



AUGUST 3-8, 2019
MANDALAY BAY / LAS VEGAS

Rogue7

Rogue Engineering Station Attacks on Simatic S7 PLCs



Eli Biham, **Sara Bitan**, Aviad Carmel, Alon Dankner, **Uriel Malin**, Avishai Wool
Technion – Israel Institute of Technology Tel-Aviv University

Who Are We?

Dr. Sara Bitan

- Senior researcher at the Technion Hiroshi Fujiwara Cyber Security Research Center



Hiroshi Fujiwara
Cyber Security
Research Center

- Founder and CEO of CyCloak - Secure System Design and Audit



Uriel Malin

- MSC Student at Tel Aviv University, advised by Prof. Avishai Wool
- Security Researcher at Medigate – Healthcare IoT Security



- Uncovered design vulnerabilities in the S7 protocol
- An exploit that performs remote stealth programming of an S7-1500 PLC

The
Operator



The
Engineer



The
Attacker



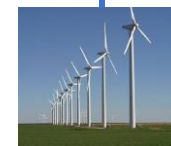
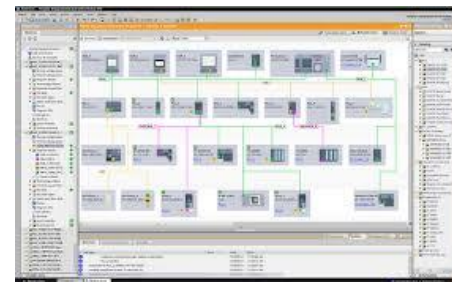
Industrial Control System

- A distributed computerized system
- Operates and monitors physical devices
- Controls critical infrastructure
 - Power plants
 - Water facilities
 - Transportation systems
 - Chemical plants



Programmable Logic Controller

- PLC - The core of the ICS
- Connected to sensors and active devices
- Runs a control program that periodically samples the sensors and triggers the devices accordingly
- A bridge between the virtual and the kinetic worlds
- The target of our attacks



PLC interfaces

HMI - SCADA

WinCC

TIA

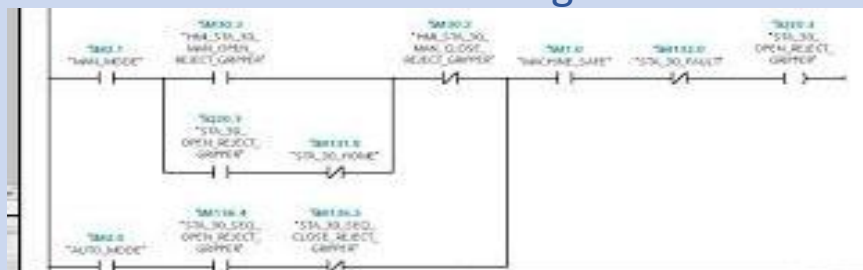
Engineering
Workstation

STEP 7



The S7 Protocol

The Control Program



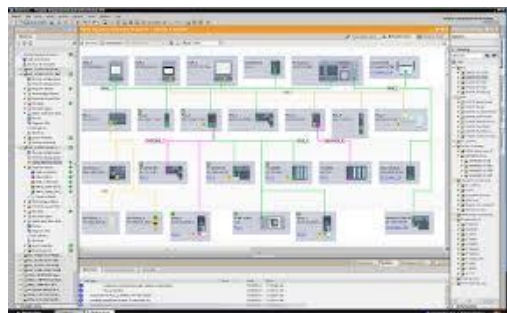
Web
Server

SNMP
Server

PLC

S7-1500

PLC Operating System



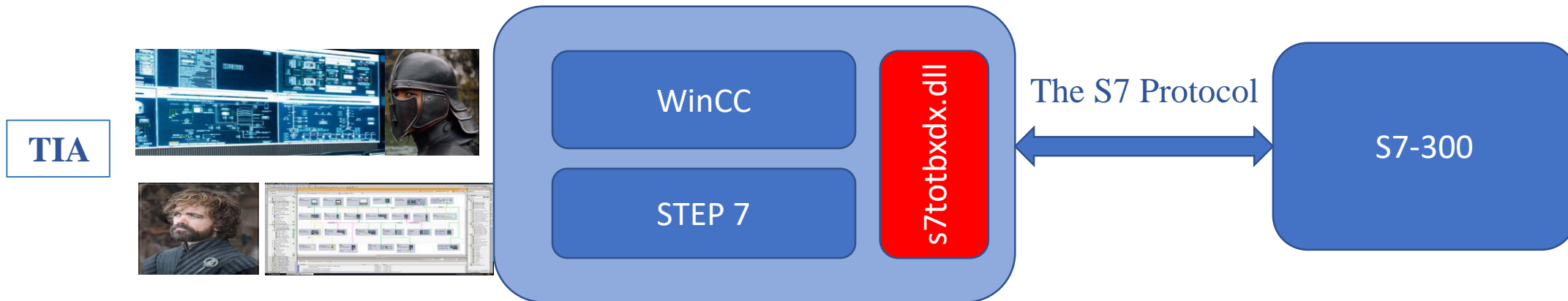
The S7 Protocol





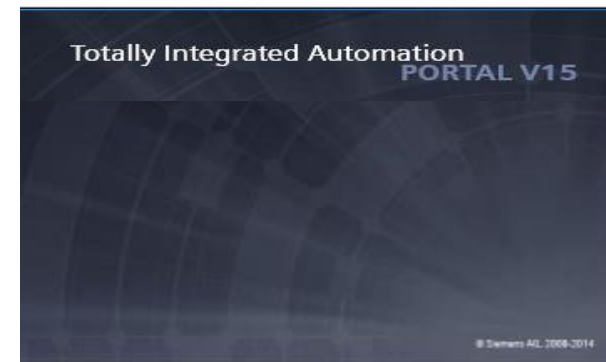
Stuxnet Malware (9/2010)

- The most famous cyber-attack on ICS
- Targeted Siemens S7-300 PLC
- Infected both WinCC and Step 7 packages



TIA as a Soft Belly

- Typically attacks are exploiting vulnerabilities the engineering station:
 - [CVE-2012-3015](#) : untrusted search path vulnerability in Siemens SIMATIC STEP7 v5.5– July-26-2012
 - [CVE-2019-10915](#): authentication bypass in TIA v15.1 –July-11-19 by Tenable Security



-



The S7-1500 PLC

- One of two new members in the SIMATIC PLCs product line
 - S7-1500 is the high-end PLC
 - The other is S7-1200



S7-1500 Security Enhancements

- New version of the S7 protocol
 - Integrity protection of the messages
- Know-how and copy protection
- PLC access control
 - Using passwords
 - Mitigates the program download attacks
 - **Not used by customers in practice**

| Protection level | Access | | | Access permission | |
|--|--------|------|-------|-------------------|--------------|
| Protection | HMI | Read | Write | Password | Confirmation |
| <input checked="" type="radio"/> Full access (no protection) | ✓ | ✓ | ✓ | | |
| <input type="radio"/> Read access | ✓ | ✓ | | | |
| <input type="radio"/> HMI access | ✓ | | | | |
| <input type="radio"/> No access (complete protection) | | | | | |

The S7 Protocol

sara_dl (1).pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

s7comm-plus

| Source | Destination | Protocol | Leng | Info |
|--------------|--------------|-------------|------|--|
| 192.168.0.61 | 192.168.0.59 | S7COMM-PLUS | 290 | +25032 Ver:[V1] Seq=1 [Req CreateObject] ObjectServerSessionContainer ClassServerSession / GetNewRIDOnServer ClassSubscrip |
| 192.168.0.59 | 192.168.0.61 | S7COMM-PLUS | 270 | +25032 Ver:[V1] Seq=1 [Res CreateObject] Retval=OK ObjId=Unknown (999), Unknown (949) |
| 192.168.0.61 | 192.168.0.59 | S7COMM-PLUS | 470 | +25032 Ver:[V2] Seq=2 [Req SetMultiVariables] ObjId=Unknown (999) |
| 192.168.0.59 | 192.168.0.61 | S7COMM-PLUS | 86 | +25032 Ver:[V2] Seq=2 [Res SetMultiVariables] Retval=OK |
| 192.168.0.61 | 192.168.0.59 | S7COMM-PLUS | 155 | +25032 Ver:[V3] Seq=3 [Req SetVariable] ObjId=Unknown (999) |
| 192.168.0.59 | 192.168.0.61 | S7COMM-PLUS | 118 | +25032 Ver:[V3] Seq=3 [Res SetVariable] Retval=OK |

> Transmission Control Protocol, Src Port: 25032, Dst Port: 102, Seq: 36, Ack: 36, Len: 236

> TPKT, Version: 3, Length: 236

> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol

> S7 Communication Plus

> Header: Protocol version=V1

Protocol Id: 0x72

Protocol version: V1 (0x01)

Data length: 221

> Data: Request CreateObject

Opcode: Request (0x31)

Reserved: 0x0000

Function: CreateObject (0x04ca)

Reserved: 0x0000

Sequence number: 1

Session Id: 0x00000120

> Transport flags: 0x36, Bit1-SometimesSet?, Bit2-AlwaysSet?, Bit4-AlwaysSet?, Bit5-AlwaysSet?

> Request Set

> Item Value: ID=ObjectServerSessionContainer (UDInt) = 0

Unknown value 1: 0x00000000

> Object: CIsId=ClassServerSession, RelId=GetNewRIDOnServer

Element Tag-Id: Start of Object (0xa1)

Relation Id: GetNewRIDOnServer

Class Id: ClassServerSession

> Class Flags: 0x00000000

Attribute Id: None

> Attribute

Packets: 18140 · Displayed: 1475 (8.1%)

Profile: Default

Session oriented.
Session begin
with a 4-ways
handshake

ISO transport
over TCP

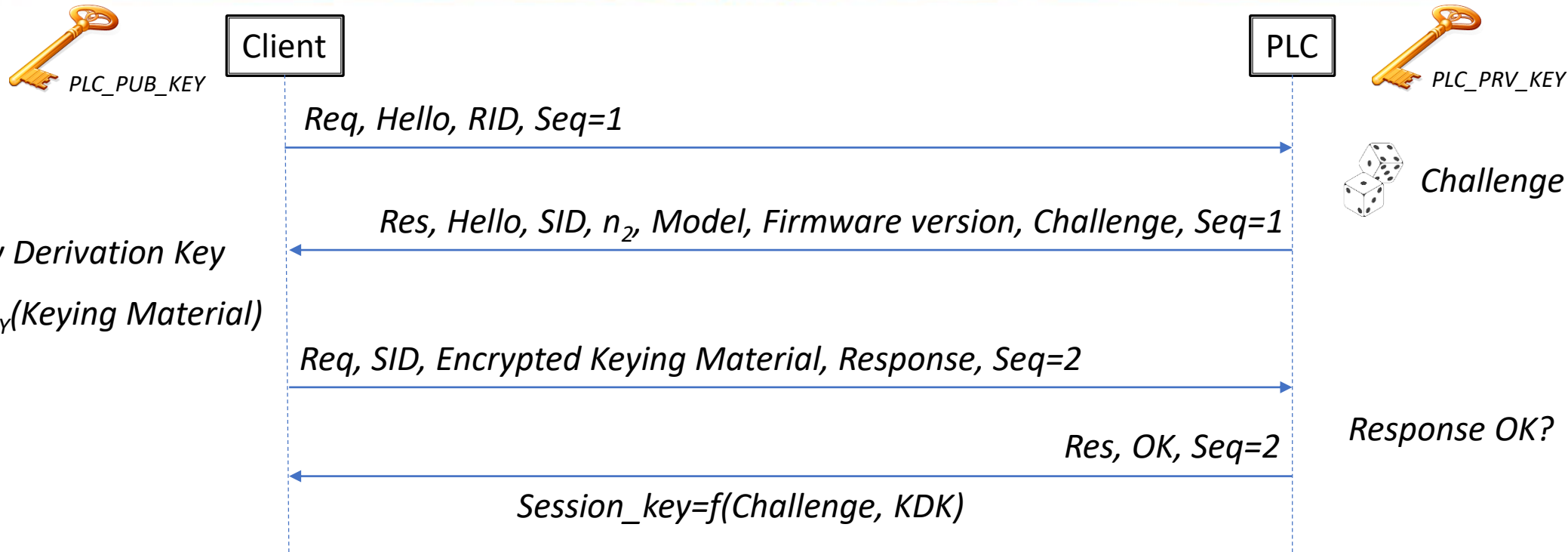
Version P3

Client can create,
modify and delete
objects in the PLC's
internal memory

Session ID

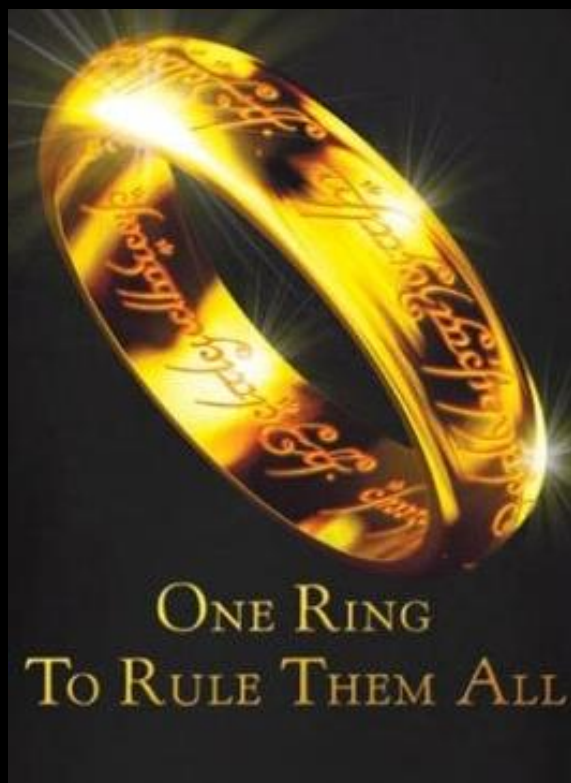
Example: create a
server session object

The P3 Handshake Protocol



| SSL | S7 P3 Handshake |
|---|---|
| Server only authentication | Server (PLC) only authentication |
| Client generates keying material. Encrypts with server public key. | Client generates keying material. Encrypts with server public key. |
| Different servers use different public keys. | All PLCs from the same model and version use a single key. |

P3 Handshake Design Vulnerabilities



With Many Working Forged Copies

Attacking the P3 program download exchange

Control Program Create Message

sara_dl (1).pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

s7comm

Expression...

> Frame 2258: 731 bytes on wire (5848 bits), 731 bytes captured (5848 bits) on interface 0

> Ethernet II, Src: Cma/Micr_a7:03:7a (00:0e:04:a7:03:7a), Dst: Dell_84:97:fa (b8:ca:3a:84:97:fa)

> Internet Protocol Version 4, Src: 192.168.0.61, Dst: 192.168.0.59

> Transmission Control Protocol, Src Port: 25033, Dst Port: 102, Seq: 10894, Ack: 6679, Len: 677

> TPKT, Version: 3, Length: 677

> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol

> S7 Communication Plus

> Header: Protocol version=V3

> Integrity part

> Digest Length: 32

> Packet Digest: 7663521c5b91161755294bb8d1f6eca9b602d3d457ca038c...

> [4 S7COMM-PLUS Fragments (3561 bytes): #2254(976), #2255(976), #2257(976), #2258(633)]

> Data: Request CreateObject

> Opcode: Request (0x31)

> Reserved: 0x0000

> Function: CreateObject (0x04ca)

> Reserved: 0x0000

> Sequence number: 40

> Session Id: 0x000003e7

> Transport flags: 0x36, Bit1-SometimesSet?, Bit2-AlwaysSet?, Bit4-AlwaysSet?, Bit5-AlwaysSet?

> Request Set

> Item Value: ID=NativeObjects.thePLCProgram_Rid (UDInt) = 0

> Unknown value 1: 0x00000000

> Unknown VLO-Value in Data-CreateObject: 10

> Object: CIsId=ProgramCycle0B.Class_Rid, RelId=0B.1

> Element Tag-Id: Start of Object (0xa1)

> Relation Id: 0B.1

> Class Id: ProgramCycle0B.Class_Rid

> Class Flags: 0x00000028, User4, Persistent

> Attribute Id: None

> Attribute

> Element Tag-Id: Attribute (0xa3)

> Item Value: ID=Block.AdditionalMAC (Struct) = 1820 (StructMAC)

> Attribute

> Element Tag-Id: Attribute (0xa3)

> Item Value: ID=FunctionalObject.Code (Blob) = 0xefbeadde7c00000001000000020000000320000000040000...

> Attribute

> Element Tag-Id: Attribute (0xa3)

> Item Value: ID=Block.BodyDescription (Blob) Sparsearray = 0x98000002787defaeae49a3d811c0c6646a515e22a8499798..., 0x00043078a83781ca537d29476e26dcd3ab8fb707348c5af...

0330 00 a3 94 14 00 14 00 81 55 ef be ad de 7c 00 00 U....|..

0340 00 01 00 00 00 02 00 00 00 32 00 00 00 04 00 -2.....

0350 00 00 00 00 00 93 79 cc 34 23 63 e4 99 04 00 00y. 4#c.....

0360 00 00 00 00 00 f5 1c ae 2a 27 0f 13 8d 01 00 00*!.....

0370 00 00 00 00 00 3c 00 00 00 03 3e 76 65 05 e5 5a<..>ve..Z

Frame (731 bytes) Reassembled S7COMM-PLUS (3561 bytes) Decompressed Data (1181 bytes) Decompressed Data (1292 bytes) Decompressed Data (2403 bytes) Decompressed Data (274 bytes)

Value of one item (s7comm-plus.data.item_value), 220 bytes

Packets: 18140 · Displayed: 1475 (8.1%) Profile: Default

Object MAC

Object Code

Source Code

Create Object Request

Create Program Cycle Object Block

HMAC-SHA256 over packet with session key

Control Program Representation

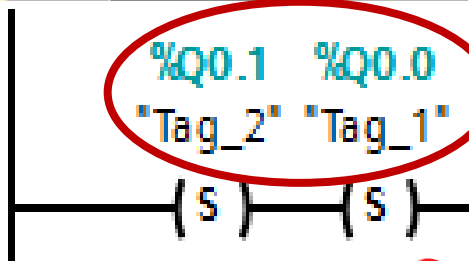
Object
MAC

```
0210 e8 ef be ad de 02 00 00 00 00 00 00 a3 9c 23
0220 00 08 00 a3 a3 62 00 14 00 15 01 00 08 00 00 00
0230 01 00 02 00 05 00 01 03 00 08 00 0f 00 00 00 a3
0240 bb 25 00 0c 00 00 00 00 a3 bd 17 00 17 00 00 07
0250 1c 8e 1d 00 04 00 8e 1e 00 17 00 00 07 08 8e 09
0260 00 04 00 8e 0a 00 02 00 8e 0b 00 17 00 00 07 21
0270 8e 22 00 05 00 8e 23 00 04 00 8e 24 00 04 00 00
0280 8e 0c 00 17 00 00 07 21 8e 22 00 05 00 8e 23 00
0290 04 00 8e 24 00 04 00 00 8e 0d 00 14 00 00 00 8e
02a0 1f 00 14 00 81 0c ea 1d ad ab 8c 00 00 00 01 00
02b0 00 00 00 00 00 a5 f2 7b 76 43 5d 0c 12 01 00
02c0 00 00 00 00 00 93 79 cc 34 23 63 e4 99 04 00
02d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02e0 00 00 00 00 00 86 92 5f df 76 c7 17 b4 7e 7d
02f0 5e 7f 39 34 06 f2 3a 5b 59 fe 65 8d 68 d4 77 d8
0300 60 01 7e 94 f5 41 97 d5 9f 60 27 83 64 41 2c eb
0310 d9 f5 c5 42 2e 37 d8 fa 1a 13 31 0f 74 44 ac cc
0320 b0 2e 12 bd ef 37 74 2d e0 d6 55 e1 25 72 32 06
0330 dc 52 00 a3 94 14 00 14 00 81 58 ef be ad de 7c
0340 00 00 00 01 00 00 00 02 00 00 00 32 00 00 00 00
0350 04 00 00 00 00 00 93 79 cc 34 23 63 e4 99 04
0360 00 00 00 00 00 00 a5 f2 7b 76 43 5d 0c 12 01
0370 00 00 00 00 00 00 3c 00 00 00 d7 3a d1 7a 6f
0380 5d a7 d3 3b 7f 7d e3 3f b0 32 83 6d 93 b2 61 cd
0390 e3 cf c9 e1 45 57 7c c0 83 6c f7 c9 3f eb a0 3f
03a0 54 43 cb cd 65 27 fe b2 b2 f6 4e a2 e9 30 31 3e
03b0 00 a9 90 6f 75 56 2f ef be ad de 00 00 00 20
03c0 00 00 00 d6 a6 62 46 72 d0 fc 43 ce 23 17 c9 be
03d0 ff 1d f1 33 26 92 8f 34 92 67 a1 83 74 b2 c9 61
03e0 39 c4 eb ef be ad de 01 00 00 00 18 00 00 00 4a
03f0 23 04 b3 0a e6 5d 3e 52 f9 f9 d1 31 b7 74 2b d6
0400 48 75 4a 0f 5a 81 fa ef be ad de 02 00 00 00 00
0410 00 00 00 a3 94 15 00 05 8a e2 8c b8 c6 ba 97 a5
0420 08 a3 94 1a 00 14 00 04 00 00 00 00 a3 94 1b 00
```

Object
Code

Source
Code

%Q0.1 %Q0.0
"Tag_2" "Tag_1"



Yellow Program



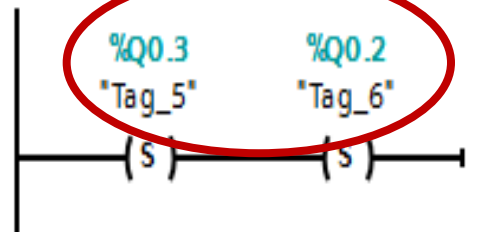
Object
MAC

```
0210 aa ef be ad de 02 00 00 00 00 00 00 a3 9c 23
0220 00 08 00 a3 a3 62 00 14 00 15 01 00 08 00 00 00
0230 01 00 02 00 05 00 01 03 00 08 00 0f 00 00 00 a3
0240 bb 25 00 0c 00 00 00 00 a3 bd 17 00 17 00 00 07
0250 1c 8e 1d 00 04 00 8e 1e 00 17 00 00 07 08 8e 09
0260 00 04 00 8e 0a 00 02 00 8e 0b 00 17 00 00 07 21
0270 8e 22 00 05 00 8e 23 00 04 00 8e 24 00 04 00 00
0280 8e 0c 00 17 00 00 07 21 8e 22 00 05 00 8e 23 00
0290 04 00 8e 24 00 04 00 00 8e 0d 00 14 00 00 00 8e
02a0 1f 00 14 00 81 0c ea 1d ad ab 8c 00 00 00 01 00
02b0 00 00 00 00 00 a5 f2 7b 76 43 5d 0c 12 01 00
02c0 00 00 00 00 00 93 79 cc 34 23 63 e4 99 04 00
02d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02e0 00 00 00 00 00 86 92 5f df 76 c7 17 b4 7e 7d
02f0 5e 7f 39 34 06 f2 3a 5b 59 fe 65 8d 68 d4 77 d8
0300 60 01 7e 94 f5 41 97 d5 9f 60 27 83 64 41 2c eb
0310 d9 f5 c5 42 2e 37 d8 fa 1a 13 31 0f 74 44 ac cc
0320 b0 2e b2 2b 07 94 cd 1d 1d eb 2d 74 ac 22 85 4e
0330 c1 79 00 a3 94 14 00 14 00 81 58 ef be ad de 7c
0340 00 00 00 01 00 00 00 02 00 00 00 32 00 00 00 00
0350 04 00 00 00 00 00 93 79 cc 34 23 63 e4 99 04
0360 00 00 00 00 00 00 a5 f2 7b 76 43 5d 0c 12 01
0370 00 00 00 00 00 00 3c 00 00 00 d7 3a d1 7a 6f
0380 5d a7 d3 3b 7f 7d e3 3f b0 32 83 6d 93 b2 61 cd
0390 e3 cf c9 e1 45 57 7c c0 83 6c f7 c9 3f eb a0 3f
03a0 54 43 cb cd 65 27 fe b2 b2 f6 4e a2 e9 30 31 3e
03b0 00 a9 90 6f 75 56 2f ef be ad de 00 00 00 20
03c0 00 00 00 d6 a6 62 46 72 d0 fc 43 ce 23 17 c9 be
03d0 ff 1d f1 2d a5 c5 91 8c af b6 fa 75 d5 1c 01 9b
03e0 03 69 9f ef be ad de 01 00 00 00 18 00 00 00 4a
03f0 23 06 b3 0a e4 5d 3e 87 c9 5b e7 51 7f 78 55 f9
0400 34 07 0b 69 b0 11 39 ef be ad de 02 00 00 00 00
0410 00 00 00 a3 94 15 00 05 8a e2 8c b8 c6 ba 97 a5
0420 08 a3 94 1a 00 14 00 04 00 00 00 00 a3 94 1b 00
```

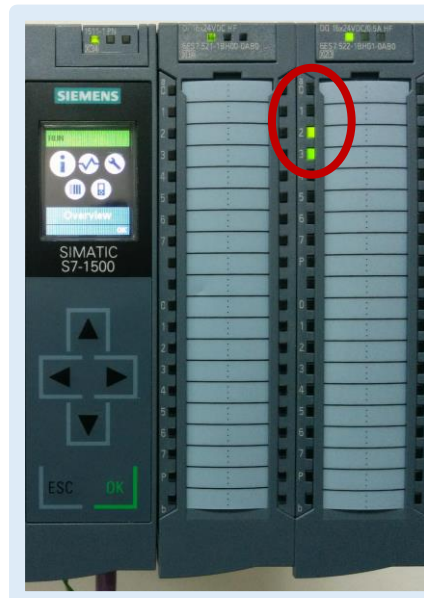
Object
Code

Source
Code

%Q0.3 %Q0.2
"Tag_5" "Tag_6"

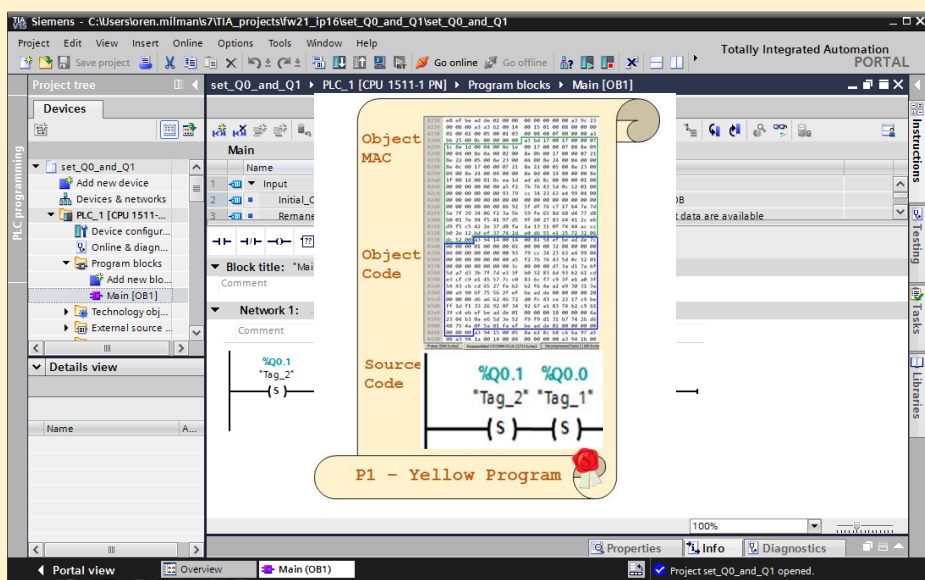


Blue Program



Program Download

Totally Integrated Automation PORTAL V15



Req, Hello, RID, Seq=1

Res, Hello, SID, n_2 , Model, Firmware version, Challenge, Seq=1

Req, SID, Encrypted Keying Material, Response, Seq=2

Res, OK, Seq=2

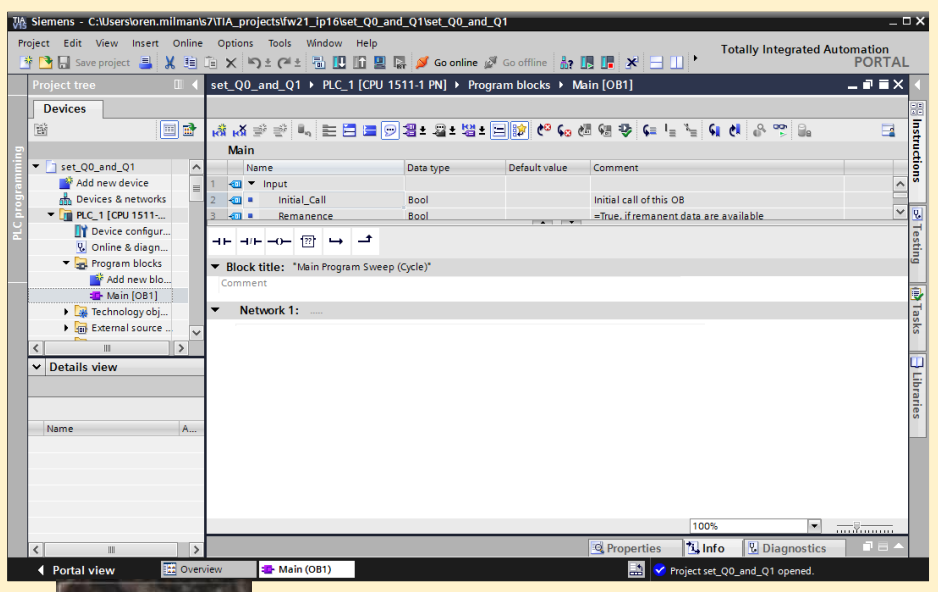
Session_key = f(Challenge, KDK)



Download

Program Upload

Totally Integrated Automation PORTAL V15



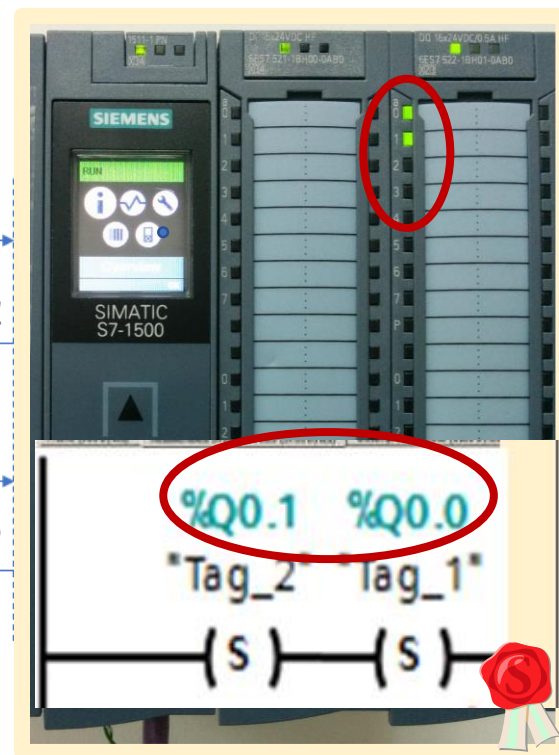
Req, Hello, RID, Seq=1

Res, Hello, SID, n_2 , Model, Firmware version, Challenge, Seq=1

Req, SID, Encrypted Keying Material, Response, Seq=2

Res, OK, Seq=2

Session_key = f(Challenge, KDK)



Upload

The Rogue TIA

- No engineering station in the production network.
- The attacker brings it with him.
- Rogue TIA: The attack system consists of a modified legitimate TIA and a malicious proxy.

**Rogue
TIA**



Rogue TIA Stealth Program Injection Setup Phase

Totally Integrated Automation
PORTAL V15



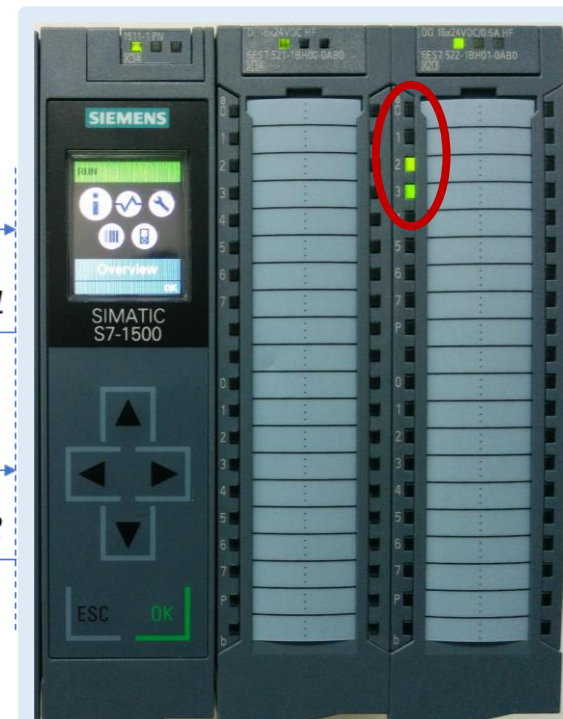
Req, Hello, RID, Seq=1

Res, Hello, SID, n_2 , Model, Firmware version, Challenge, Seq=1

Req, SID, Encrypted Keying Material, Response, Seq=2

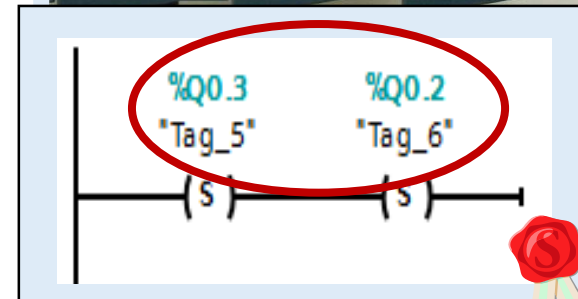
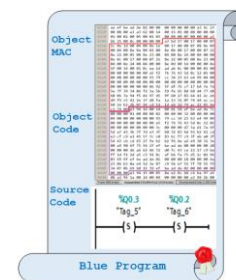
Res, OK, Seq=2

$Session_key = f(Challenge, KDK)$



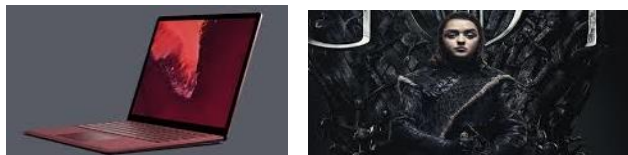
Download

Record



Rogue TIA Stealth Program Injection Attack Phase

Totally Integrated Automation
PORTAL V15



Req, Hello, RID, Seq=1

Res, Hello, S

Req, SID, Encrypte

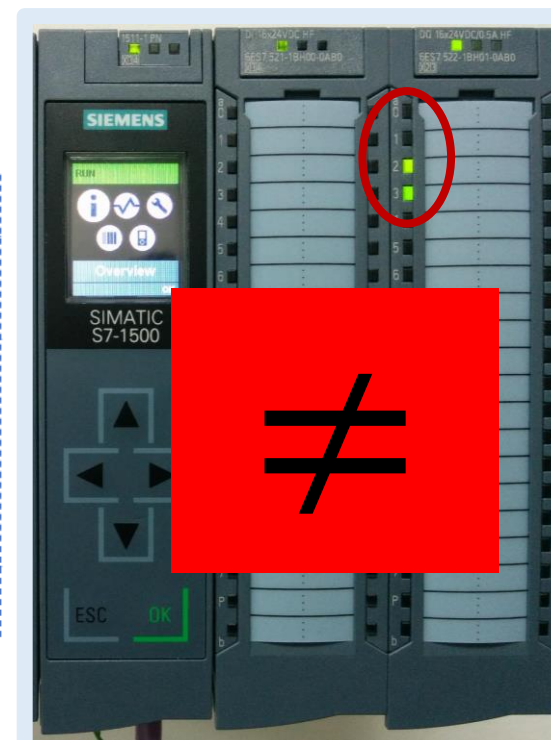
Seq

version, Challenge, Seq=1

onse, Seq=2

Res, OK, Seq=2

(DK)



Re-calc
integrity

Download

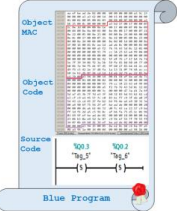
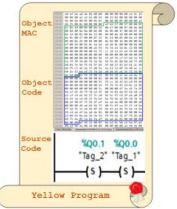
%Q0.1 %Q0.0
"Tag_2" "Tag_1"

But we cannot always bring
our own TIA with us....

Introducing

The Rogue Engineering Station

Rogue Engineering Station



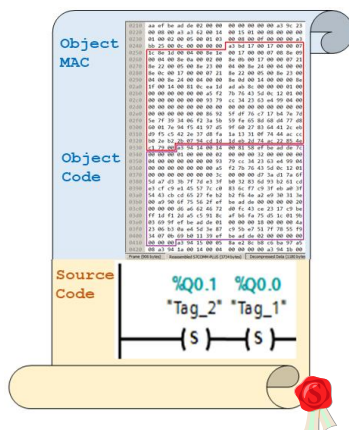
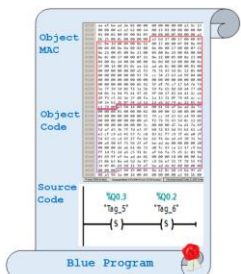
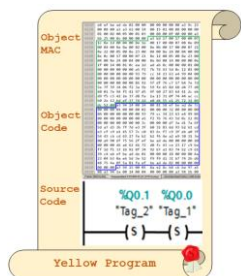
**Rogue
Engineering Station**



- A Python attack script that impersonates a TIA
- Inputs: PLC's IP address; Yellow and blue programs download pcap files

- Runs a live download session with the PLC
 - Four-way hand-shake
 - All S7 message fields are taken from the yellow pcap file
 - Except, the object code and MAC that are taken from blue pcap file
 - Fixes message SID, integrity protection and additional cookies

Rogue Engineering Workstation Stealth Program Injection



Fix SID
Fix other
cookies
Re-calc
integrity

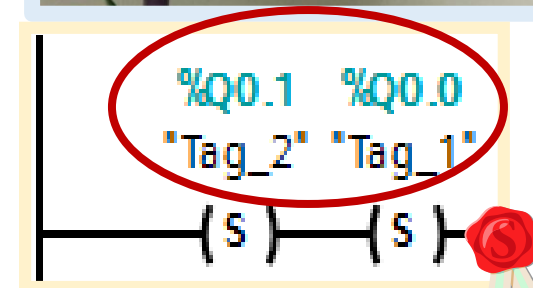
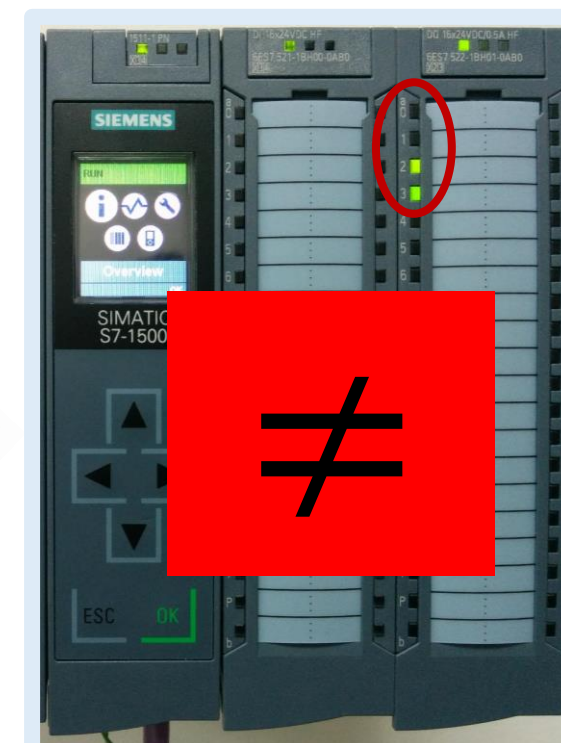
Req, Hello, RID, Seq=1

Res, Hello, SID, n_2 , Model, Firmware version, Challenge, Seq=1

Req, SID, Encrypted Keying material, Response, Seq=2

Res, OK, Seq=2

Session_key=f(Challenge, Keying material)



Download

Step7 Impersonation



My Lab



The Wall



King's Landing

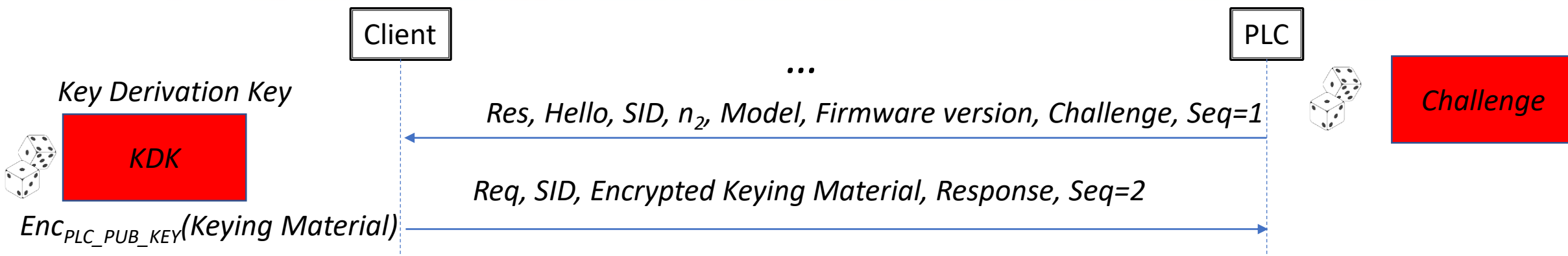


P3 Handshake Details



Deep technical details not for the faint-hearted

P3 Session Key Establishment



- Recall that:

$$Session_key = f(Challenge, KDK)$$

- What is this *EncryptedKeyingMaterial* Structure?

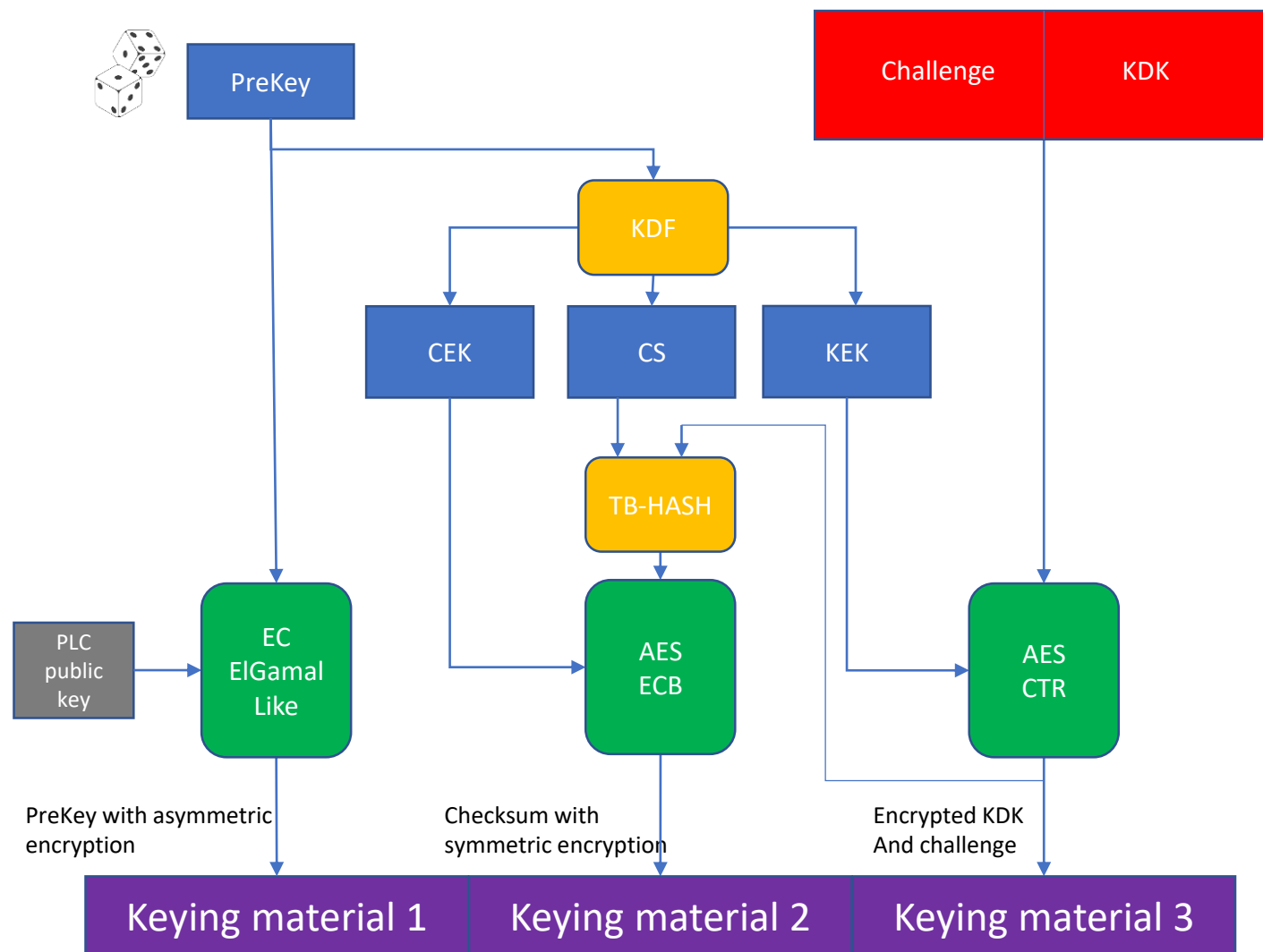
P3 Handshake

Cryptographic Primitives in Use

- Public-private key based asymmetric encryption
 - Elliptic curve: EC-ElGamal
- Symmetric encryption
 - AES
 - Electronic Code Book – ECB Mode
 - Counter mode
- Key Derivation Function: KDF
- Non-cryptographic checksum: Tabulation hash

P3 – KDK Sharing

1. Generate 20 bytes PreKey
 1. Encrypt it using EC-ElGamal-like encryption with the plc public key and add it to Keying material
2. Calculate KDF on PreKey and get
 1. Checksum Encryption Key (CEK)
 2. Checksum Seed (CS)
 3. Key Encryption Key (KEK)
3. Concatenate the KDK to the challenge, encrypt them using AES-CTR with the KEK, and add to Keying material
4. Initiate the Tabulation Hash with CS and calculate checksum over (3)
5. Encrypt (4) using AES-ECB with CEK and add to Keying material



P3 – Asymmetric Keys

- The public keys are stored in compressed .key files at
[TIA INSTALLATION]\Data\Hwcn\Custom
- Each key file contains
 - Metadata (version, key type, key family, etc.)
 - **Key data – PLC public key for the EC-ElGamal-like encryption**

P3 – An Example .key File

version: 1

orderNumber: s71500-connection

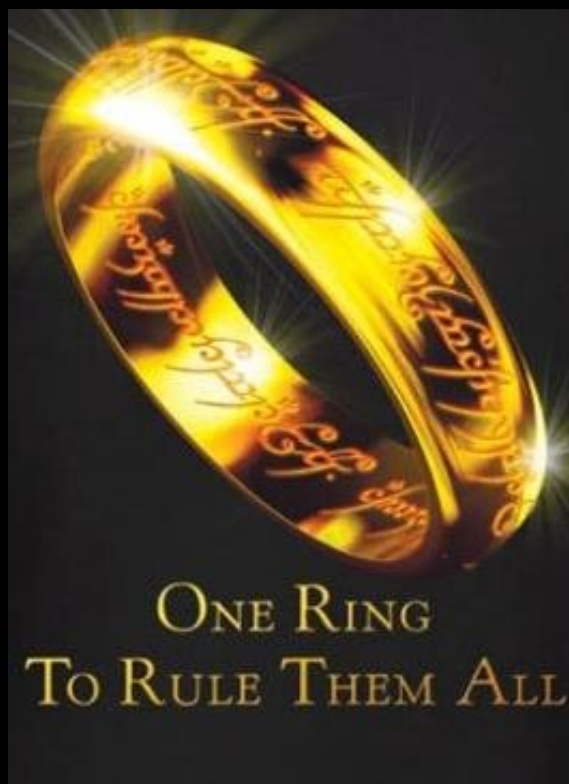
firmwareVersion:

keyType: connection

familyType: S7-1500

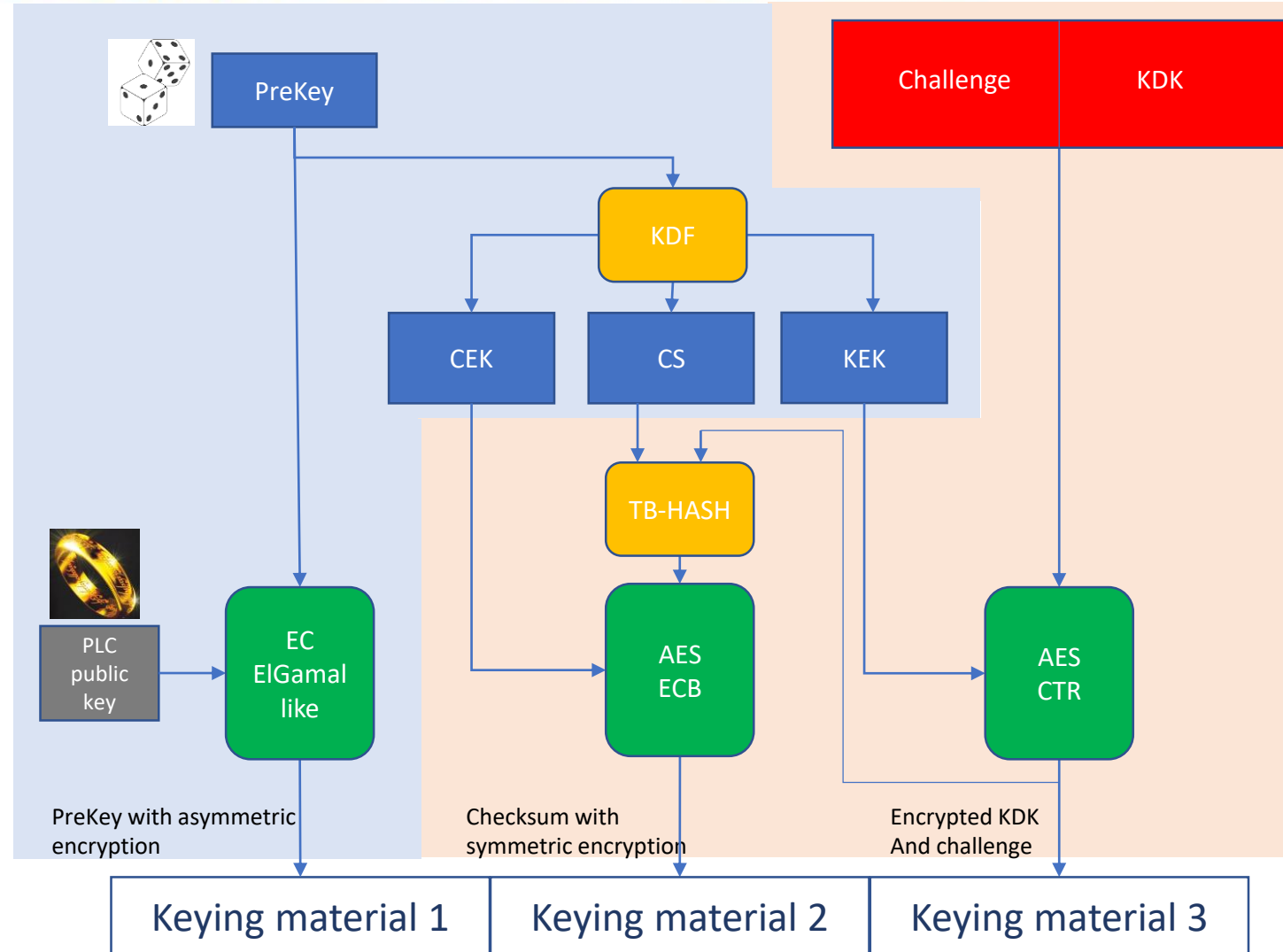
key data: 8456...

P3 Handshake Design Vulnerabilities



Rogue Engineering Station Session Key Establishment

1. Pre-calculate PreKey encryption and CEK, CS, KEK
2. Let Python do the symmetric encryptions and the checksum calculation to build keying material 2 & 3
3. The Python script wraps the session key derivation function from the relevant dll
 - We didn't reverse the session key derivation function f , due to lack of time



REing the TIA Handshake

- TIA is huge – find your target files
 - OMSp_core_managed.dll – main S7 communication DLL
 - Mixed mode DLL
 - Managed CIL bytecode – C#
 - Native (unmanaged) x86/x64 opcodes – C++
- Choose your tools
 - Managed code (.NET) - Reflector/dnSpy
 - Native C++ code - IDA Pro
- Improve your reverse engineering starting point



REing the TIA Handshake

- TIA is huge – find your target files
 - OMSp_core_managed.dll – main S7 communication DLL
 - Mixed mode DLL
 - Managed CIL bytecode – C#
 - Native (unmanaged) x86/x64 opcodes – C++
- Choose your tools
 - Managed code (.NET) - Reflector/dnSpy
 - Native C++ code - IDA Pro
- Improve your reverse engineering starting point



Tip #1 – Identify Native Code Entry Points

- omsp_core_managed.dll managed part opened in dnSpy

```
1669     public unsafe static Error SetServerPublicKeyGlobal(uint keyType, byte[] key)
1670     {
1671         if (key != null && key.Length != 0)
1672         {
1673             ref byte byte& = ref key[0];
1674             byte* ptr = ref byte&;
1675             uint num = (uint)key.Length;
1676             Blob blob;
1677             <Module>.OMS.Blob.{ctor}(ref blob);
1678             Error result;
1679             try
1680             {
1681                 long num2 = <Module>.OMS.StructValue.copy_data(ref blob, (void*)ptr, num, 0u, false);
1682                 if (((num2 & -9223372036854775808L) != 0L) ? 1 : 0) != 0)
1683                 {
1684                     Error error = new Error(ref num2);
1685                     result = error;
1686                 }
1687             }
1688             else
1689             {
1690                 num2 = <Module>.OMS.ClientSession.s_set_server_public_key(keyType, ref blob);
1691                 Error error2 = new Error(ref num2);
1692                 result = error2;
1693             }
1694         }
1695     }
```

- Let's click on `s_set_server_public_key`

Tip #1 – Identify Native Code Entry Points

```
13524 // Token: 0x0600037A RID: 890 RVA: 0x00109780 File Offset: 0x00108B80
13525 [SuppressUnmanagedCodeSecurity]
13526 [MethodImpl(MethodImplOptions.Unmanaged) | MethodImplOptions.PreserveSig]
13527 internal unsafe static extern long s_set_server_public_key(uint, OMS.Blob*);
```

- Wait...Where is the function body???
- Looks like it's an unmanaged function...
- How can we find its implementation?

Tip #1 – Identify Native Code Entry Points

- RVA for the native part

Image Base
(0x1800000000)
+ RVA
(0x109780)

=
0x180109780

```
13524 // Token: 0x0600037A RID: 890 RVA: 0x00109780 File Offset: 0x00108B80
13525 [SuppressUnmanagedCodeSecurity]
13526 [MethodImpl(MethodImplOptions.Unmanaged) MethodImplOptions.PreserveSig]]
13527 internal unsafe static extern long s_set_server_public_key(uint, OMS.Blob*);
```

```
.text:0000000180109780 OMS__ClientSession__s_set_server_public_key proc near
.text:0000000180109780
.text:0000000180109780 var_38          = dword ptr -38h
.text:0000000180109780 var_28          = xmmword ptr -28h
.text:0000000180109780 arg_0           = qword ptr 8
.text:0000000180109780 arg_8           = qword ptr 10h
.text:0000000180109780 arg_10          = qword ptr 18h
.text:0000000180109780 arg_18          = qword ptr 20h
.text:0000000180109780
.text:0000000180109780 ; __unwind { // j__CxxFrameHandler3
.text:0000000180109780 mov     rax, rsp
.text:0000000180109783 push    rdi
.text:0000000180109784 push    r14
.text:0000000180109786 push    r15
.text:0000000180109788 sub     rsp, 50h
.text:000000018010978C mov     qword ptr [rax-48h], 0FFFFFFFFFFFFFFEh
.text:0000000180109794 mov     [rax+8], rbx
.text:0000000180109798 mov     [rax+10h], rbp
.text:000000018010979C mov     [rax+20h], rsi
.text:00000001801097A0 mov     r15, rdx
.text:00000001801097A3 mov     ebp, ecx
.text:00000001801097A5 and     ebp, 0FF00h
.text:00000001801097AB or      ebp, 10h
.text:00000001801097AE mov     r14, cs:18096D0B0h
.text:00000001801097B5 add     r14, 2C8h
.text:00000001801097BC mov     [rax-40h], r14
.text:00000001801097C0 mov     rax, [r14]
.text:00000001801097C3 mov     r8, [rax+8]
.text:00000001801097C7 lea     rcx, OMS_Recursive_Thread_Mutex_vtable
```

Tip #2: C++ and RTTI

- Run-Time Type Information
- Allow C++ programmers to examine object types dynamically
- The information must be inside the binary
- We could use it

Tip #2: C++ and RTTI – Virtual Tables

1. Find RTTI Objects
2. Locate the relevant virtual tables

```
; const OMS::Parser::`RTTI Complete Object Locator'
??_R4Parser@OMS@@@6B@_5 dd 1          ; DATA XREF: .rdata:000000001806C69C0↑o
                                   ; .rdata:000000001807AAFE4↓o
                                   ; signature
                                   ; signature
dd 0                                ; offset of this vtable in complete class (from top)
                                   ; offset of this vtable in complete class (from top)
dd 0                                ; offset of constructor displacement
                                   ; offset of constructor displacement
dd rva ??_R0?AVParser@OMS@@@8 ; reference to type description
                                   ; reference to type description
dd rva ??_R3Parser@OMS@@@8 ; reference to hierarchy description
                                   ; reference to hierarchy description
dd rva ??_R4Parser@OMS@@@6B@_5 ; reference to object's base
                                   ; reference to object's base
db 0
db 0
```

Tip #2: C++ and RTTI – Virtual Tables

1. Find RTTI Objects
2. Locate the relevant virtual tables

The screenshot shows a debugger window with a list of references to a constant object. The window title is "xrefs to const OMS::Parser::`RTTI Complete Object Locator'". The list has two entries:

| Direction | Type | Address | Text |
|-----------|------|--------------------------|--|
| Up | o | .rdata:000000001806C69C0 | dq offset ??_R4Parser@OMS@@@6B@_5; const OMS::Parser::`RTTI Complete Object Loc... |
| D... | o | .rdata:000000001807AAFE4 | dd rva ??_R4Parser@OMS@@@6B@_5; reference to object's base |

Below the list are buttons for "OK", "Cancel", "Search", and "Help". The status bar at the bottom indicates "Line 1 of 2".

Tip #2: C++ and RTTI – Virtual Tables

1. Find RTTI Objects
2. Locate the relevant virtual tables
3. Rebuild them

```
.rdata:00000001806C69C0 dq offset ??_R4Parser@OMS@@6B@_5 ; const OMS::Parser::`RTTI Complete Object Locator'
.rdata:00000001806C69C8 off_1806C69C8 dq offset sub_1800ACA20 ; DATA XREF: sub_1800AC380+131↑o
.rdata:00000001806C69C8 dq offset sub_1800ACA20 ; sub_1800ACAB0+1C↑o ...
.rdata:00000001806C69D0 dq offset sub_1800AD020
.rdata:00000001806C69D8 dq offset sub_1800CC970
.rdata:00000001806C69E0 dq offset sub_1800ABC40
.rdata:00000001806C69E8 dq offset sub_1800AB8C0
.rdata:00000001806C69F0 dq offset sub_1800AB8D0
.rdata:00000001806C69F8 dq offset sub_1800ACD10
.rdata:00000001806C6A00 dq offset sub_1800CC820
```

Tip #2: C++ and RTTI – Virtual Tables

1. Find RTTI Objects
2. Locate the relevant virtual tables
3. Rebuild them

```
.rdata:00000001806C69C0      dq offset ?? R4Parser@OMS@@6B@_5 ; const OMS::Parser::`RTTI Complete Object Locator'
.rdata:00000001806C69C8 ParserOOMS_vtable ParserOOMS::vtable <offset ParserOOMS_sub_1800ACA20, \
.rdata:00000001806C69C8      ; DATA XREF: ParserOOMS__ParserOOMS+131fo
.rdata:00000001806C69C8      ; sub_1800ACAB0+1Cfo ...
.rdata:00000001806C69C8      offset ParserOOMS_sub_1800AD020, \
.rdata:00000001806C69C8      offset ParserOOMS_sub_1800CC970, \
.rdata:00000001806C69C8      offset ParserOOMS_sub_1800ABC40, \
.rdata:00000001806C69C8      offset ParserOOMS_sub_1800AB8C0, \
.rdata:00000001806C69C8      offset ParserOOMS_sub_1800AB8D0, \
.rdata:00000001806C69C8      offset ParserOOMS_sub_1800ACD10, \
.rdata:00000001806C69C8      offset ParserOOMS_sub_1800CC820>
```

Combine Them All

- Scan the .NET metadata and grab the native entry points
- Scan your .idb and grab the relevant RTTI data
- Use IDAPython to add this information to your .idb

Combine Them All - Example

```
1 signed __int64 __fastcall sub_1800ABC80(DWORD *a2, char a1)
2 {
3     DWORD *v2; // rdi
4     char v3; // si
5     __int64 v4; // rcx
6     __int64 v5; // rcx
7     DWORD *v6; // rbx
8     DWORD *v7; // rax
9
10    v2 = a2;
11    v3 = a1;
12    v4 = *((_QWORD *)a2 + 385);
13    if ( !v4 || (*(unsigned __int8 (**)(void))((_QWORD *)v4 + 8i64))() != a1 )
14    {
15        v5 = *((_QWORD *)v2 + 385);
16        v6 = 0i64;
17        if ( v5 )
18        {
19            v6 = *(DWORD **)(v5 + 32);
20            (**(void (__fastcall **)(__int64, signed __int64))v5)(v5, 1i64);
21        }
22        v7 = (DWORD *)sub_18017DBE0(v3, v2);
23        *((_QWORD *)v2 + 385) = v7;
24        if ( !v7 )
25            return -6701032989083762692i64;
26        if ( v6 )
27            *((_QWORD *)v7 + 4) = v6;
28    }
29    return 0i64;
30 }
```

Combine Them All - Example

```
1 signed __int64 __fastcall ParserOOMS::createScanner(ParserOOMS *this, char scanner_type)
2 {
3     PackedScannerOOMS *v4; // rcx
4     PackedScannerOOMS *old_scanner; // rcx
5     ACE_Message_Block *v6; // rbx
6     PackedScannerOOMS *v7; // rax
7
8     v4 = this->packedScanner;
9     if ( !v4 || (unsigned __int8)v4->vtable->PackedScannerOOMS::get_default_scanner_type() != scanner_type )
10    {
11        old_scanner = this->packedScanner;
12        v6 = 0i64;
13        if ( old_scanner )
14        {
15            v6 = old_scanner->base.message_block;
16            old_scanner->vtable->PackedScannerOOMS::destructor(old_scanner, 1);
17        }
18        v7 = f_create_scanner(scanner_type, this);
19        this->packedScanner = v7;
20        if ( !v7 )
21            return -6701032989083762692i64;
22        if ( v6 )
23            v7->base.message_block = v6;
24    }
25    return 0i64;
26 }
```

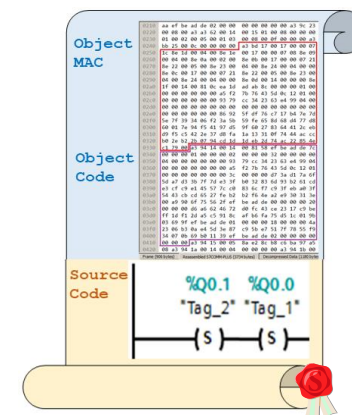
- Much better, isn't it?

Rogue Engineering Station Stealth Program Injection

Attack Demo

Summary

- Vulnerabilities in the S7 protocol – P3
 - TIA is not authenticated
 - “One Ring to Rule them All”
- A Python attack tool that impersonates TIA
 - Download a recorded program to any S7-1500 PLC
 - Stealth program injection attack



Thank You!