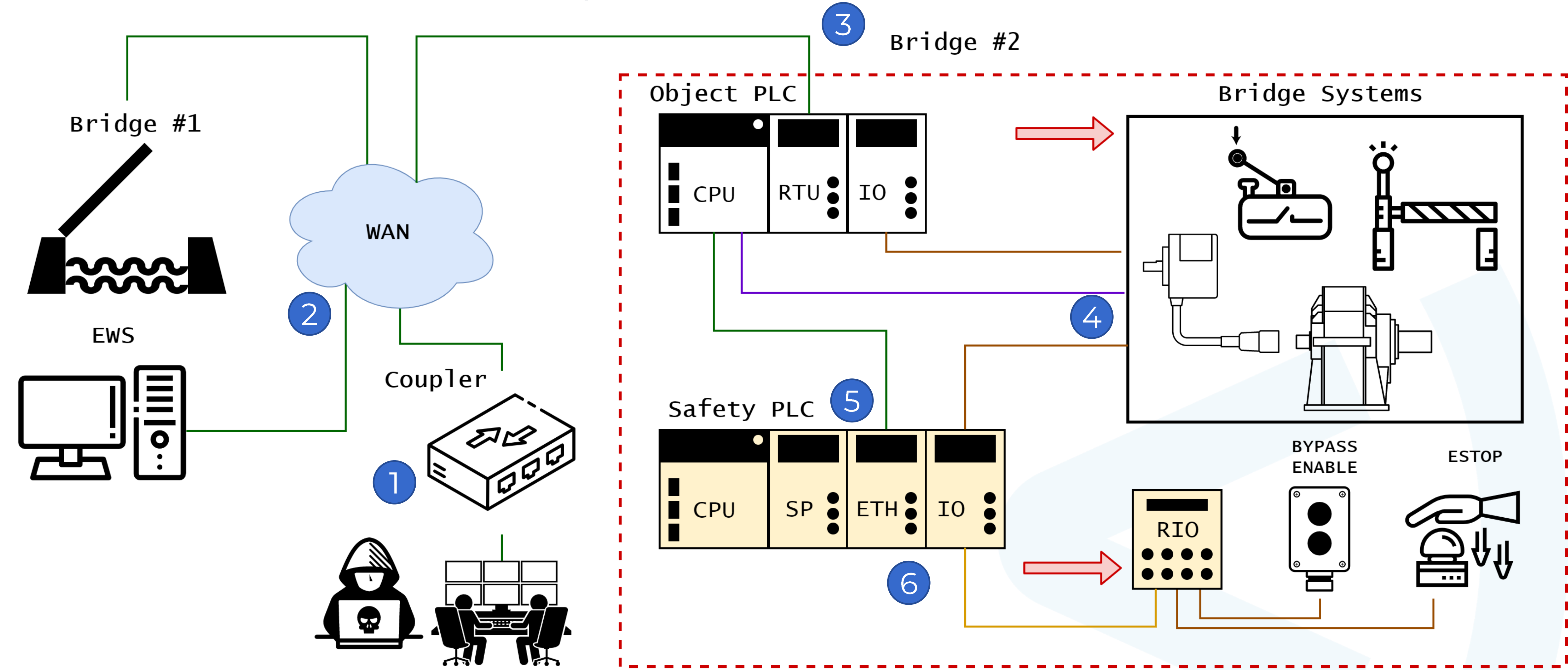
– *Bypass* leaf/lock *limit switches*

▶ Scenario 2 : Close at full speed, trigger E-STOP

– Wait until max velocity
– E-STOP not graceful, CWT inertia

– Bypass creep speed

Attack Path – Likely can't do this from SCADA

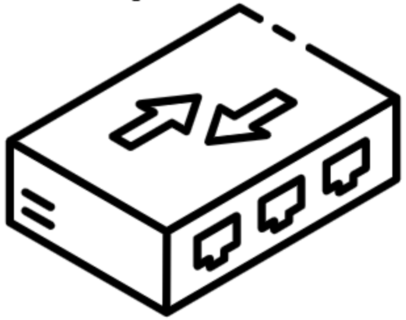


(1) RCE on Coupler (2) Auth Bypass (3) RCE on Object PLC

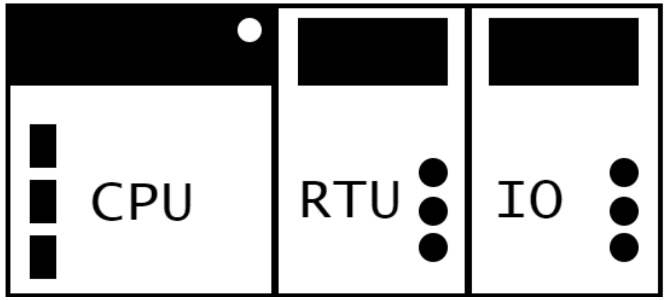
(4) Move into fieldbus (5) Cross SIS PTP link (6) Enable SIS bypass across backplane

Demo Setup

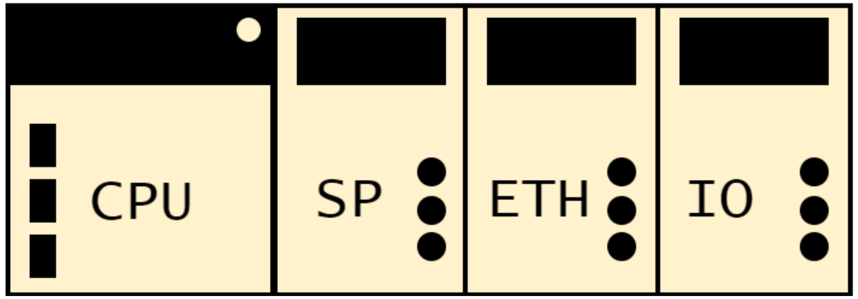
Coupler



Object PLC



Safety PLC



Wago 750-852

Schneider Electric M340

(BMXP3420302, BMXNOR0200h)

Allen-Bradley Guardlogix

(1756-Lx1S, 1756-EN2T/D)

Coupler → Object PLC RTU module

Cannot talk directly to M340 via Wago coupler

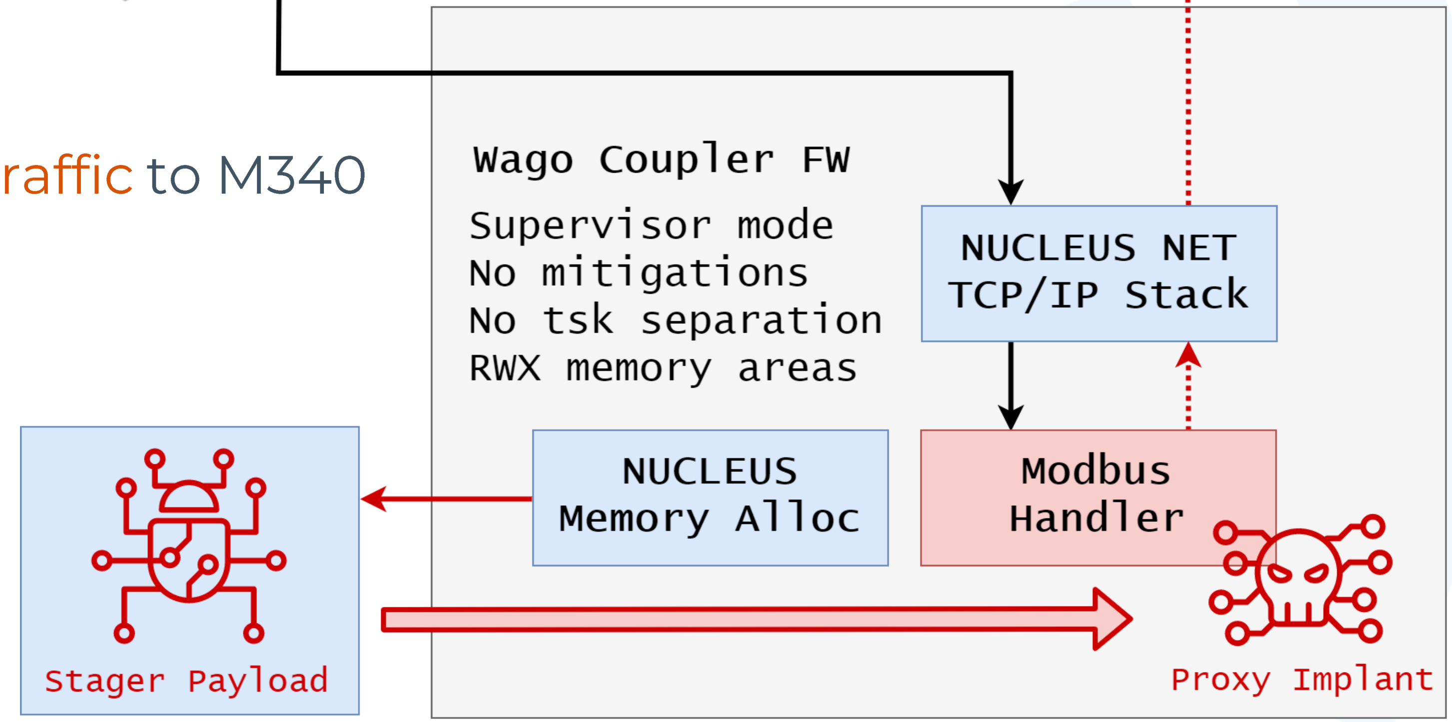
Various protocols

Limited Modbus Mapping

No TDA (routing)

Get RCE on coupler via N-day → Proxy traffic to M340

Hook Modbus handler,
turn into proxy

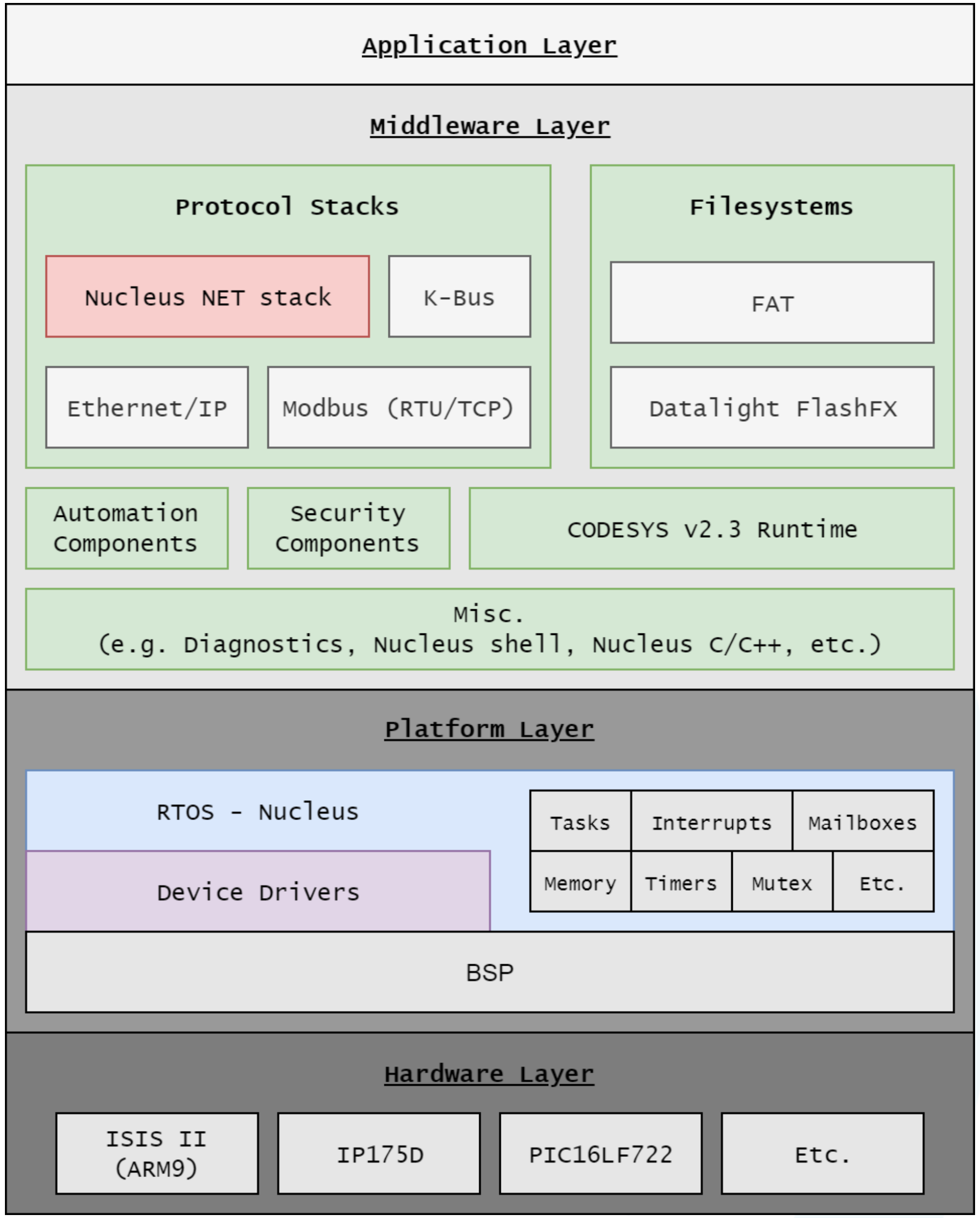


* As of v01.09.25 (16)

Wago 750-852 Firmware*

- Wago 750-x Firmware ZIP
  - .bif: descriptive text file
  - .hex: Intel hex fw
- 60456550.hex → loaded at base address
  - Nucleus RTOS on ARM
  - No symbols
  - Use BinDiff / Diaphora / debug strs
- Nucleus NET TCP/IP stack (**NUCLEUS:13!**)

```
FTP-CONTROL(%s): Closing control connection. Socket %d\r\n
FTP-CONTROL(%s): Cannot delete Event Group. Status %d.\r\n
FTP-CONTROL(%s): NU_Send_To_Queue error. Status %d.\r\n
FTP-CONTROL(%s): NU_Deallocate_Memory error: buffer.\r\n
FTP-CONTROL(%s): NU_Deallocate_Memory error: replyBuff.\r\n
```



CVE-2021-31886* on Wago 750-852

- Stack bof in Nucleus FTPd “USER” cmd
  - Check via strlen() but copy until '\r’ → use fake 0x00

```
while ( server->replyBuff[index + 5] != 13 && index <= 250 )
{
  server->user[index] = server->replyBuff[index + 5];
  ++index;
}
server->user[index] = 0;
// ...
```

```
struct FTP_SERVER {
    // ...
    CHAR *replyBuff;
    CHAR *fileSpec;
    CHAR *path;
    CHAR *renamePath;
    CHAR *currentWorkingDir;
    CHAR *filename;
    CHAR *renameFile;
    struct FLAGS cmdFTP;
    CHAR user[32];
    NU_EVENT_GROUP FTP_Events;
    STATUS transferStatus;
    INT32 restart;
}
```

- Overwrite *FTP_Events* linked list after *user*

- Upon FTP disconnect → triggers LL unlink → gives us write-4 primitive

- Figure out way to write shellcode to RWX .bss area

* NUCLEUS:13, Dissecting the Nucleus TCP/IP stack – Forescout & Medigate Labs

CVE-2021-31886* on Wago 750-852

- Use write-4 to set span_process_packet func ptr to shellcode area

```
.bss:000B14F8                 EXPORT span_
.bss:000B14F8 ; UINT32 (*span_process_packe
.bss:000B14F8 span_process_packet % 4
.bss:000B14F8
```

```
if ( span_process_packet && (protocol_type == 38 || protocol_type == 7) )
{
  MEM_Buffer_List.head->data_ptr -= device->dev_hdrlen;
  MEM_Buffer_List.head->data_len += device->dev_hdrlen;
  MEM_Buffer_List.head->me_data.me_pkthdr.me_buf_hdr.total_data_len += device->dev_hdrlen;
  span_process_packet(MEM_Buffer_List.head->data_ptr, device->dev_index, protocol_type);
}
```

- New FTP session → overwrite *buffer* ptr after *FTP_SERVER*
  - Set to shellcode area
  - Subsequent FTP data will be written to shellcode area

```
void __cdecl Control_Task(UNSIGN
{
  FSP_CB *control_blocka; // [sp
  CHAR nu_drive[3]; // [sp+14h]
  MNT_LIST_S *mount_list; // [sp
  NU_TASK *pointerToThisTask; //
  FTP_SERVER server; // [sp+20h]
  CHAR commandBuf[8]; // [sp+158
  CHAR *buffer; // [sp+160h] [bp
```

- Send LLC frame to trigger shellcode via span_process_packet

- Supervisor mode, no task separation → No need for privesc

* NUCLEUS:13, Dissecting the Nucleus TCP/IP stack – Forescout & Medigate Labs

CVE-2021-31886* on Wago 750-852

- Want bigger payloads?
  - Staged approach!
- Stage 0: loader
  - alloc mem → recv stage 1 in chunks → set ppe_process_packet to stage 1
- Trigger stage 1 via PPOE frame
- Stage 1: implant installer
  - Create Nucleus RTOS task
  - Hook Modbus handler to Proxy FC 0x5A to M340

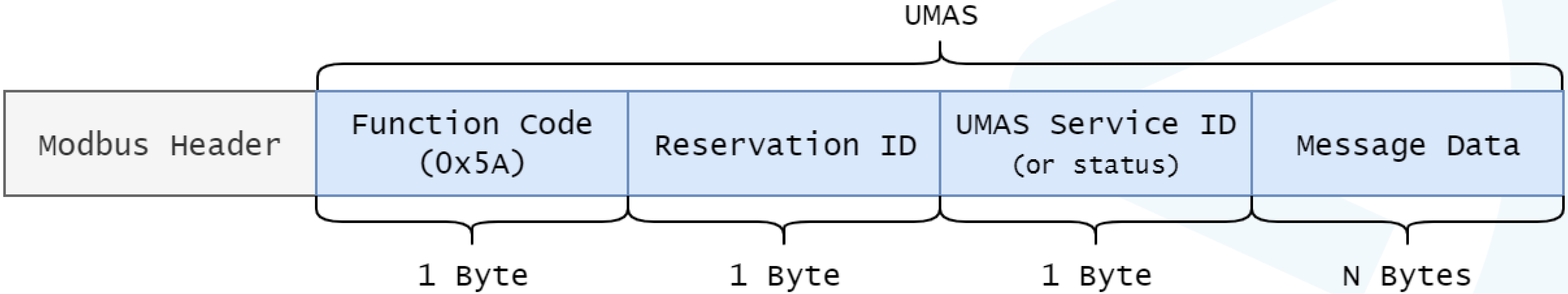
```c
UINT32 *p_ppe = (UINT32*)ppe_process_packet;
UINT32 index = 0;
UINT32 checksum = 0 ;
UINT32 checksm_calc = 0;

//#ifdef WAGO
// the first Dword holding the index
index = *((UINT32*)ptr_packet+IDX_OFF);
if (index == 0)
{
    UINT8 stat = NU_Allocate_Aligned_Memory((void*)pool_ICODEMEM, &stage1_addr, STAGE1_SIZE, 0, 0);

    // make sure the allocation works
}
//if is the end of stream, validate checksum
if(index ==-1)
{

    //NU_Release_Semaphore(TCP_Resource);
    checksum =*((UINT32*)ptr_packet+CHECKSUM_OFF);
    for(UINT32 i =0;i<STAGE1_SIZE;i++)
    {
        checksm_calc +=   ((UINT8*)stage1_addr)[i];

    }

    int good_checksum = checksum == checksm_calc;
    if (good_checksum)
    {

        //patch the p_ppe function pointer to point to the allcocated area
        *p_ppe = (UINT32)stage1_addr;
        #ifdef QEMU
        my_printf("h\n");
        #endif
        //*((UINT8 *)ip_addr + 3) = good_checksum;
        //ICMP_Send_Echo_Request(ip_addr,100);
    }
}

else
{
//copy the stage1 content from ptr_packet to the allcocated area as a fregmented data.
    UINT32 offset =(UINT32)stage1_addr+ (index*FRAG_SIZE);
    //my_printf("%x\n\r",*((UINT32*)ptr_packet+STAGE1_OFF_IN_PACKET));
    my_memcpy(offset, (UINT8*)ptr_packet+STAGE1_OFF_IN_PACKET, FRAG_SIZE);
}
```

* NUCLEUS:13, Dissecting the Nucleus TCP/IP stack – Forescout & Medigate Labs

Object PLC: Schneider Electric UMAS

- Proprietary SE Modicon engineering protocol under Modbus FC 0x5A
  - Much prior work, well-reversed (up to a point)[1,2,3,4]
  - Start/Stop PLC, download/upload logic, read/write memory blocks, etc.

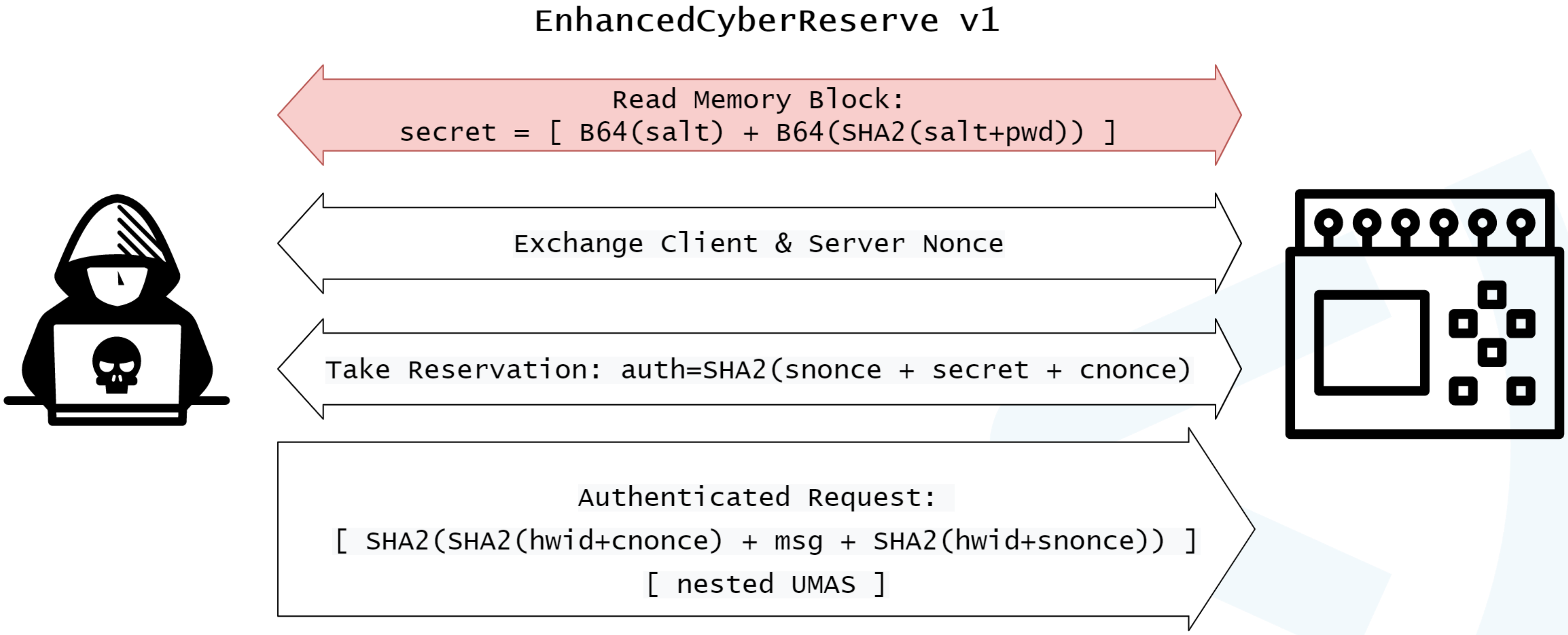
- SE ControlExpert Security Features
  - Project File Encryption (AES-CBC-256)
  - Program/Safety password (weak crypto, client-side)[4]
  - UMAS historically unauth, introduced Application Password[2,3,4]



[1] Project Basecamp – Digital Bond
[2] The secrets of Schneider Electric's UMAS protocol – P. Nesterov et al.
[3] Going Deeper into Schneider Modicon PAC Security – G. Jian
[4] Examining Crypto and Bypassing Authentication in Schneider Electric PLCs (M340 / M580) – N. Miles

CVE-2021-22779: Auth Bypass

- Read secret from mem → Don’t need to know pwd…

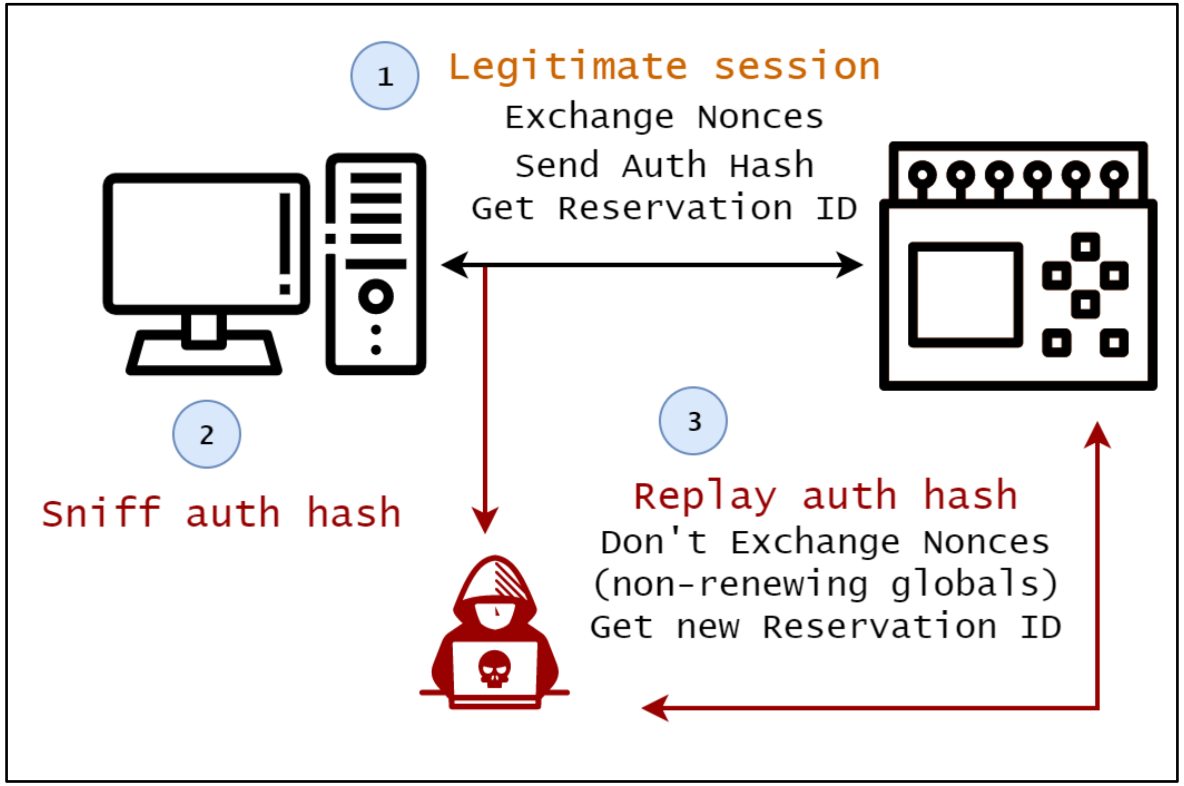


[1] Project Basecamp – Digital Bond, [2] The secrets of Schneider Electric’s UMAS protocol – P. Nesterov et al., [3] Going Deeper into Schneider Modicon PAC Security – G. Jian
[4] Examining Crypto and Bypassing Authentication in Schneider Electric PLCs (M340 / M580) – N. Miles, [5] ModiPwn – G. Kauffman et al.

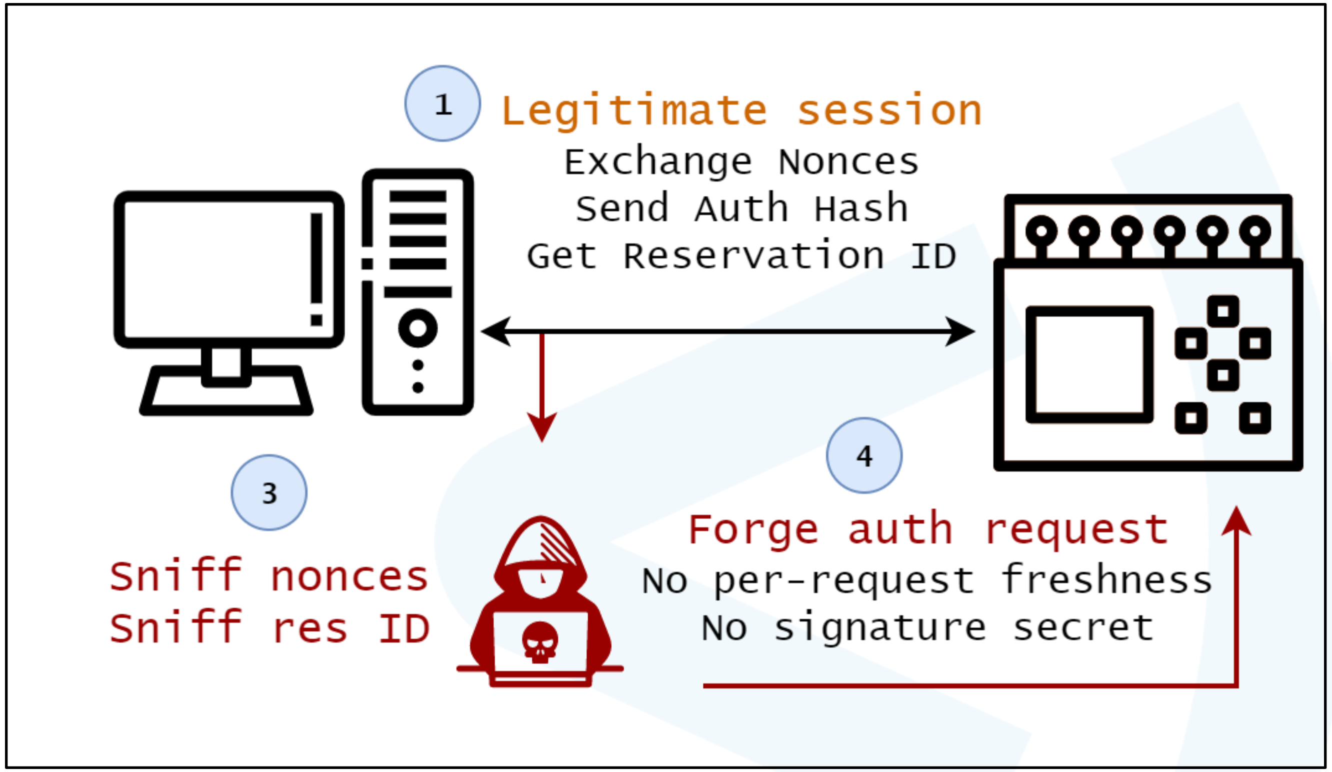
CVE-2022-45789 – Authentication Bypass*

- Patch → PW no longer in mem block, <u>**however**</u>

Reservation Replay



Authenticated Request Forgery



* Affects latest M340 and M580 CPU module FW, see SEVD-2023-010-06

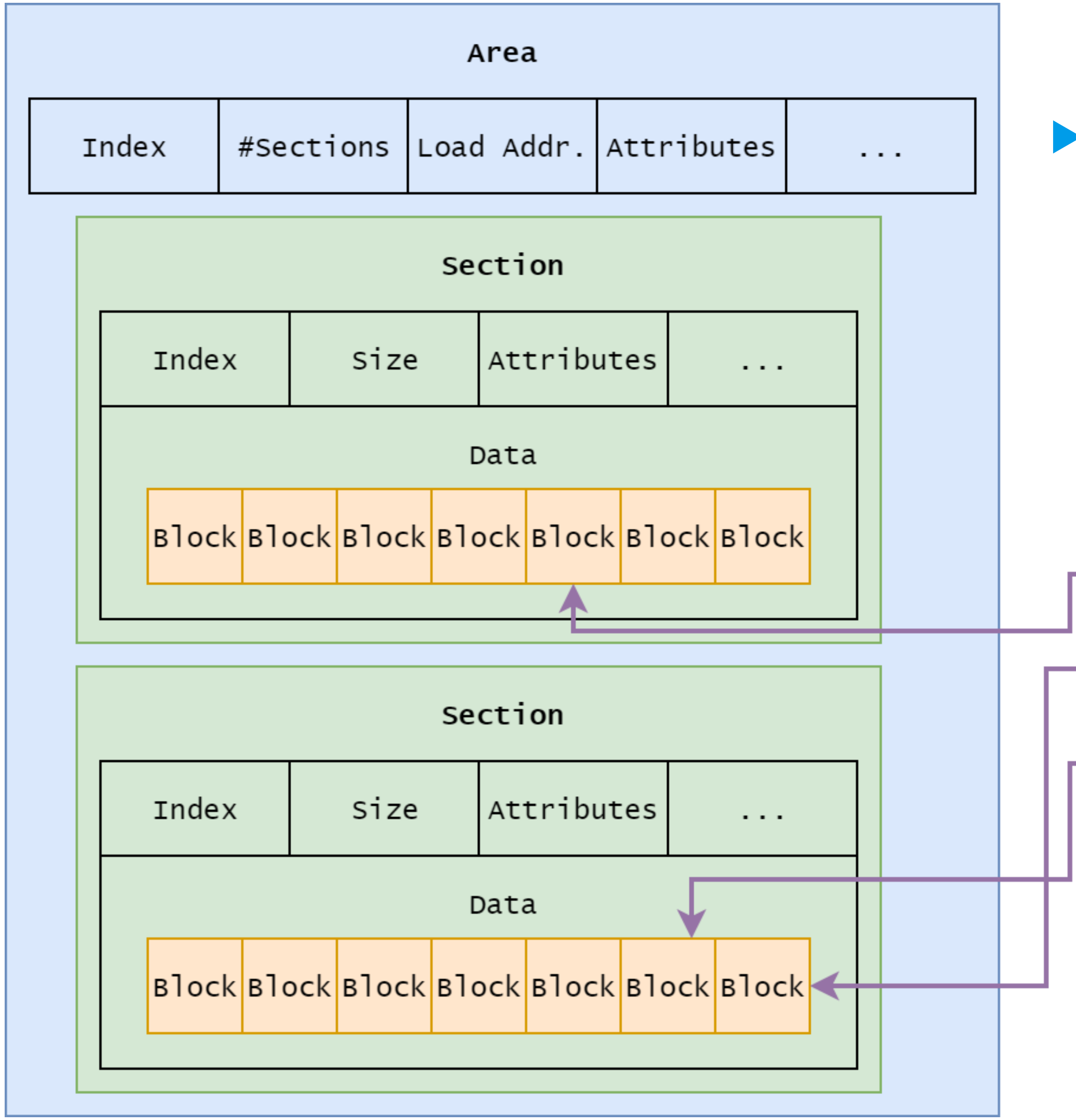
Route to CPU Module RCE

- Different approaches in prior work
  - UMAS: Download logic (0x31) [1,2], vulnerable messages[3,4]
  - TCP/IP stack RCE (M580 but not M340)[5]

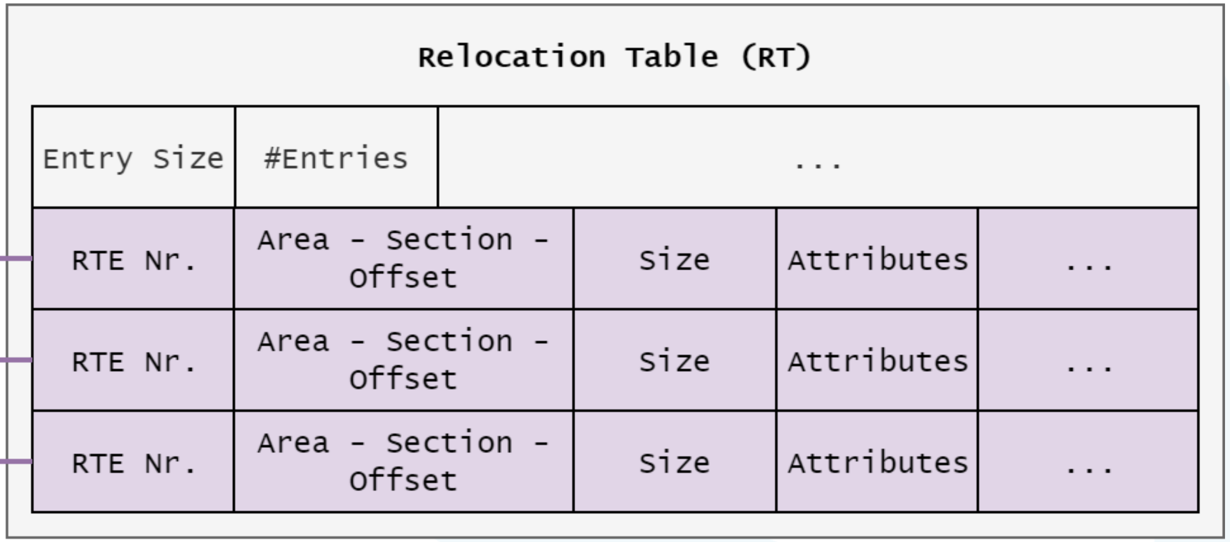
- Want method allows *hotpatching* on *updated* PLC
  - No logic restarts
  - DFIR hostile (project checksums, invisible in source)
  - Using *obscure* protocol features to evade most IDS

[1] TALOS-2018-0742 – J. Rittle
[2] Applying a Stuxnet Type Attack to a Modicon PLC – F. Dola
[3] Going Deeper into Schneider Modicon PAC Security – G. Jian
[4] ModiPwn – G. Kauffman et al.
[5] Exploring and Exploiting PLCs with Urgent/11 Vulnerabilities – B. Hadad et al.

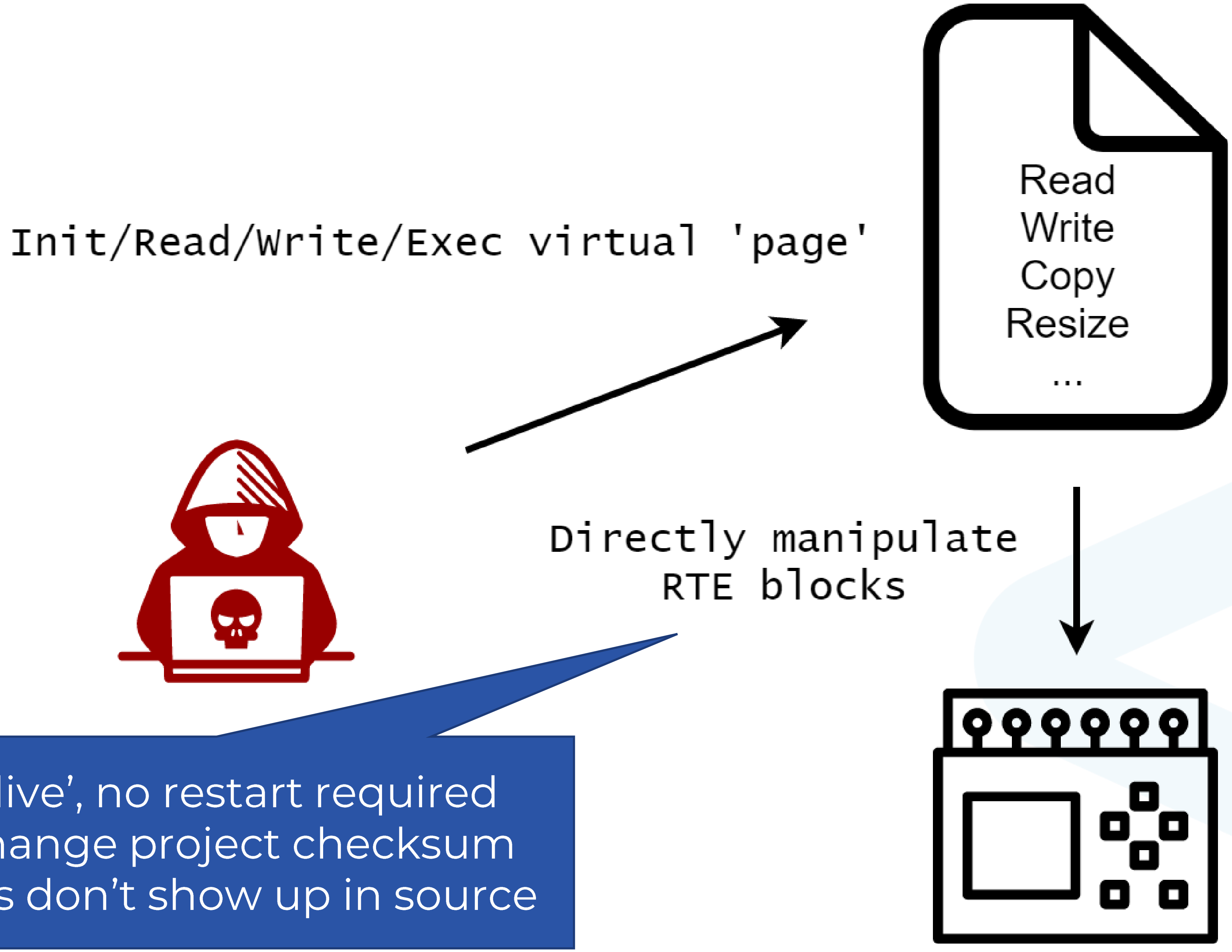
Background: Modicon Application Binary File (APX)



- Block Types
  - Data / Exec / Upload Info / FB Data / Constant / etc.



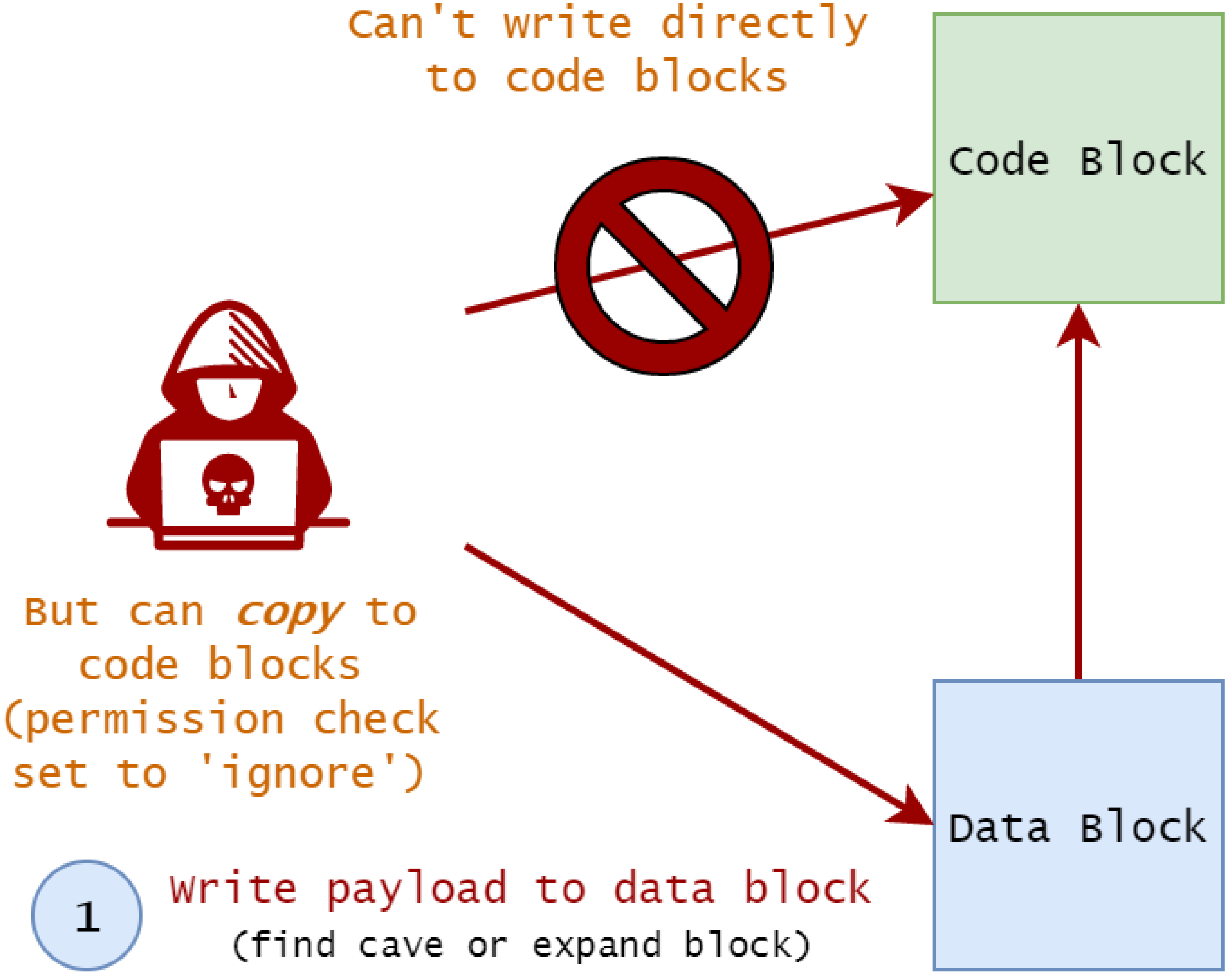
Unexplored UMAS CSA Requests (0x50)



Subsystem with proprietary command set

- Happens ‘live’, no restart required
- Doesn’t change project checksum
- Exec mods don’t show up in source

CVE-2022-45788 – Modicon CPU RCE*



3 Get RCE when block executes as part of logic

2 Copy from data block to code block

(find cave or expand block, then hijack control flow)

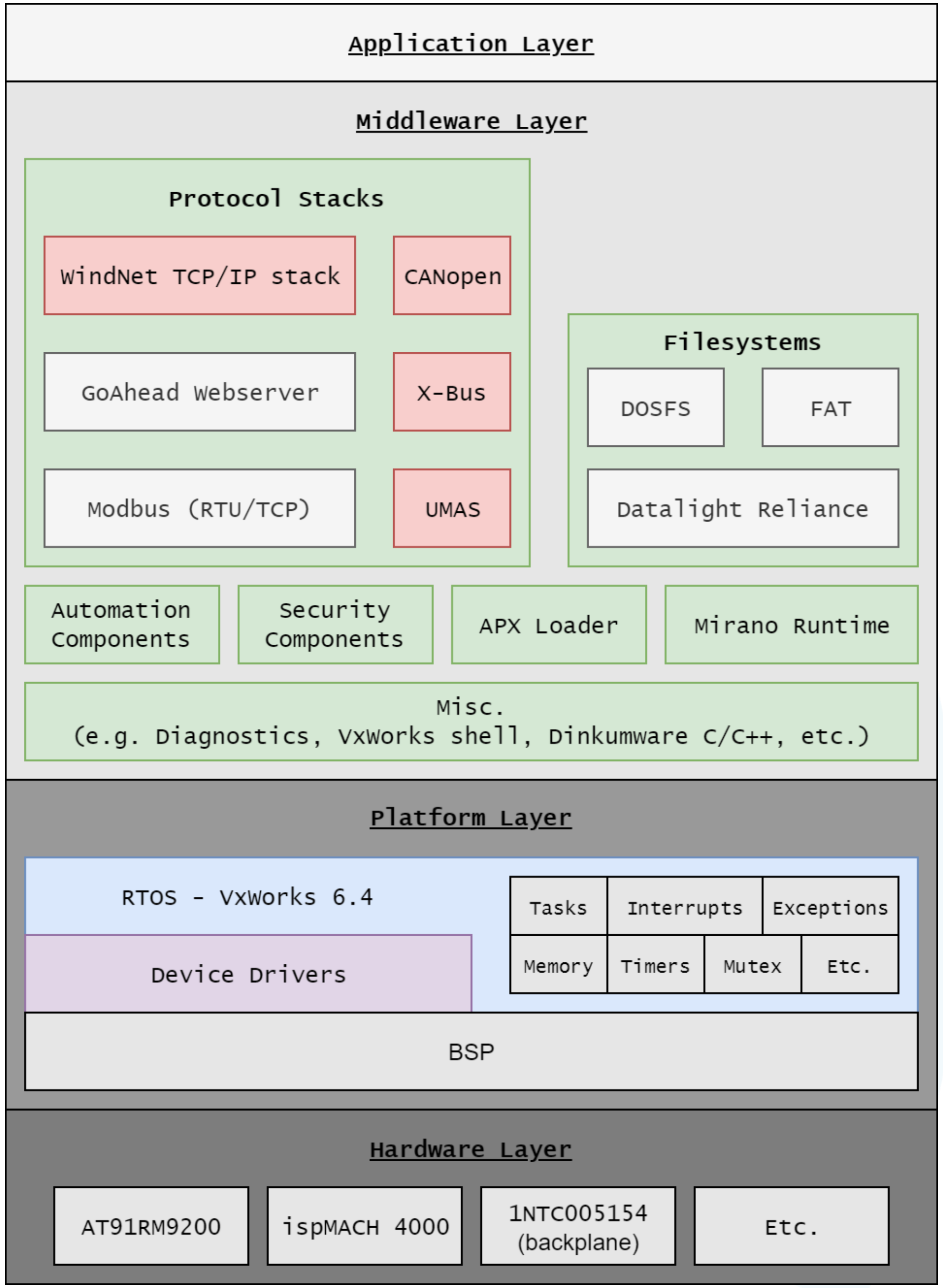
```
if ( !ignore )
{
  if ( rte_ptr )
  {
    if ( (rte_ptr->attr & 0x10000) != 0 )
    {
      return 0x9191;
    }
    else
    {
      blocktype = rte_ptr->attr & 0xF;
```

* Affects latest M340, M580, M1E, MC80, Quantum, Premium CPU module FW, see SEVD-2023-010-05

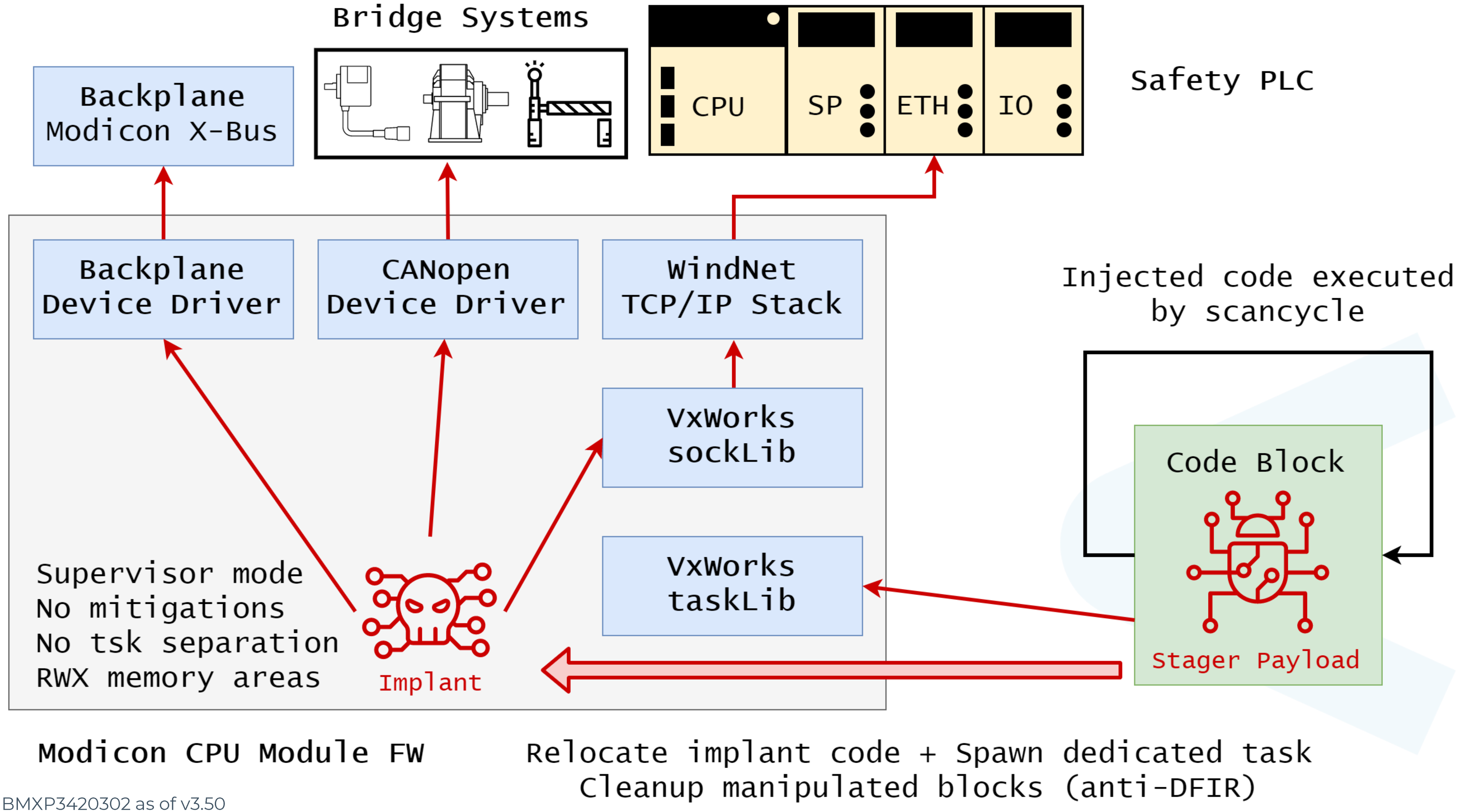
SE BMXP3420302 Firmware*

- SE Firmware LDX = ZIP
- vxWorks_bmx*.bin → UNITYM binary
  - Segment base @ 0x20000000
  - FW code start @ 0x20010110
  - Runtime base @ 0x28000000
  - VxWorks 6.4 on ARMv4 (so no XN)
  - Manually reconstruct symbol table
- Runtime exec blocks via sas_UserCodeExec
  - Scancycle timer is in the way
  - Hijack triggerable func to escape

```
v4 = kl_userTimeEn(result);
v5 = sas_UserCodeExec(v4);
kl_userTimeDis((int)v5);
```



Stager Payload & Implant



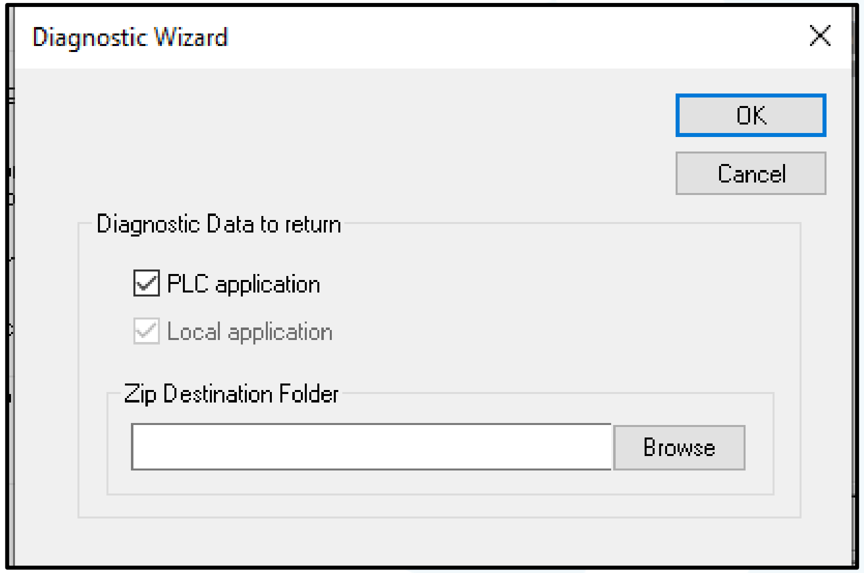
* BMXP3420302 as of v3.50

(Counter) Forensics

- Like all PLCs, no introspection on M340
  - Also won’t notice anything in ControlExpert EWS

- But: project upload will fetch *all* blocks from memory (incl. exec)
  - Carve APX* to extract exec blocks, contain raw ARM code
  - Compare to known-good, RE for malicious patterns (GetPC, egghunters, etc.)

- Clever attacker will clean up after relocating implant
  - Inject code into exec block
  - Hijack triggerable func ptr. to escape runtime
  - Spawn implant task
  - Restore old exec block contents



- Attacker mistakes might still be logged
  - Watchdog triggers, crashdumps (help → about → technical support)

* http://lirasenlared.blogspot.com/2018/10/the-apx-logic-file-format.html, https://talosintelligence.com/vulnerability_reports/TALOS-2020-1144

CANopen payload

▶Talk to **M340 CANopen API**, use CiA funcs

```
can_SWrite_SDO(ND, 0x1F51, 1, START_BOOT,
can_SWrite_SDO(ND, 0x1F51, 1, ERASE_FLASH,
...
can_SWrite_SDO(ND, 0x1F50, 1, block[i],
```

Index	SDO Name
0x1023	OS CMD[2]
0x1024	OS CMD Mode[2]
0x1025	OS Debugger[2]
0x1026	OS Prompt[2]
0x1F50	Download Program[3]
0x1F51	Program Control[3]

▶RCE via SDO: override firmware (safety) limits

- In-band code dndl – trigger bootloader via NMT/SDO
- Memory read/write – hotpatching RCE
- If auth at all: (static) 32-bit value written to some SDO

[1] CAN-in-Automation (CiA) 302-2, [2] CAN-in-Automation (CiA) 301, [3] CAN-in-Automation (CiA) 302-3

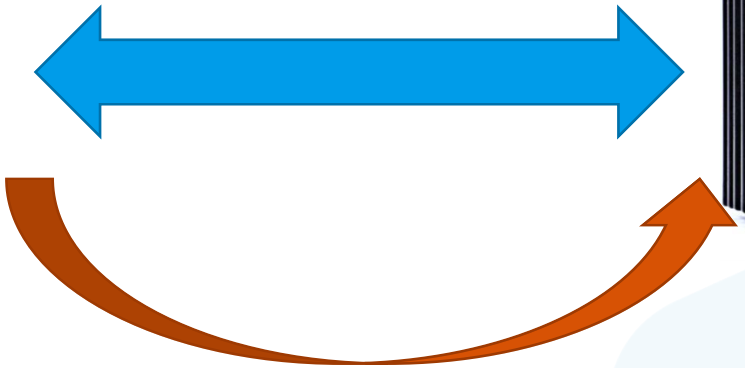
Object PLC → Safety PLC Ethernet module

Cannot talk directly to GuardLogix CPU module or route CIP

Non-routable PTP link

Only Modbus TCP (AOI)

Explicit protected mode



Exploit N-day vuln in TCP/IP stack for RCE on Ethernet Module → hop to rest of SIS

Allen-Bradley GuardLogix Safety PLC 1756-EN2T/D Ethernet Module

* As of v11.01

AB 1756-EN2T/D Firmware*

- Allen-Bradley Firmware ZIP
  - .nvs: descriptive text file
  - .plt: binary fw
  - .der: certificates

Name	Start	End	R	W	X
.text	00010000	005C2FE8	R	.	X
.vfp11_veneer	005C2FE8	005C32E8	R	.	X
.wrs_build_vars	005C32E8	005C3450	R	.	.
LOAD	005C3450	005C3460	R	W	X
.data	005C3460	006631F0	R	W	.
LOAD	006631F0	00664000	R	W	X
.vectors	00664000	00665000	R	W	.
.bss	00665000	0089FE80	R	W	.

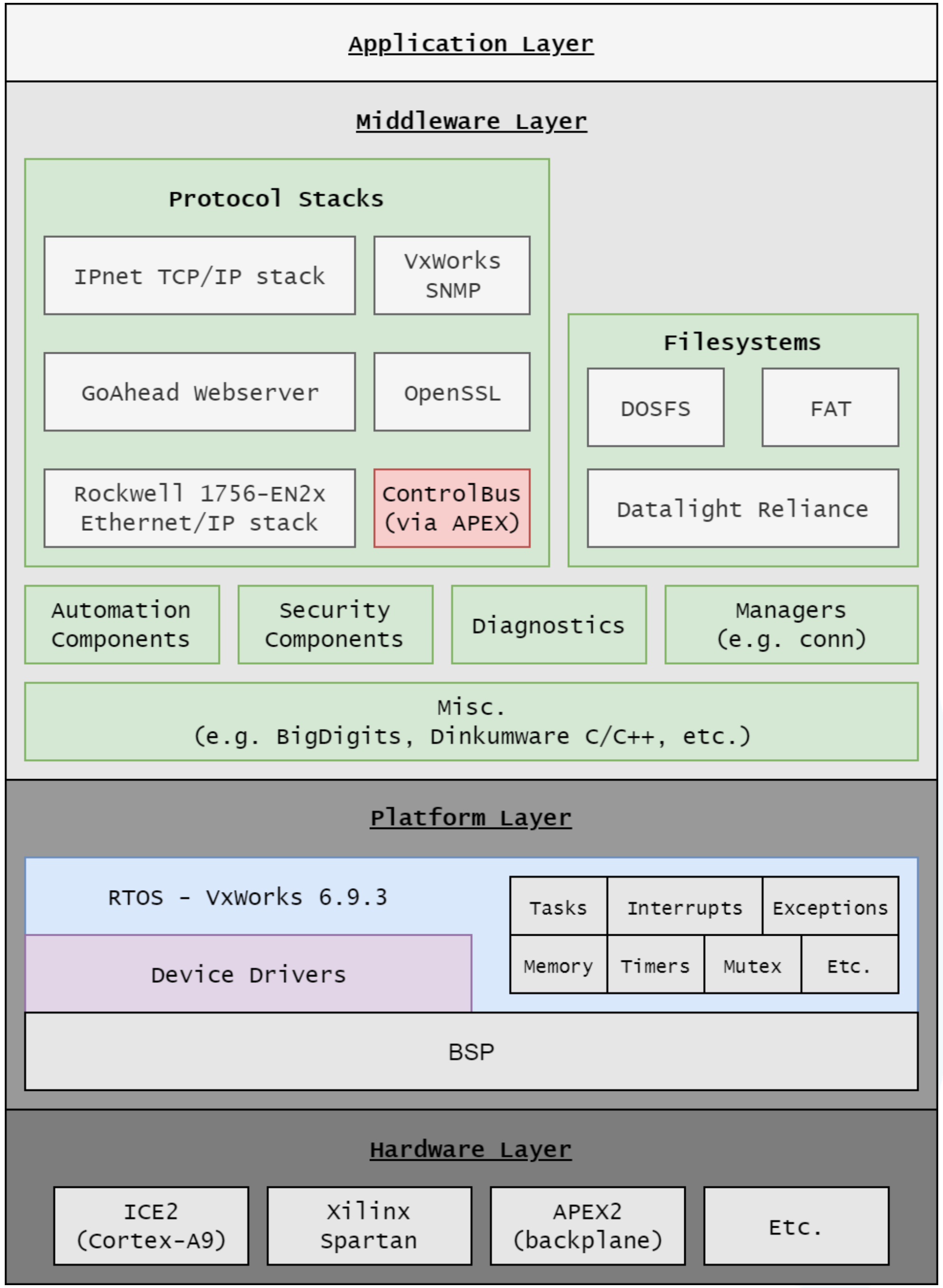
- PN-497069.plt → ELF binary
  - Segments pre-loaded
  - VxWorks 6.9.3.3 on ARM
  - Manually reconstruct symbol table

```
symbol <0, aAccessDescript_0, ACCESS_DESCRIPTIO
                                  ; DATA XREF: usrStandalo
                                  ; usrStandaloneInit+7C↑
symbol <0, aAccessDescript_1, ACCESS_DESCRIPTIO
symbol <0, aAccessDescript_2, ACCESS_DESCRIPTIO
symbol <0, aAcmAllocateele, ACM_AllocateElement
symbol <0, aAcmAllocatetar, ACM_AllocateTarget,
```

AB 1756-EN2T/D Firmware*

- Allen-Bradley ICE2
  - Main SoC (Quad-core Cortex-A9)
  - ICE2 = ENIP, ICE3 = PROFINET
- Allen-Bradley APEX2 (NEC) backplane IC
  - ControlBus is CIP-based
- InterPeak Ipnet stack (**URGENT/11!**)
- Interesting device drivers for payload
  - Display LEDs, Backplane comms

- _ZN12bsp_ApexImpl12DownloadCodeEv
- _ZN12bsp_ApexImpl13StartFirmwareEv
- _ZN12bsp_ApexImpl13InitBackplaneEb
- _ZN12bsp_ApexImpl9IsFaultedEv
- _ZN12bsp_ApexImpl13IsCbaAssertedEv
- _ZN12bsp_ApexImpl13IsCbbAssertedEv



CVE-2019-12256* on Allen-Bradley 1756-EN2T/D

- Send malformed IP options (URGENT/11) via VxWorks raw sockets
  - Multiple Source Record Route (SRR) opts generate ICMP error response
  - Stack buffer overflow (opts copied to response without validation)
  - First exploited against 1756-EN2TR/C by Armis*

```
srr_opt->ptr = 4;
while ( offset_to_current_route_entry > 0 )
{
  memcpy((char *)srr_opt + (unsigned __int8)srr_opt->len, current_route_entry, 4);
  current_route_entry -= 4;
  offset_to_current_route_entry -= 4;
  srr_opt->len += 4;
}
memcpy((char *)srr_opt + (unsigned __int8)srr_opt->len, icmp_param + 12, 4);
v18 = srr_opt->len + 4;
```

- XN enabled (no other mitigs) → need ROP chain
  - Carefully pick & align SRRs
  - Hijack PC & control stack layout
  - Write-4 ROP + stack fixup → cont. exec

```
def en2t_d_www( interface, host, fw_ver, what, where ):
    # R10 = ........ -> new R10  (<data>)
    # R11 = ........ -> new R11  (<offset - 0x1AC>)
    # PC  = ........ -> gadget 0 (popret)
    # ........       -> new R3   (unused)
    # ........       -> gadget 1 (www_restore)
    # ........       -> unused

    # POP {R3, PC}
    gadget_0 = version_lut[ 'en2t_d' ][ fw_ver ]['popret']
    # STR R10, [R11, #0x1AC]; ...; ADD SP, SP, #0x18; POP {R6-R11, PC}
    gadget_1 = version_lut[ 'en2t_d' ][ fw_ver ]['www_restore']
```

* Exploring & Exploiting PLCs with URGENT/11 – B. Hadad et al.

CVE-2019-12256* on Allen-Bradley 1756-EN2T/D

- Use write-4 to deliver payload
  - Large RWX 'LOAD' segment (NULLs)
  - Chop shellcode into chunks of 4 → write to RWX seg via ROP chain
  - Ret 2 payload

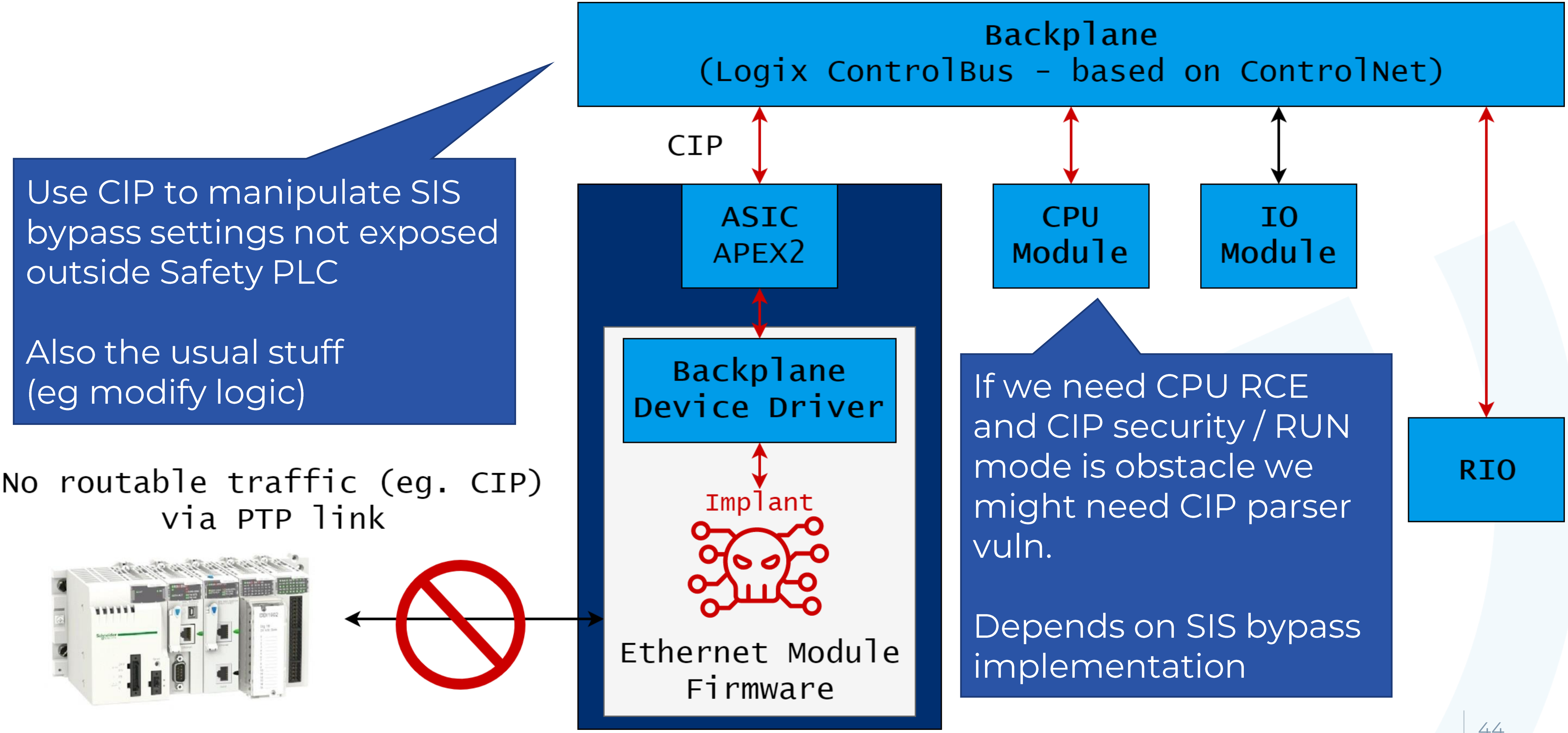
```
LOAD:006631F0 ; Segment type: Pure code
LOAD:006631F0                 AREA LOAD, CODE, READWRITE,
LOAD:006631F0                 ; ORG 0x6631F0
LOAD:006631F0                 CODE32
LOAD:006631F0                 DCD 0, 0, 0, 0, 0, 0, 0, 0,
LOAD:006631F0                 DCD 0, 0, 0, 0, 0, 0, 0, 0,
```

- Only slight diffs with Armis exploit against 1756-EN2TR/C
  - ROP chain construction, RWX/gadget/func addrs

- Supervisor mode, no task separation → No need for privesc
  - Spawn VxWorks task for stable implant
  - Talk directly to device drivers

* Exploring & Exploiting PLCs with URGENT/11 – B. Hadad et al.

Move across Safety PLC backplane



Demo Video

Disclosure

* SEVD-2023-010-05, SEVD-2023-010-06

- Coordinated disclosure with Schneider Electric
  - Issues reported in April and July 2022
  - Advisories* released in January 2023, updated in March 2023

- CVE-2022-45788 (RCE)
  - Remediations available for M580 (excluding safety), M1E
  - Mitigations for others

- CVE-2022-45789 (auth bypass)
  - Currently mitigations only

- We suggested retrofit fix: Secure Remote Password(SPR) + HMAC
  - Auth user to PLC with SRP (zero-knowledge, MitM-resistant, discrete-log based)
  - Derive HMAC key from shared SRP key K
  - Sign messages with HMAC

(some) Mitigation, Detection, and DFIR advice

Attack Step	Controls
Wago 750 implant	• Alert on UMAS to non-Modicon devices • Monitor Modbus TCP statistics
UMAS Auth Bypass (CVE-2022-45789)	• Restrict UMAS flow to EWS (IP ACLs, FW) • Look for auth request (SVC 0x38) without none exchange (SVC 0x6E)
UMAS RCE (CVE-2022-45788)	• Alert on UMAS CSA (SVC 0x50) • Monitor watchdog errors • Upload PLC project, extract & carve APX, look for ARM shellcode
1756-EN2T* RCE (CVE-2019-12256)	• Monitor IP & assert statistics
1756-EN2T* implant	• Monitor task statistics

Task Statistics

Name	Entry Point	ID	Priority
tJobTask	1e7208	efc4e8	0
tExcTask	1e69fc	7f85b8	0
tErfTask	10b9c	f00f70	10
tLogTask	1e76bc	f04110	0
tNet0	1bdc8	f11e00	50

IP Statistics

Forwarding	1
Default TTL	64
In receives	812
In header errors	4
In address errors	0
Forwarded datagrams	0

- For full overview, see report*

* https://www.forescout.com/resources/l1-lateral-movement-report

Conclusions

- There's likely a lot of network 'crawl space' that's not on your radar
- If a L1 device sits between segments, it needs a perimeter security profile
- Stop treating certain links (serial, PTP, couplers, non-routable) as if they're immune
- Impact of compromise not limited to explicit link capabilities or 1st order connectivity
- With *deep access*, things become possible which change potential impact

Thank you. | FORESCOUT

Full report

https://www.forescout.com/resources/l1-lateral-movement-report