安全开发者峰会

工控系统供应链攻击大揭秘

高　剑

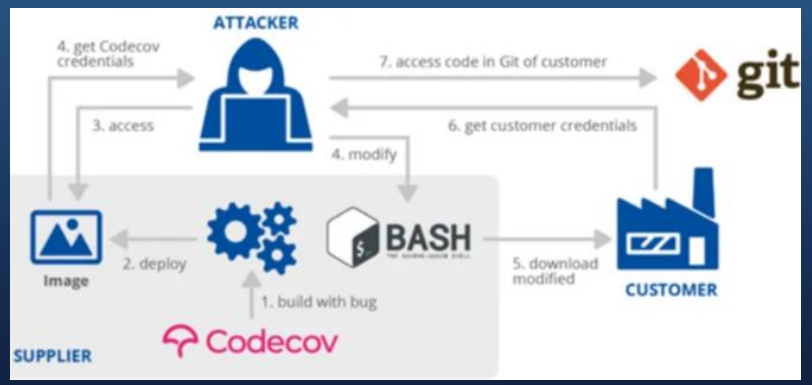
自我介绍

- 高 剑
- 工作单位：宁波和利时信息安全研究院
- 工作经历：10年工控厂商+4年安全厂商
- 研究方向：工控系统及设备漏洞挖掘与分析、工控业务场景风险评估与系统安全测试
- 已获得50多个CVE、CNVD等编号（ Siemens、ABB、Codesys、Schneider、等）
- BlackHat EU 2022、 KCON 2022、 HITCON 2021、 ICS Cyber security conference 2021、HITB AMS & SIN 2021、看雪 SDC 2020 、CIS 2020演讲嘉宾、工联众测大讲堂讲师、 “方班”企业导师等
- 入选Siemens、Schneider等国际知名工控厂商名人堂（hall-of-thanks）

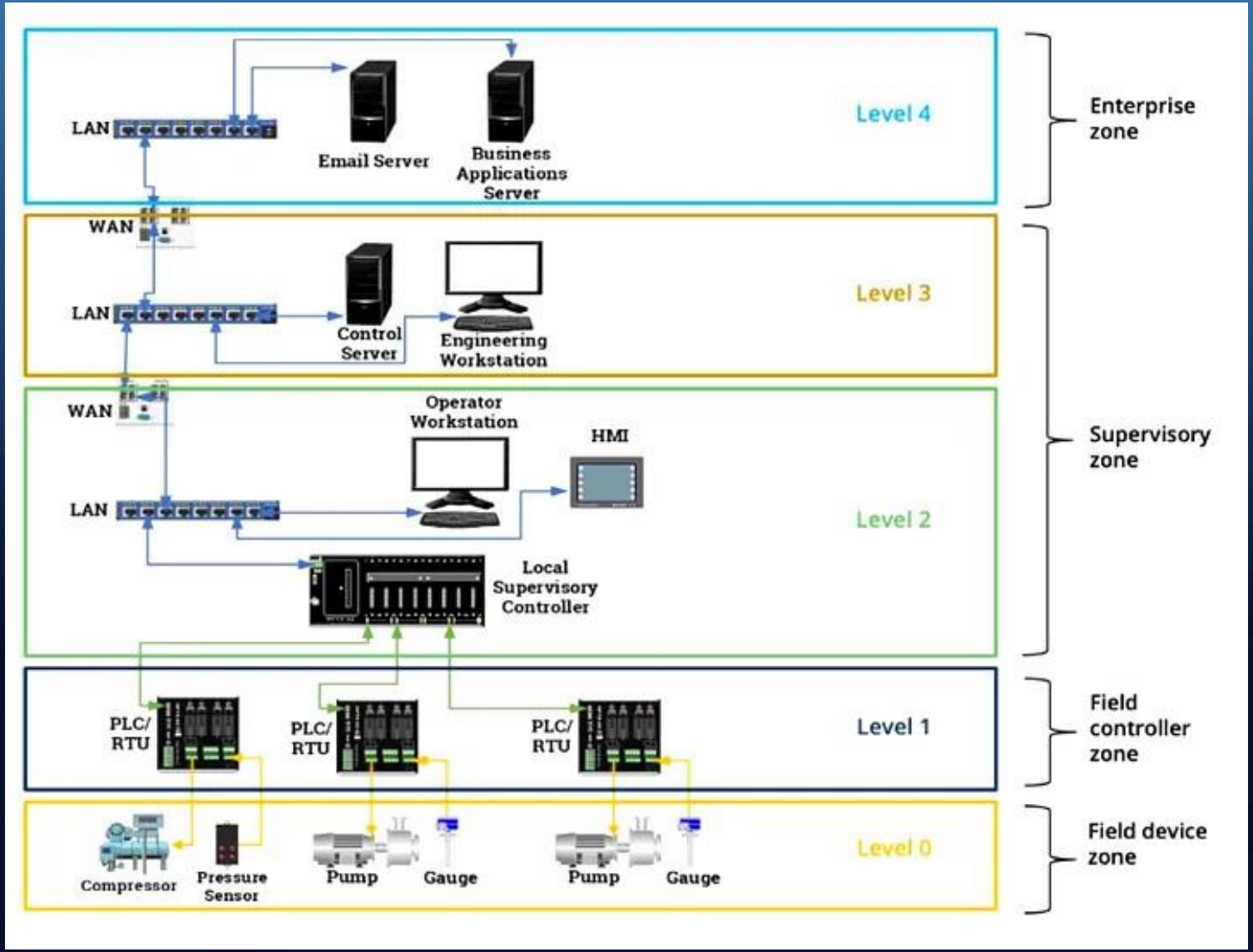
目 录

供应链攻击是什么

供应链攻击使用**第三方组件或服务**（统称为“供应链”）来渗透目标系统、设备或网络。供应链攻击是间接的：它们针对其最终目标所使用的第三方依赖项。依赖项是指由**第三方提供商提供的硬件、程序、代码段或服务。**



工控系统中的供应链


LAN
Email Server
Business Applications Server
Level 4
Enterprise zone
WAN
LAN
Control Server
Engineering Workstation
Level 3
WAN
LAN
Operator Workstation
HMI
Local Supervisory Controller
Level 2
Supervisory zone
PLC/ RTU
PLC/ RTU
PLC/ RTU
Level 1
Field controller zone
Compressor
Pressure Sensor
Pump
Gauge
Pump
Gauge
Level 0
Field device zone


HTML & CSS
WIDGET APPLICATION
C++
Qt


Microsoft
Windows xp


Windows


Android


Windows
Embedded CE


OPC UA


Modbus
TCP/IP


MQTT


CODESYS


infoteam
software


WIBU
SYSTEMS


PROFI
BUS
PROFI
NET


EtherNet/IP

工控系统中供应链安全现状

基础组件漏洞多

工控系统中使用较多的基础组件存在大量漏洞，比如OPC UA组件，从2017年至今多个厂商的组件暴漏出大量漏洞，影响产品数以万计，严重威胁到工业现场运行。除此之外还有使用较多的基础操作系统更为严重，比如VxWorks、Linux等。

欠缺对第三方依赖审查机制

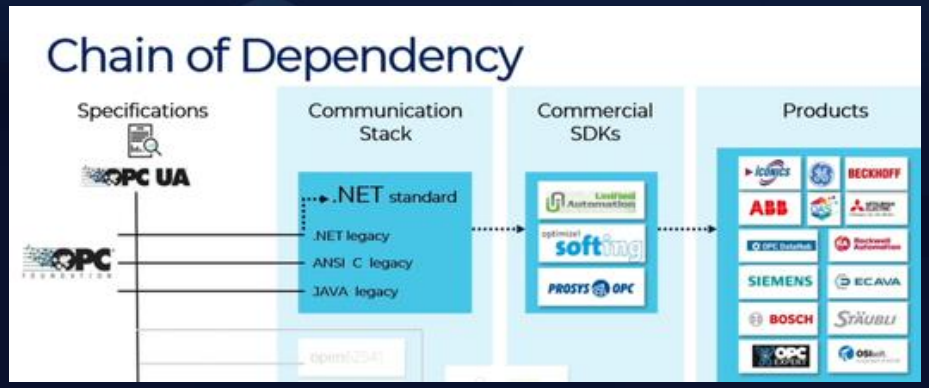
开发或者使用过程往往是直接引入第三方依赖的源代码、组件、服务等，没有针对性的检测或者审查机制，导致安全风险剧增。

工业产品供应渠道易遭劫持

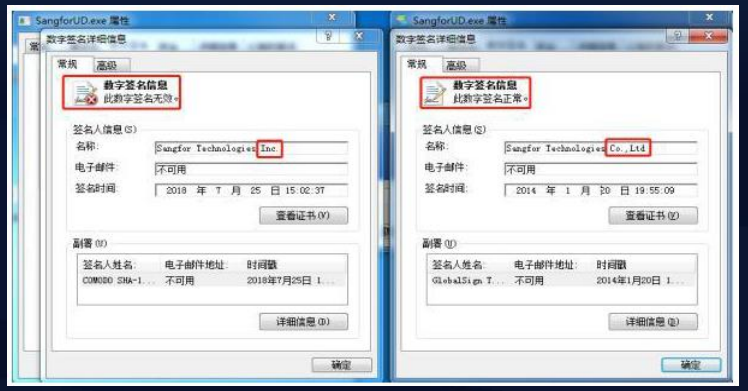
工业产品在采购、销售、物流等供应渠道中被劫持和篡改，攻击者在产品中构建后门或漏洞以实现入侵。“方程式”组织拥有的超级信息武器库，包括能对数十种常见品牌硬件固件重编程的恶意模块；该攻击可以通过在特定目标采购、返修主机或硬盘过程中修改硬盘固件程序实现。BP机爆炸事件也是此类攻击过程。

工业软件升级过程不安全

第三方软件产品在整个生命周期中要进行功能升级、补丁修复等更新，攻击者通过劫持软件升级过程中的更新模块或下载链接，在工业软件中植入恶意代码。微软蓝屏事件正是此种安全威胁，即便是升级过程中的小缺陷都使全球相关行业受到影响。

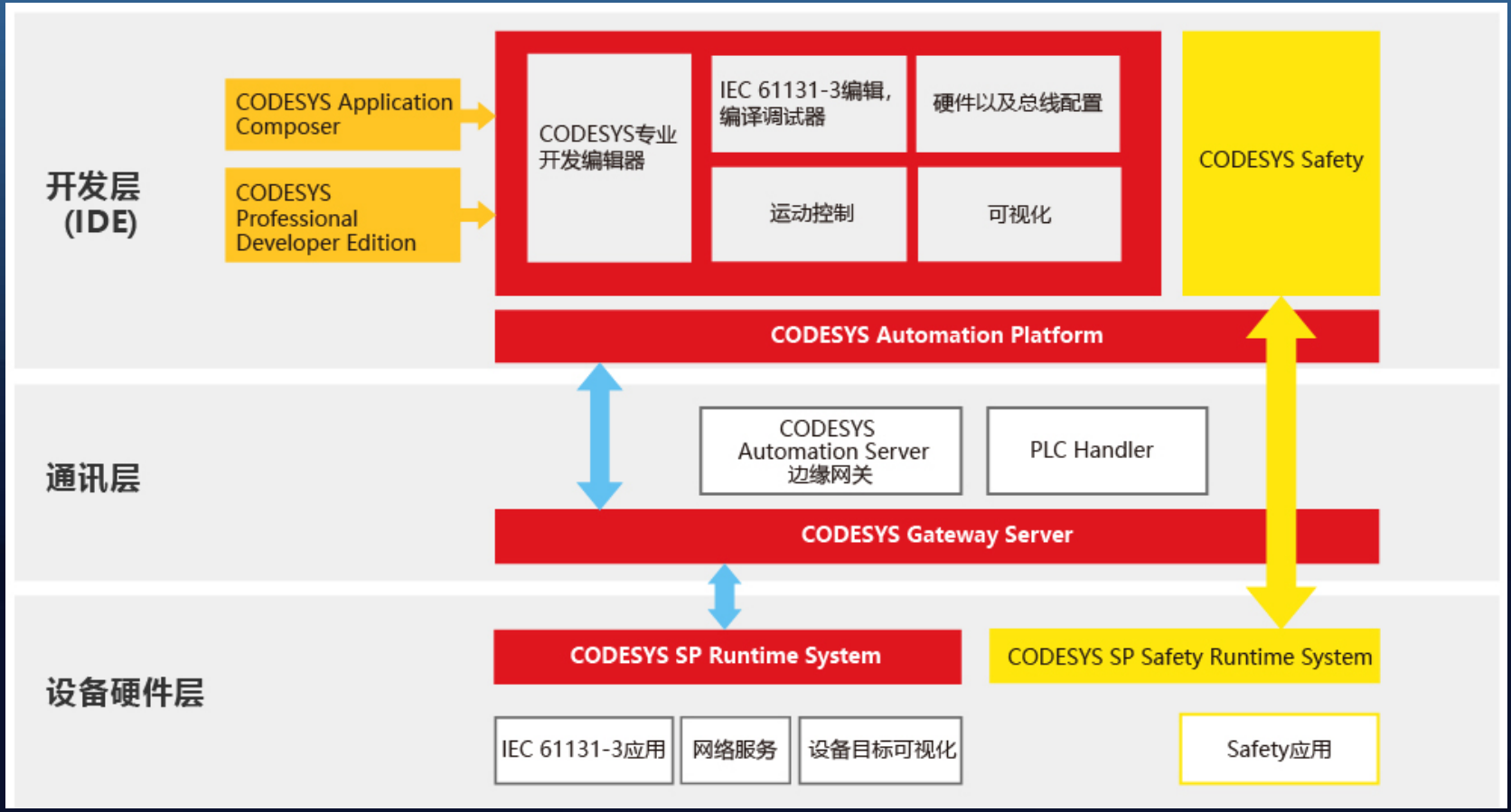


(TS//SI//NF) Left: Intercepted packages are opened carefully; Right: A “load station” implants a beacon



工业大脑中广泛使用的Codesys方案

关注最接近控制对象的工业大脑——工业控制器，目前全球工业控制器**使用最多的第三方解决方案-Codesys**



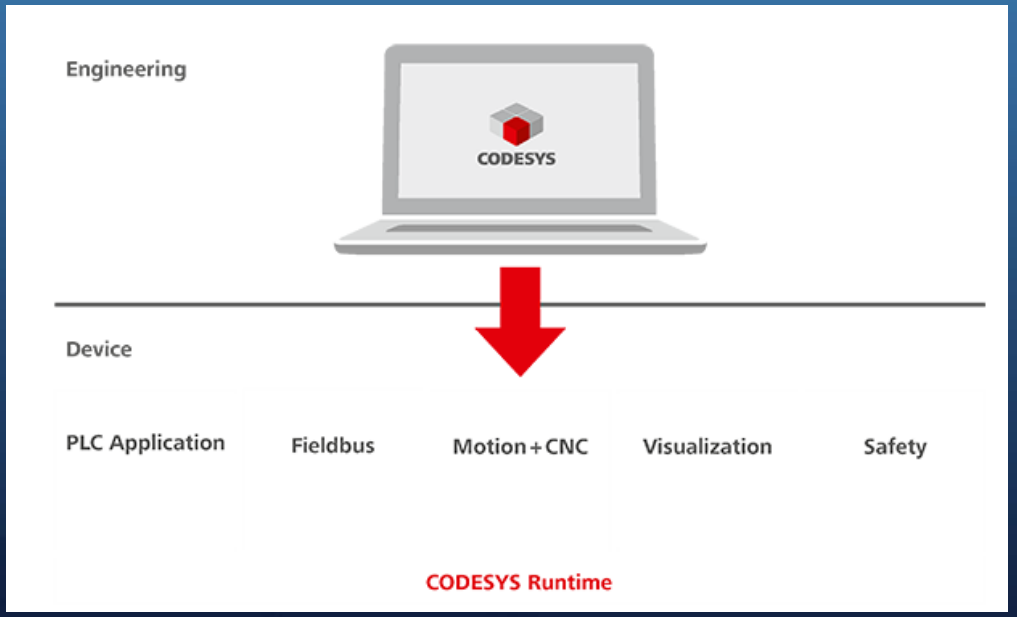
快速开发、快速上市
多平台支持、功能强大

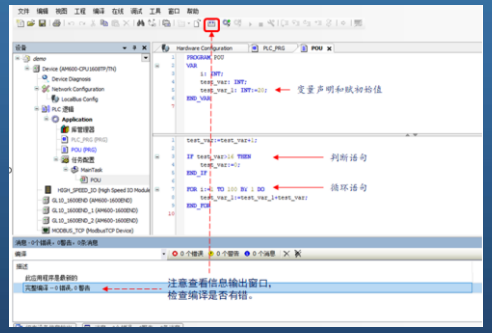
CODESYS Runtime System（运行时系统软件），它具有强大的跨平台可移植性，能够完美支持全球主流的 CPU 芯片（Intel/AMD/ARM/MIPS/R）和操作系统(Windows/Linux/QNX/VxWorks)。

自 2020 年起，CODESYS已全面支持国产CPU和国产操作系统。

* CPU 芯片：龙芯、全志、飞腾、瑞芯微、芯驰、兆易创新等；
* OS（操作系统）：SylixOS、OneOS、ReWorks、Loongnix等。

Codesys方案在PLC中的应用





- Codesys方案包括两部分：Development System和Runtime System。
- Development System是用来编程的软件（就像Visual Studio、Eclipse等软件，也可以称为IDE），设计、调试、编译PLC程序都在IDE中进行；IDE一般安装在开发者的电脑上。
- Runtime System则位于起控制作用的硬件设备上比如PLC，在IDE上编译的程序通过网络下载到Runtime中运行。

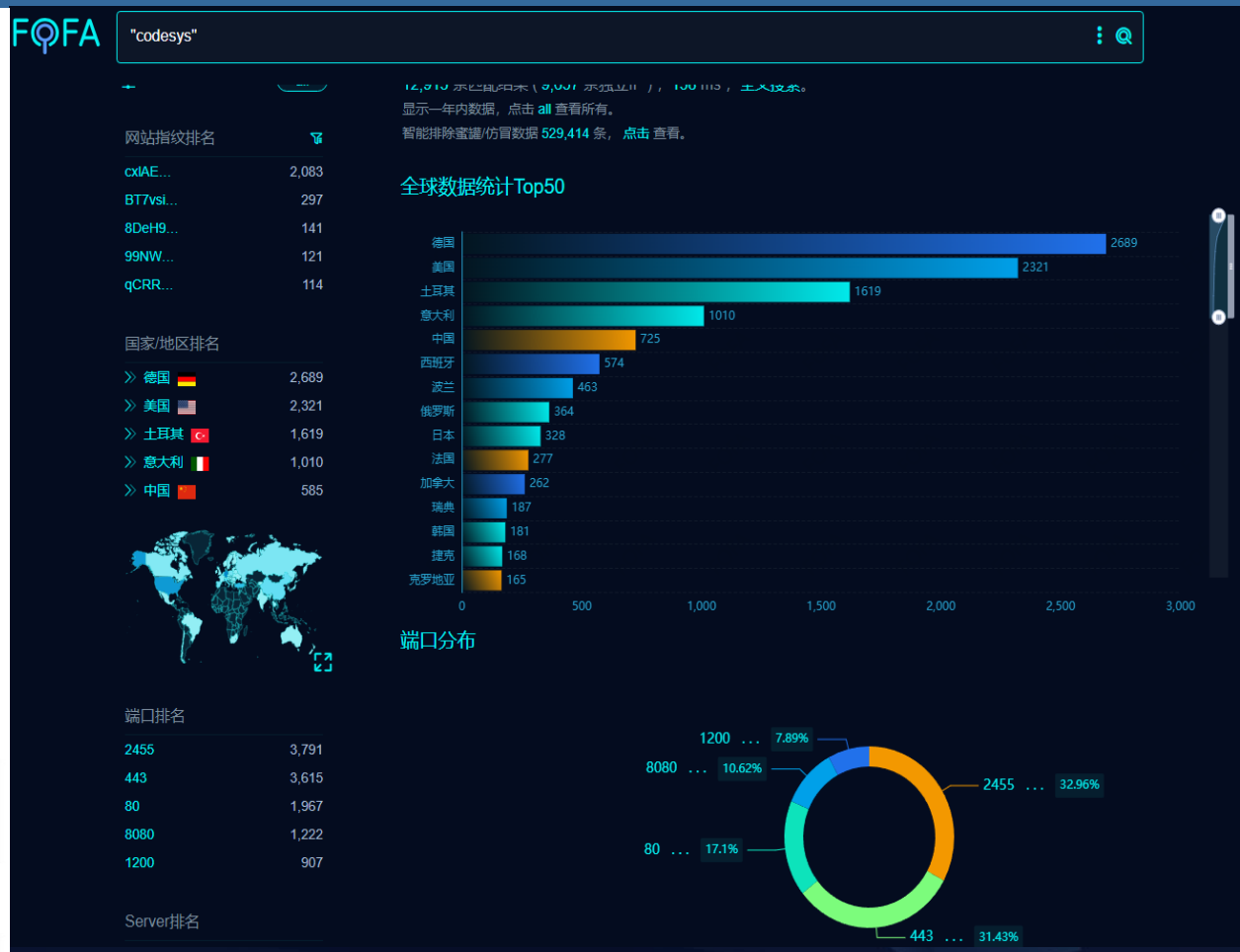
CODESYS 在工厂自动化中的应用

- 塑料机械
- 玻璃成型机
- 滚压机
- 数控机床
- 轮胎成型机
- 木工机械
- 涂布机
- 雕刻机
- 贴标机
- 激光和等离子切割机
- 纺织机械
- 纸张处理机械
- 包装机
- 卷烟机
- AGV
- 装配生产线
- 工业装卸机
- 印刷包装机械
- 灌装机
- 工业机器人

Codesys 内核控制器遍布全球

全球约有 600 家的控制系统生厂商和10000多家设备制造商是 CODESYS 软件的用户。

ABB
施耐德电气
博世力士乐
倍福
万可电子
博世
三菱电机
欧姆龙
费斯托
芯控智能
松下
日立
科控
伦茨
图尔克
易福门电子
伊顿
阿尔斯通
利勃海尔
丹佛斯
控创
路斯特
SEW
科比传动
巴合曼
EPEC
曼罗兰
穆格
豪迈集团
北尔电子



Codesys security advisories = 工业控制器安全?

CODESYS Security Advisories

Letzte Änderung	Advisory-Nummer	Advisory (PDF)
24.09.2024	2024-05	CODESYS Control V3 web server
12.09.2024	2024-04	OSCAT Basic Library
05.07.2024	2024-03	CODESYS Control V3 - OPC UA Stack
05.06.2024	2024-02	CODESYS Control Win and CODESYS (Edge) Gateway for Windows
06.05.2024	2024-01	CODESYS Development System V2.3
26.02.2024	2023-11	CODESYS Control V3 on Linux and QNX operating systems
05.12.2023	2023-10	CODESYS products containing WIBU CodeMeter Runtime
31.10.2023	2023-07	CODESYS Development System V3
31.10.2023	2023-05	CODESYS Control V3
03.08.2023	2023-08	CODESYS Development System V3
03.08.2023	2023-06	CODESYS Development System V3

- 如此大厂，肯定会重视安全问题，你说的供应链攻击不存在。？？？
- 合作厂商肯定会看安全公告，肯定知道自己产品有安全问题的。？？？
- 新售卖的产品当然是最新的runtime内核版本啊，不可能给终端用户用有漏洞的版本吧。？？？
- 漏扫扫出来后，终端用户及时打补丁升级固件不就可以了么，没你说的那么严重。？？？

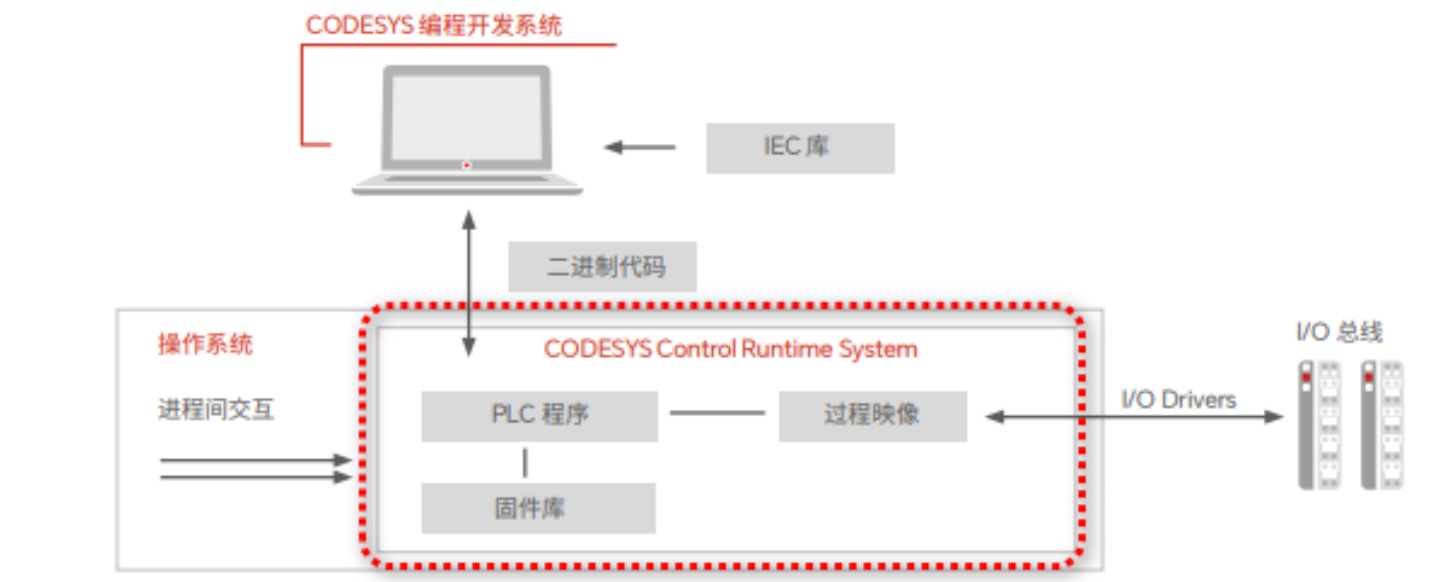
```
LogAdd(0, 1u, 32, 0, 0, "================================================================================");
LogAdd(0, 1u, 1, 0, 4, "%s V3", "CODESYS Control");
LogAdd(0, 1u, 1, 0, 5, "Copyright (c) 3S - Smart Software Solutions GmbH");
LogAdd(0, 1u, 1, 0, 6, "<version>%s</version> <builddate>%s</builddate>", "3.5.11.50", "Dec  3 2021");
LogAdd(0, 1u, 32, 0, 0, "================================================================================");
return 0;
```

```
}
else
{
  sub_87FE0("%s V%s for %s-%s - build %s\n", "CODESYS Control", "3.5.15.50", "CORTEX", "32Bit", "Jul  8 2021");
}
++v3;
```

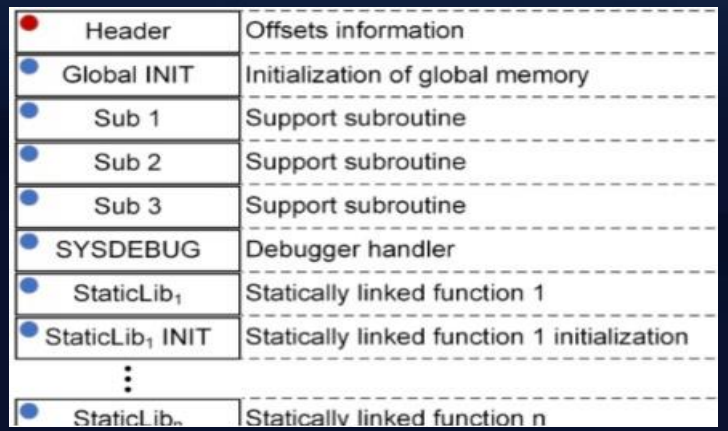
•**永久许可**：购买者可以永久使用所选版本的功能，无需支付额外费用。

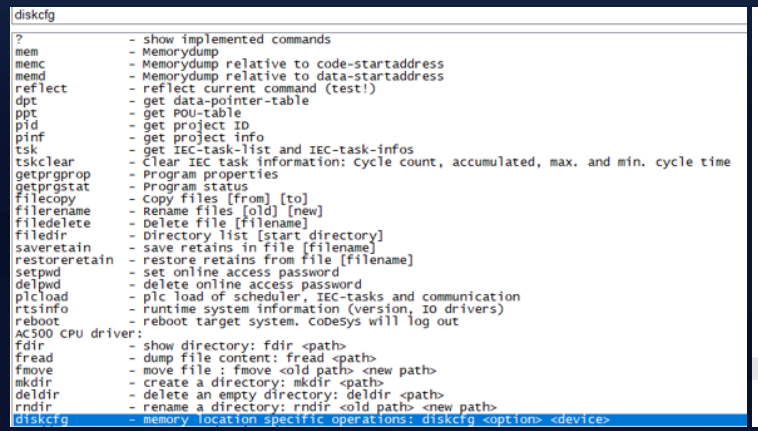
•**订阅许可**：通过定期付费（通常按年计费）获得访问最新功能和持续支持的权利。这种方式更适合那些需要频繁更新软件以获取最新技术和安全补丁的企业。

Codesys 内核攻击面分析



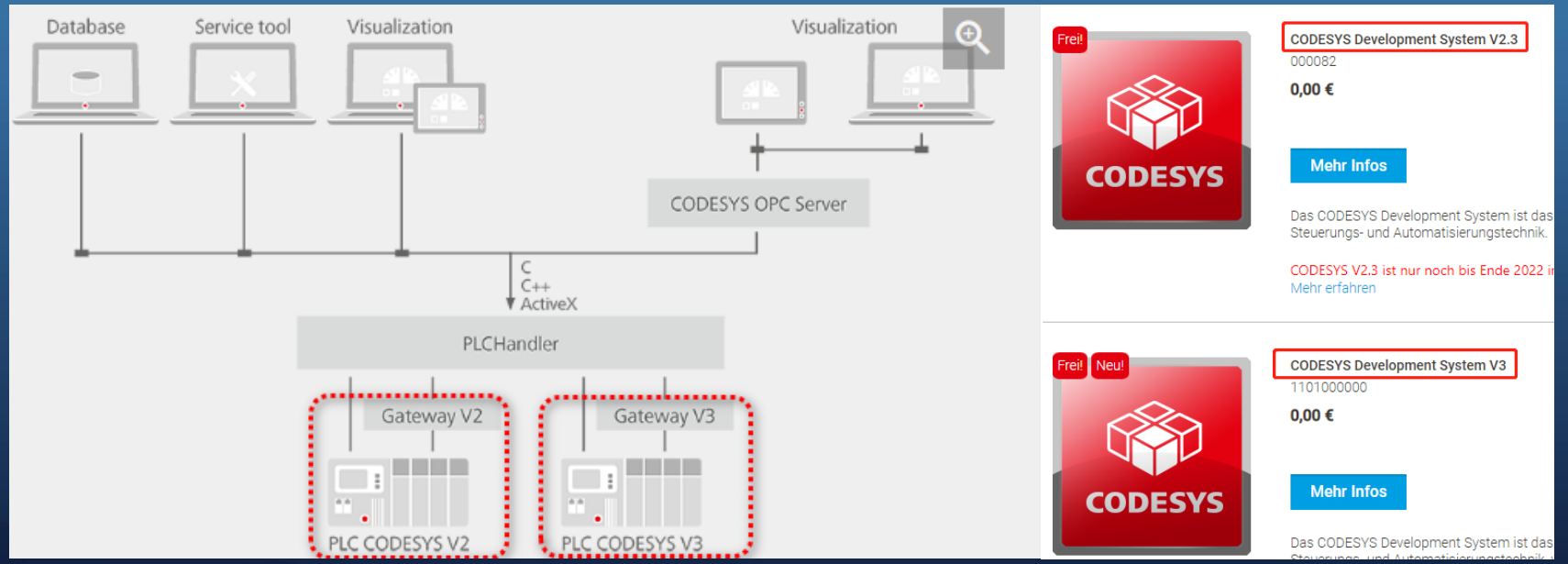
- 工程文件解析
- Shell命令
- **私有通讯协议**
- 自定义的算法库
- 固件升级
- 总线协议滥用及伪造攻击
- ……






```
case 0x3F:
  sub_14CBE0("SRV: RTS_VISU_READY\n");
  **(_WORD **)(a1 + 40) = 0;
  goto LABEL_413;
case 0x40:
  sub_14CBE0("SRV: RTS_DOWNLOAD_PRJINFO\n");
  v11 = *(_WORD **)(a1 + 40);
  *v11 = sub_18DA20(*(_DWORD *)(a1 + 52) + 2, *(_DWORD *)(a1 + 56) - 6);
  goto LABEL_413;
case 0x41:
  if ( *(_DWORD *)(v6 - 18948) )
  {
    v12 = sub_1EDC90("DEFAULT.BAK", dword_1D1F3C);
    if ( v12 )
      sub_1EDCD0(v12);
    else
      sub_14CBE0("*** RtsSrv::RTS_CHECKBOOTPRJ: could not open file: %s\n", "DEFAULT.BAK");
    v13 = "DEFAULT.BAK";
  }
```

Codesys V2通讯协议



- Codesys分为V2版本与V3版本
- V2版本属于早期版本，工业现场服役的控制器使用较多
- V3版本属于流行版本，现阶段国内外大量控制器均在使用

报文头	长度域	功能码	数据部分
bb bb	00 00 00 0c	92	payload.....

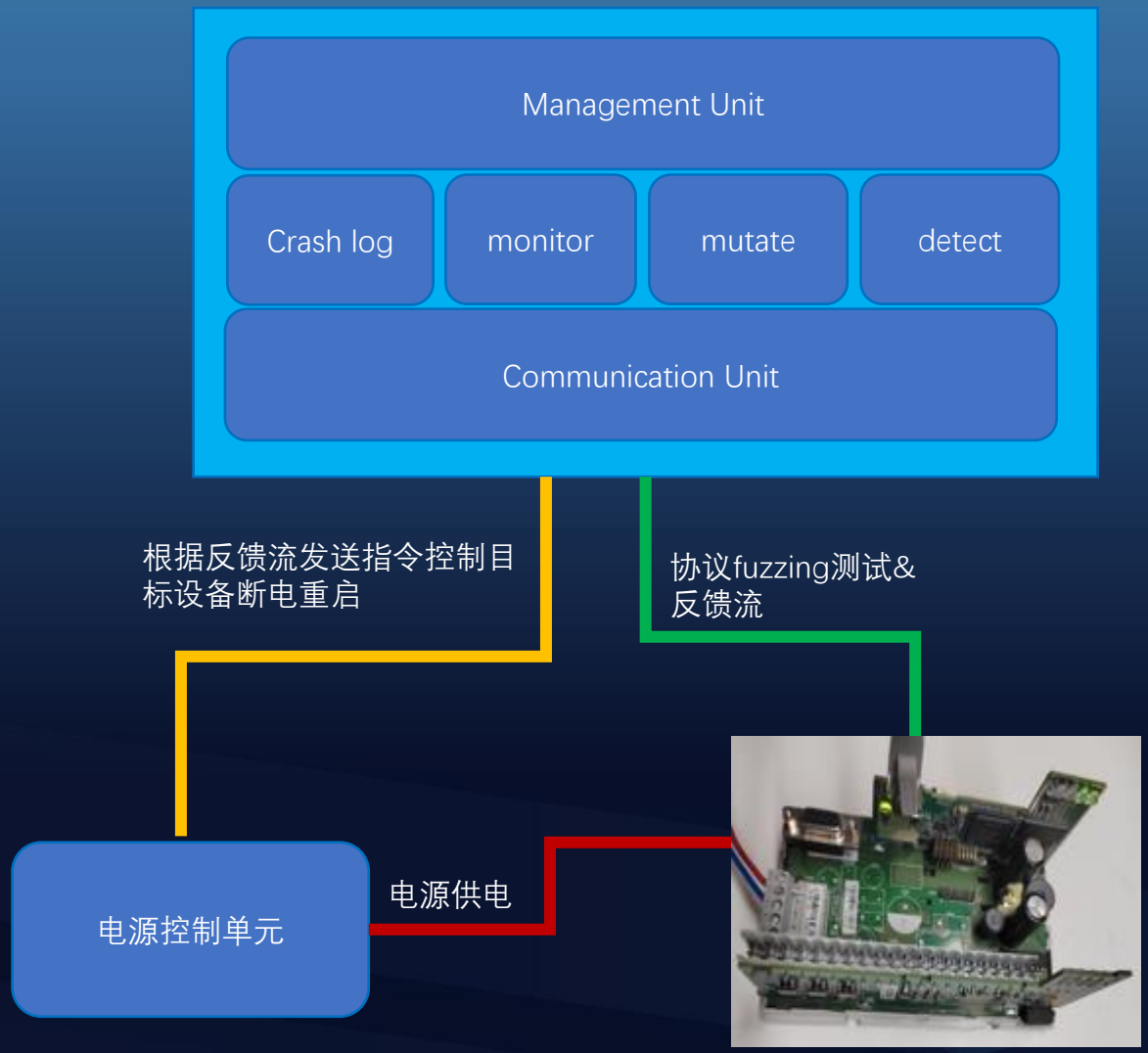
```
Frame 41: 72 bytes on wire (576 bits), 72 bytes captured (576
Ethernet II, Src: AsixElectron_51:49:50 (00:0e:c6:51:49:50),
Internet Protocol Version 4, Src: 192.168.0.33, Dst: 192.168.
Transmission Control Protocol, Src Port: 53803, Dst Port: 120
Codesys V2 Protocol Data
    Header: 0xbbbb
    Length: 12
    Function Code: 0x92 (RTS_BROWSERCOMMAND)
    Payload: 00000000727473696e666f
```

```
0000  00 24 59 0a 18 63 00 0e  c6 51 49 50 08 00 45 00   ·$Y··c·· ·QIP··E·
0010  00 3a f0 d8 40 00 80 06  00 00 c0 a8 00 21 c0 a8   ·:··@··· ·····!··
0020  00 42 d2 2b 04 b1 e7 94  64 28 00 00 19 c2 50 18   ·B·+···· d(····P·
0030  fa f0 81 e0 00 00 bb bb  00 00 00 0c 92 00 00 00   ········ ········
0040  00 72 74 73 69 6e 66 6f                           ·rtsinfo
```

```
local function get_function_description(code)
    local descriptions = {
        [0x01] = "RTS_LOGIN",
        [0x02] = "RTS_LOGOUT",
        [0x03] = "RTS_START",
        [0x04] = "RTS_STOP",
        [0x05] = "RTS_READ_VAR",
        [0x06] = "RTS_WRITE_VAR",
        [0x0A] = "RTS_STEP_OVER",
        [0x0B] = "RTS_STEP_OVER",
        [0x0C] = "RTS_BP_SET",
        [0x0D] = "RTS_BP_DEL",
        [0x0E] = "RTS_BP_DEL_ALL",
        [0x10] = "RTS_READ_STATUS",
        [0x11] = "RTS_READ_IDENTITY",
        [0x12] = "RTS_READ_BP_LIST",
        [0x13] = "RTS_RESET",
```

V2通讯协议Fuzzing测试-环境构建

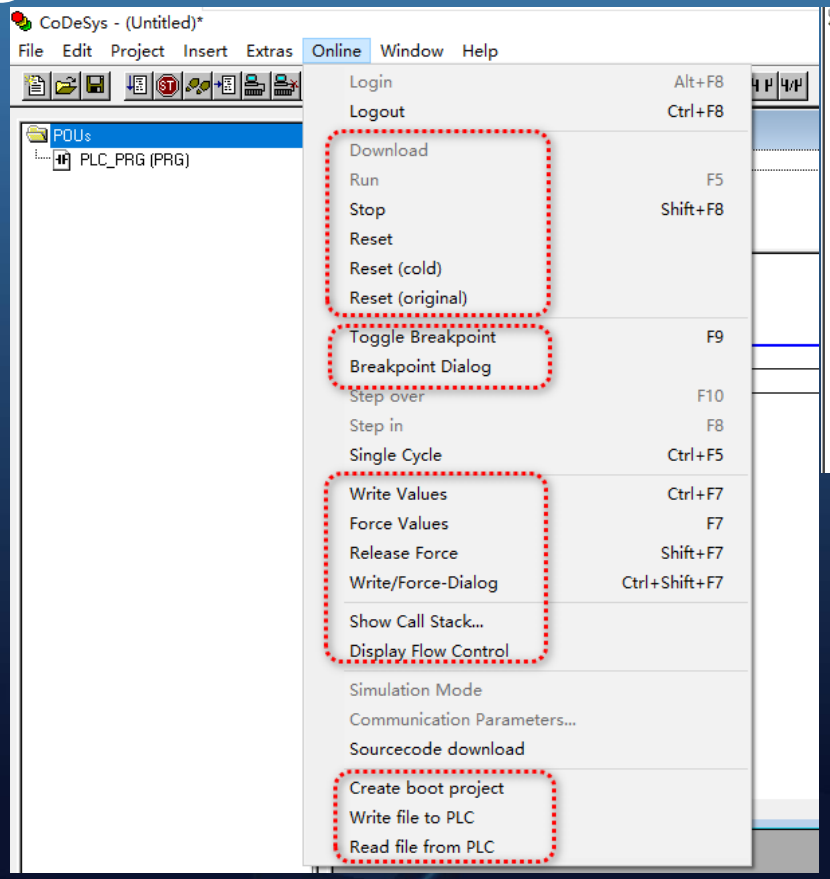
- 使用基于变异的模糊测试技术-更高效
- 在模糊测试过程中保存流量有利于重现和分析漏洞
- 变异算法可以采用融合式的变异算法；
- 崩溃日志和完整的流量日志需要备份，用以快速识别和自动化复现漏洞
- 监视和检测模块可以根据目标的特点设计-利用TCP的链接或者利用指示灯等信息做异常判据
- 加入电源控制模块的作用：当待测试目标进入故障状态时重启设备以继续进行FUZZ测试，提高测试效率实现fuzzing测试的自动化

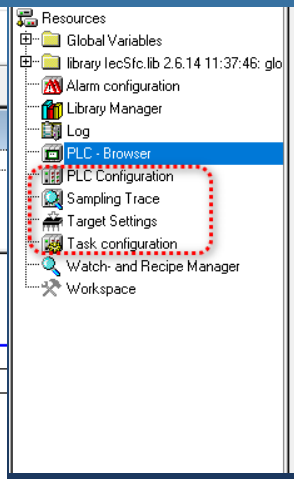


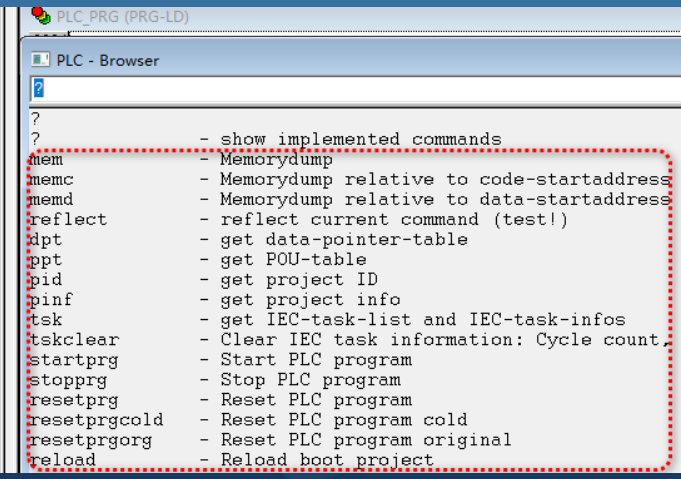
某国际知名品牌PLC

V2通讯协议Fuzzing测试-样本生成

- 丰富的样本会覆盖更多的路径找到更多的漏洞
- 在IDE编程软件上进行各种控制器相关操作，抓取报文，处理后得到fuzzing样本文件
- 尽可能多的覆盖CodesysV2协议中的功能码，此处有必要进行该协议的逆向工程
- 提取多款使用V2协议的PLC固件进行分析







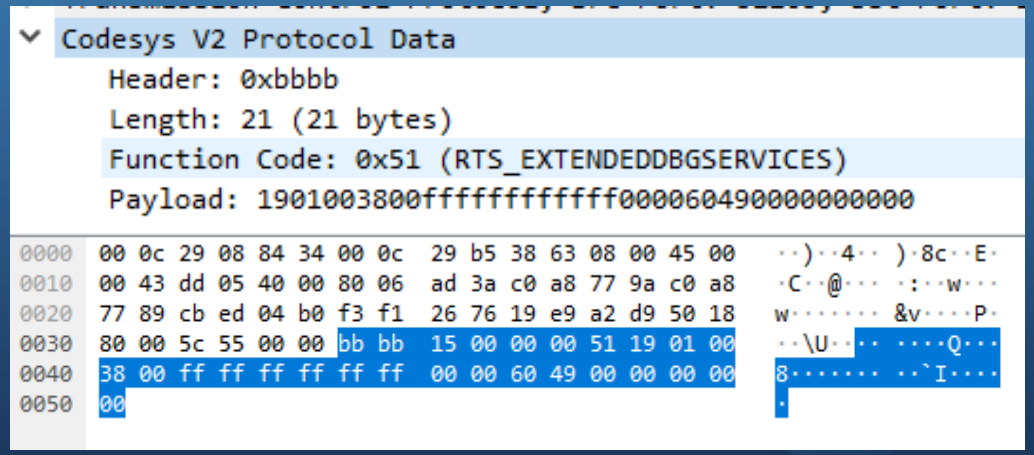
15 4.268973	192.168.119.154	192.168.119.137	CODESYSV2	61
17 4.272803	192.168.119.154	192.168.119.137	CODESYSV2	75
19 4.274824	192.168.119.154	192.168.119.137	CODESYSV2	61
22 5.628149	192.168.119.154	192.168.119.137	CODESYSV2	62
26 5.745911	192.168.119.154	192.168.119.137	CODESYSV2	389
29 5.786150	192.168.119.154	192.168.119.137	CODESYSV2	128

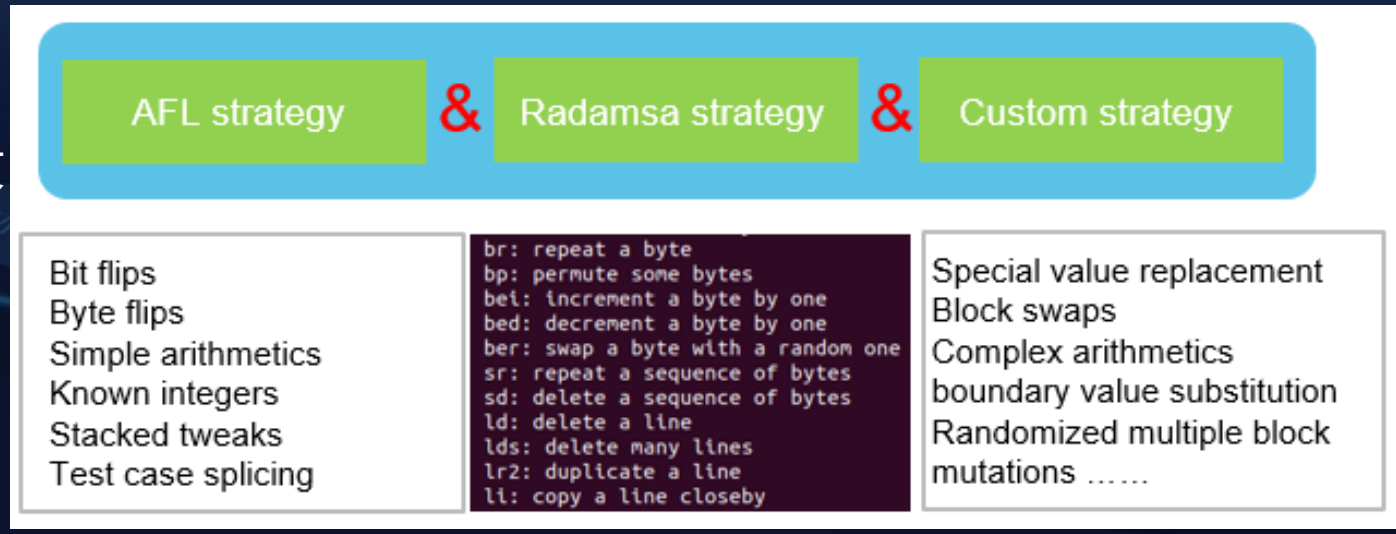
Fuzzer files

```
bbbb050000004251fbfa04
bbbb1600000043444f574e4c4f41442e5344420004000000046040000
bbbb1500000043444454641554c542e5052470004000000ce9a0200
bbbb1200000043424f4f542e5344420004000000046040000
bbbb1b00000043436f6e74726f6c4578706572742e657865000400000
bbbb1600000043313131312e776962752e696e690004000000b6070000
bbbb05000000429008e515
bbbb050000004230794412
bbbb1600000043444f574e4c4f41442e5344420004000000b6030000
bbbb1400000043736f757263652e6461740004000000a6752e00
bbbb05000000424627ee2c
bbbb0f000000010400000006000000000000000cd
bbbb0500000042145dda17
bbbb0500000042ac4aff15
bbbb1600000043444f574e4c4f41442e5344420004000007e030000
bbbb050000004288afa207
bbbb05000000425b9c4a18
bbbb150000000104000000060000000600000031323339383700cd
```

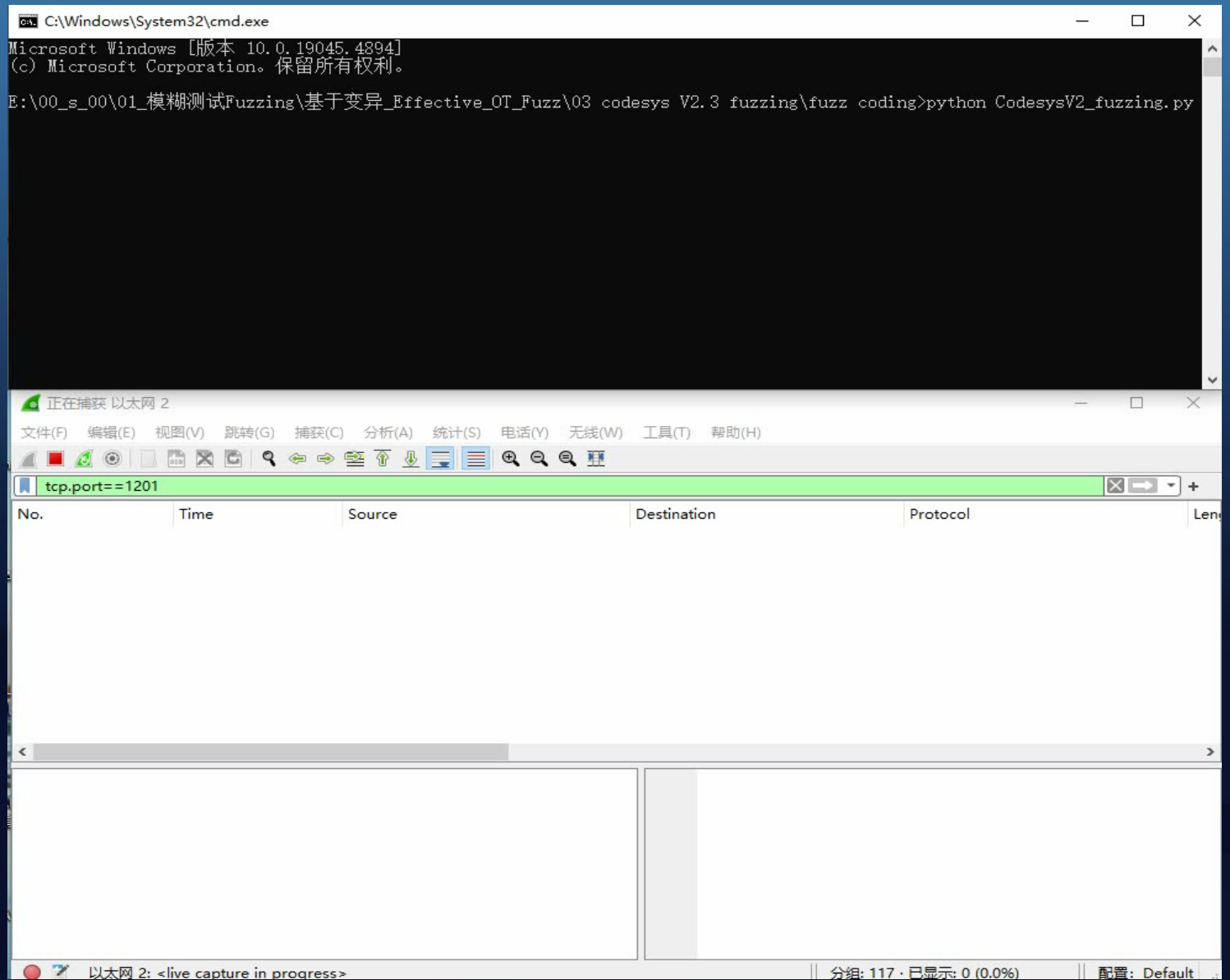
V2通讯协议Fuzzing测试-变异策略&异常监测

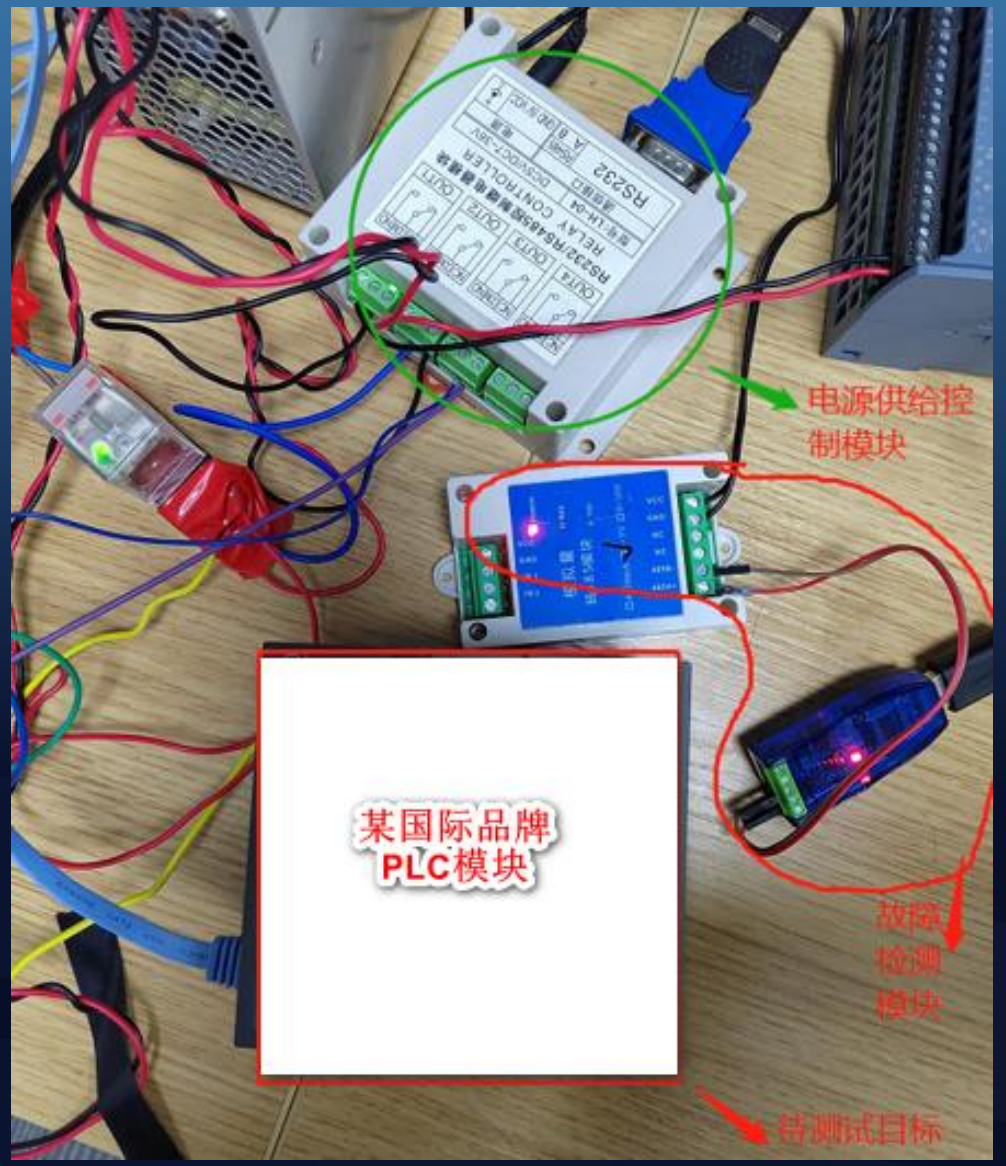
- 功能码列表纳入至变异策略中，每间隔一定周期在功能码列表中随机抽取
- 变异目标对象重点为payload域
- 变异算法可以有多种多样，原则是效率高出洞快。我们参考了AFL、Radamsa等变异算法，结合经验增加了定制化的变异算法
- 每次从语料库选取若干个样本送入变异引擎中
- 变异后的报文与login等必备的报文组合成序列报文发送至目标控制器
- 异常监测可以利用ping命令、利用RTS服务中正常功能比如rtsinfo报文、利用控制器的状态指示灯等等





V2通讯协议Fuzzing测试-DEMO





V2通讯协议Fuzzing测试-更高效的方案

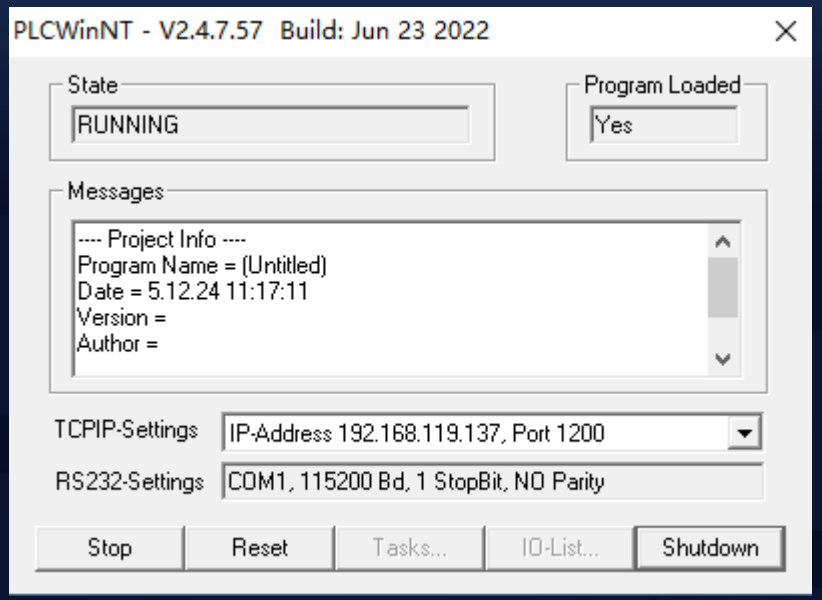
Fuzzing测试工业控制器的缺点

- 即便使用了自动化的方法，出现异常后断电重启需要花费时间，效率依旧不高；
- 发现漏洞后的复现与分析工作较难；
- 售卖的工业控制器一般无法调试，即便可以调试效率也不是很高，只有在利用环节上调试手段才有必要有价值
- .....

利用Codesys V2软件携带的**模拟仿真软件**搭建基于软件的fuzzing测试环境

```
377         case 2:
378           sub_14CBE0("SRV: RTS_LOGOUT\n");
379           sub_1CDD30(v5);
380           if ( !*(_DWORD *)(v5 + 72) )
381           {
382             sub_1CDC50(v5);
383             sub_1CDF60(a1, v5);
384             v5 = 0;
385           }
386           **(_WORD **)(a1 + 40) = 0;
387           goto LABEL_413;
388         case 3:
389           sub_14CBE0("SRV: RTS_START\n");
390           v23 = sub_191F50();
391           if ( (_BYTE)v23 )
392             goto LABEL_46;
393           **(_WORD **)(a1 + 40) = 50;
394           goto LABEL_413;
395         case 4:
396           sub_14CBE0("SRV: RTS_STOP\n");
397           sub_187010(v24);
398           **(_WORD **)(a1 + 40) = 0;
399           goto LABEL_413;
```

```
408     case 2:
409       sub_4026DA(aSrvRtsLogout, v51);
410       sub_4029A0(v230);
411       if ( !*(_DWORD *)(v230 + 72) )
412       {
413         sub_4021AD(v230);
414         sub_40123A(a1, v230);
415         v230 = 0;
416       }
417       sub_40262B();
418       **(_WORD **)(a1 + 40) = 0;
419       goto LABEL_508;
420     case 3:
421       sub_4026DA(aSrvRtsStart, v51);
422       if ( (unsigned __int8)sub_4023FB() )
423         goto LABEL_87;
424       **(_WORD **)(a1 + 40) = 50;
425       goto LABEL_508;
426     case 4:
427       sub_4026DA(aSrvRtsStop, v51);
428       sub_402BE9();
429       **(_WORD **)(a1 + 40) = 0;
430       goto LABEL_508;
```



V2通讯协议Fuzzing测试-自动化测试

构建测试框架 → 搭建自动化fuzzing测试框架，启动测试程序-监控程序crash-送入变异语料-异常后重新启动

PyDbgEng记录 → 利用PyDbgEng记录crash日志

样本获取变异处理 → 从语料库中随机抽取若干个样本送入变异引擎中进行变异

修复报文后发送至目标 → 将变异后的报文修复成目标格式后发送至待测试端口，并监测目标状态

异常后重新开启fuzzing → 持续监测模拟软件，崩溃后记录现场信息后即可重新开启新一轮fuzzing测试

PyDbgEng

Python Wrapper For Windows Debugging Engine

```
1C1F254E__2021-12-24 17-41-45.207673_EXPLOITABLE_WriteAV_0xe852b410_0x15b6a663.crash
1C4B06EF__2021-11-19 14-04-28.251422_UNKNOWN_WriteAVNearNull_0xb170deee_0x1eb13371.crash
1DC1F48E__2021-11-19 08-08-18.308314_UNKNOWN_TaintedDataControlsBranchSelection_0x92d898d4_0xbdf7a18a.crash
2C0A38D6__2021-12-26 21-22-31.297044_UNKNOWN_ReadAV_0x0ad21b75_0x6c893d86.crash
03CCC33E__2021-12-24 13-49-34.229253_PROBABLY_EXPLOITABLE_TaintedDataControlsWriteAddress_0xe852b410_0x5c009052...
3E96F07D__2021-12-26 21-02-28.885731_UNKNOWN_TaintedDataControlsBranchSelection_0x84e7cf9c_0xe176774e.crash
```

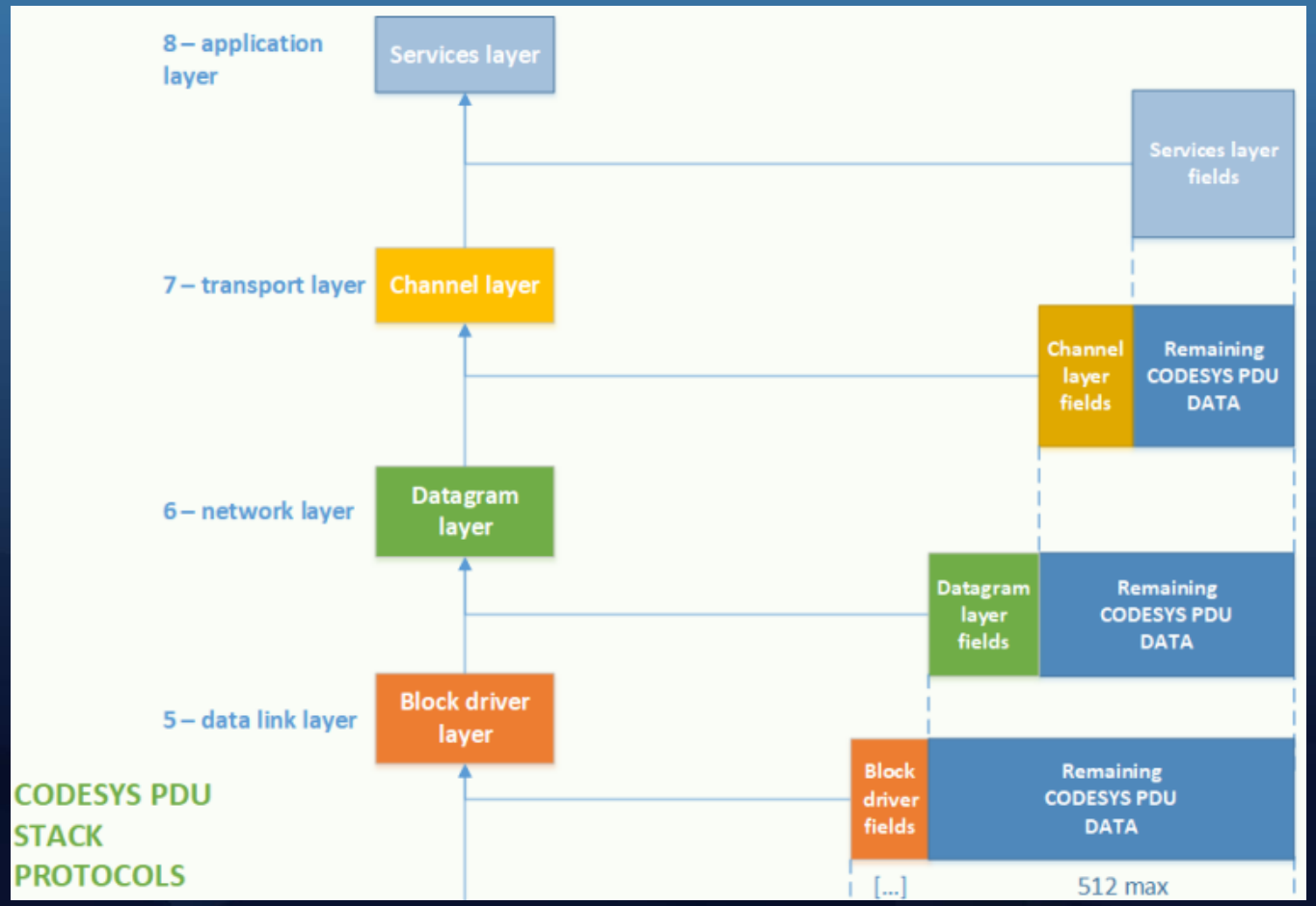
```
EXCEPTION_SUBTYPE:READ
FAULTING_INSTRUCTION:0050ec5c mov eax,dword ptr [esi+ecx*4-4]
BASIC_BLOCK_INSTRUCTION_COUNT:6
BASIC_BLOCK_INSTRUCTION:0050ec5c mov eax,dword ptr [esi+ecx*4-4]
BASIC_BLOCK_INSTRUCTION_TAINTED_INPUT_OPERAND:ecx
BASIC_BLOCK_INSTRUCTION_TAINTED_INPUT_OPERAND:esi
BASIC_BLOCK_INSTRUCTION:0050ec60 mov dword ptr [edi+ecx*4-4],eax
BASIC_BLOCK_INSTRUCTION_TAINTED_INPUT_OPERAND:eax
BASIC_BLOCK_INSTRUCTION_TAINTED_INPUT_OPERAND:ecx
BASIC_BLOCK_INSTRUCTION:0050ec64 lea eax,[ecx*4]
BASIC_BLOCK_INSTRUCTION_TAINTED_INPUT_OPERAND:ecx
BASIC_BLOCK_INSTRUCTION:0050ec6b add esi,eax
BASIC_BLOCK_INSTRUCTION_TAINTED_INPUT_OPERAND:eax
BASIC_BLOCK_INSTRUCTION_TAINTED_INPUT_OPERAND:esi
BASIC_BLOCK_INSTRUCTION:0050ec6d add edi,eax
BASIC_BLOCK_INSTRUCTION_TAINTED_INPUT_OPERAND:eax
BASIC_BLOCK_INSTRUCTION:0050ec6f jmp dword ptr image00000000_00400000+0x10ec78 (0050ec78)[edx*4]
```

Codesys V3内核介绍

Codesys V3 内核不仅继承了早期产品的优点，还引入了一系列创新特性，使其在性能、可靠性、安全性和易用性方面都达到了新的高度。

特性	CODESYS V2	CODESYS V3
组件化结构	单体系统	支持组件化模块化结构
工具可扩展性	非标准，需要额外插件	支持基于接口的扩展
项目加密	支持密码加密	支持密码和安全密钥
runtime组件化结构	不支持	支持
现场总线协议栈	支持有限的协议	支持更多的协议
用户管理	支持8个预设用户组	支持自定义用户权限
通讯协议	简单的报文结构	复杂的层级报文结构

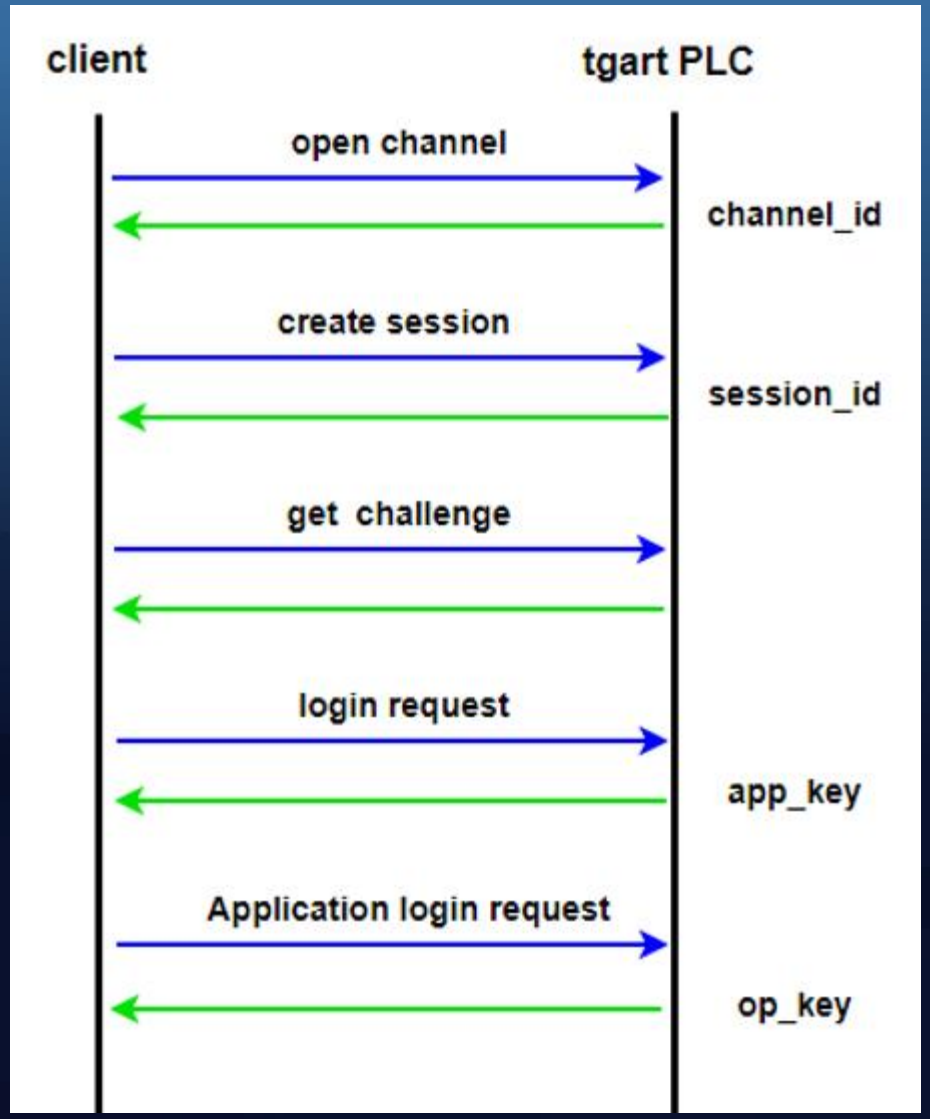
Codesys V3协议介绍



块驱动层 | 数据报文层 | 通道层 | 服务层

```
CodeSys V3 Protocol Data
  Block Driver Header
      magic: 0xe8170100
      length: 348
  Datagram Header
  Channel Header
  services Header
  Data
CodeSys V3 UDP Protocol Data
  Datagram Header
  Channel Header
      package_type: 0x01
      flags: 0x81
      channel_id: 0x8b70
      channel_session_key: 0x634dbb62
      Blk_id: 0x00000004
      Ack_id: 0x00000004
      Remaining_data_size: 304
      checksum: 0x2aa55bec
  services Header
      protocol_id: 0xcd55
      header_size: 16
      service_group: CmpDevice (0x0001)
      service_id: 0x0002
      session_id: 0x77a63b75
      data_size: 284
      add_data: 0x00000000
  Data
    Tag#0
        tagid: 34
        tagsize: 4
        tagdata: 02000000
```

Codesys V3协议授权流程



1)打开通道，目标控制器会给客户端返回channel_id,该值在后续的通信中会用到；

2)创建会话连接，目标控制器会给客户端返回session_id，同样该值在后续的请求报文中需要携带；

3)获取challenge值，用来进行授权验证相关算法使用；

4)发送授权后的登录请求，输入正确的用户名和密码才可进行敏感操作；

5)发送应用程序的登录请求，此时目标控制器会返回op_key，后续的敏感操作还需要携带该值；

6)至此才可以发送服务码群组中子功能码进行相应的操作与通讯；

构建重放攻击&模糊测试环境

Codesys V3协议模糊测试技术-样本制作

```
local service_group_desc={
    [0x18]="CmpAlarmManager",
    [0x02]="CmpApp",
    [0x12]="CmpAppBP",
    [0x13]="CmpAppForce",
    [0x1d]="CmpCodeMeter",
    [0x1f]="CmpCoreDump",
    [0x01]="CmpDevice",
    [0x08]="CmpFileTransfer",
    [0x09]="CmpIecVarAccess",
    [0x0b]="CmpIoMgr",
    [0x05]="CmpLog",
    [0x1b]="CmpMonitor",
    [0x22]="CmpOpenSSL",
    [0x06]="CmpSettings",
    [0x0f]="CmpTraceMgr",
    [0x0c]="CmpUserMgr",
    [0x04]="CmpVisuServer",
    [0x11]="PlcShell",
    [0x07]="SysEthernet",


}
```

```
int __cdecl AppServiceHandlerEx(HEADER_TAG *pHeaderTag, PROTOCOL_DATA_UNIT pduData, PROT
{
  int v6; // [sp+Ch] [bp-30h]
  RTS_RESULT Result; // [sp+24h] [bp-18h]

  Result = 0;
  if ( !pduSendBuffer )
    return 2;
  if ( pduData.ulCount < pHeaderTag->ulServiceLength )
    return 2;
  switch ( pHeaderTag->usService )
  {
    case 1u:
      AppSrvLogin(pHeaderTag, pduData, pduSendBuffer, bReplay, ulChannelId);
      goto LABEL_30;
    case 2u:
      AppSrvLogout(pHeaderTag, pduData, pduSendBuffer, bReplay);
      goto LABEL_30;
    case 3u:
      Result = AppSrvCreateApp(pHeaderTag, pduData, pduSendBuffer, bReplay);
      goto LABEL_30;
    case 4u:
      Result = AppSrvDeleteApp(pHeaderTag, pduData, pduSendBuffer, bReplay);
      goto LABEL_30;
    case 5u:
    case 6u:
    case 0x34u:
      Result = AppSrvDownload(pHeaderTag, pduData, pduSendBuffer, bReplay, ulChannelId);
      goto LABEL_30;
    case 0x10u:
    case 0x11u:
      AppSrvStartStop(pHeaderTag, pduData, pduSendBuffer, bReplay);
      goto LABEL_30;
    case 0x12u:
      Result = AppSrvReset(pHeaderTag, pduData, pduSendBuffer, bReplay);
      goto LABEL_30;
    case 0x14u:
      AppSrvReadStatus(pHeaderTag, pduData, pduSendBuffer, bReplay);
      goto LABEL_30;
    case 0x17u:
      AppSrvGetAreaOffset(pHeaderTag, pduData, pduSendBuffer, bReplay);
      goto LABEL_30;
    case 0x18u:
      AppSrvReadApplicationList(pHeaderTag, pduData, pduSendBuffer, bReplay);
      goto LABEL_30;
```

```
RTS_RESULT __cdecl TraceMgrServiceHandler(RTS_UI32 ulChannelId, HEADER_TAG *pH
{
  RTS_RESULT v5; // [sp+4h] [bp-30h]

  if ( pduData.ulCount < pHeaderTag->ulServiceLength )
    return 2;
  switch ( pHeaderTag->usService )
  {
    case 1u:
      TraceMgrSrvPacketReadList(pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
    case 2u:
      TraceMgrSrvPacketCreate(pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
    case 3u:
      TraceMgrSrvPacketDelete(pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
    case 4u:
      TraceMgrSrvPacketComplete(pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
    case 5u:
      TraceMgrSrvPacketOpen(pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
    case 6u:
      TraceMgrSrvPacketClose(pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
    case 7u:
      TraceMgrSrvPacketRead(pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
    case 8u:
      TraceMgrSrvPacketGetState(pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
    case 9u:
      TraceMgrSrvPacketGetConfig(pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
    case 0xAu:
      TraceMgrSrvPacketStart(pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
    case 0xBu:
      TraceMgrSrvPacketStop(pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
```

基于对通讯协议格式的详细分析，制作每个子服务的通讯报文样本，选择基于生成&变异的模糊测试技术

Codesys V3协议模糊测试技术-DEMO

```
RTS_RESULT TraceMgrInitServer()
{
  return Serve                    (0xFu, (PFServiceHandler)T                    );
}
```

```
print('fuzz_data is ----------------***')
print(b2a_hex(fuzz_msg))
tag_data_map[index[pos]]=fuzz_msg

tags=tag_data_map
tagblob=self.send_data(0x000f,0x0002,tags)
```

```
Data
  Tag#0
      tagid: 64
      tagsize: 4
      tagdata: 5051dec0
  Tag#1
      tagid: 65
      tagsize: 8
      tagdata: 434f444553595300
  Tag#2
      tagid: 66
```

```
RTS_RESULT __cdecl                          'RTS_UI32 ulChannelId, HEADER_TAG *pHeader
{
  RTS_RESULT v5; // [sp+4h] [bp-30h]

  if ( pduData.ulCount < pHeaderTag->u                 )
    return 2;
  switch ( pHeaderTag->u                 )
  {
    case 1u:
                                (pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
    case 2u:
                                (pHeaderTag, pduData, &pduSendBuffer);
      goto LABEL_24;
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.19045.4894]
(c) Microsoft Corporation。保留所有权利。

E:\>python Codesys_V3_fuzzing.py
```

SYS

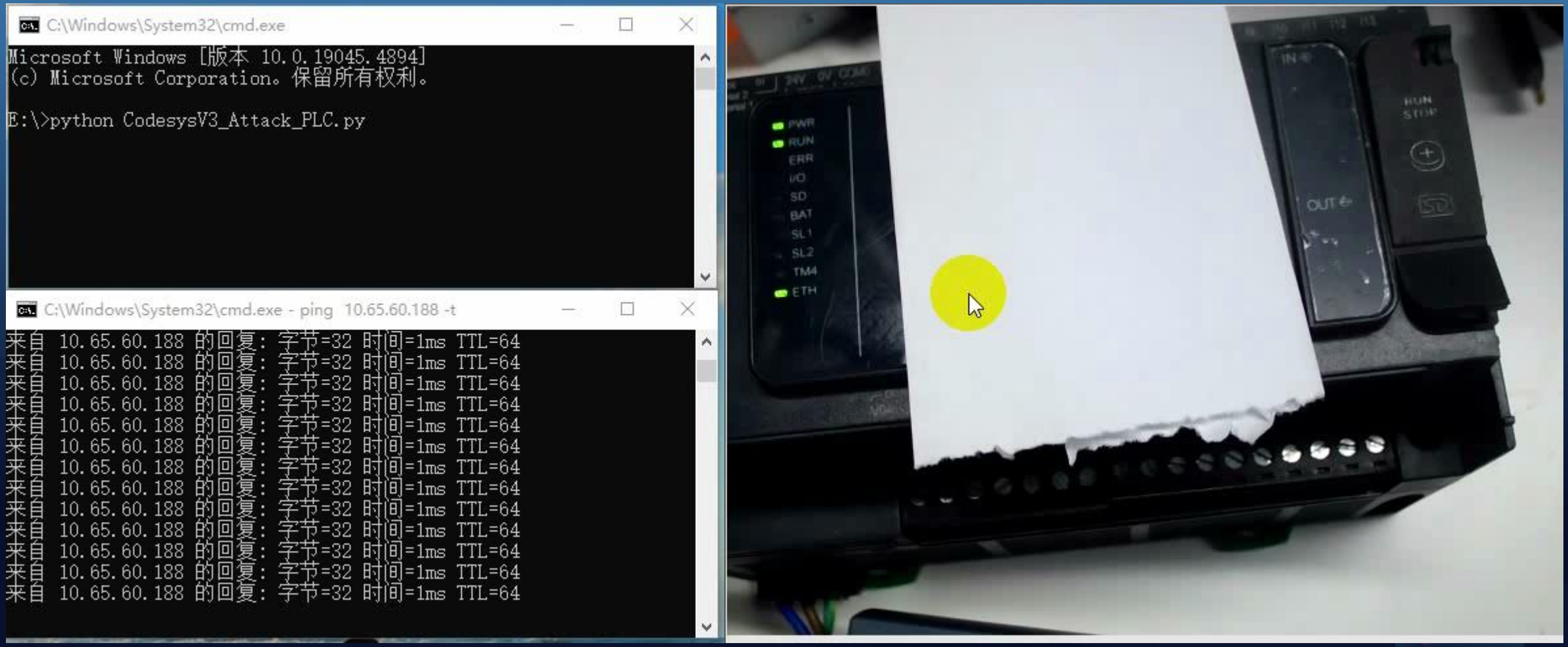
CODESYS Control Win V3 Version 3.5.20.30

内存 1 | 内存 2 | 内存 3 | 内存 4 | 内存 5 | 监视 1 | [x=] 局部变量 | 结构体

地址	十六进制	ASCII
77360000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....ÿÿ..

01B7E334	75B54599	返回到 kernelbase.
01B7E338	00000394	
01B7E33C	00000000	
01B7E340	00000000	

Codesys V3协议模糊测试技术-PLC攻击效果



利用CodesysV3模糊测试技术在winV3模拟器上发现的漏洞在工业现场的PLC上同样存在

Codesys V3协议高危漏洞分析

```
1  int TraceMgrInitServer_477270()
2  {
3    return [redacted] (int) [redacted]);
4  }
```

```
53        [redacted](v29, &a5, &a6);
54        goto LABEL_108;
55      case 2:
56        [redacted](a2, a3, a4, &a5);
57        goto LABEL_108;
```

```
65        [redacted]
66        [redacted](v30, (int *)&v18, &v19);
67        [redacted]((unsigned int)v26, (unsigned int)v18, v19);
68        break;
69        [redacted]
70        [redacted](v30, (int *)&v18, &v19);
71        [redacted](unsigned int)v27, (unsigned int)v18, v19);
72        break;
73        {
74        [redacted](v30, (int *)&v18, &v19);
75        [redacted]((unsigned int)v28, (unsigned int)v18, v19);
76        break;
```

```
memcpy_6D3060((unsigned int)v27, (unsigned int)v18, v19);
break;
```

```
EAX   07ABE6BC   "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBX   07ABFE40
ECX   00000000
EDX   000018AC
EBP   07ABF27C   "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ESP   07ABE664   &"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ESI   000018B0
EDI   03D7A11C

EIP   00A78720   codesyscontrolservice.00A78720

EFLAGS   00000201
ZF 0  PF 0  AF 0
OF 0  SF 0  DF 0
CF 1  TF 0  IF 1

默认 (stdcall)   5   解锁
1: [esp+4]  03D7A120 03D7A120 "AAAAAAAAAAAAAAAAAAAAAAAA
2: [esp+8]  000018AC 000018AC
3: [esp+C]  07ABED44 07ABED44 "AAAAAAAAAAAAAAAAAAAAAAAA
4: [esp+10] 07ABE69C 07ABE69C &"AAAAAAAAAAAAAAAAAAAAAAAA
5: [esp+14] 07ABE6A0 07ABE6A0
```

```
07ABE664  07ABE6BC  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...
07ABE668  03D7A120  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...
07ABE66C  000018AC
07ABE670  07ABED44  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...
07ABE674  07ABE69C  &"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...
07ABE678  07ABE6A0
07ABE67C  00017604
07ABE680  000018B0
07ABE684  03D7A108
07ABE688  00000000
07ABE68C  000018B0
07ABE690  00000013
07ABE694  00000000
07ABE698  00000000
07ABE69C  03D7A120  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...
07ABE6A0  000018AC
07ABE6A4  00000000
07ABE6A8  00000000
07ABE6AC  00000000
07ABE6B0  00000000
07ABE6B4  00000000
07ABE6B8  00000000
07ABE6BC  41414141
07ABE6C0  41414141
07ABE6C4  41414141
07ABE6C8  41414141
07ABE6CC  41414141
07ABE6D0  41414141
07ABE6D4  41414141
07ABE6D8  41414141
07ABE6DC  41414141
07ABE6E0  41414141
07ABE6E4  41414141
07ABE6E8  41414141
07ABE6EC  41414141
07ABE6F0  41414141
07ABE6F4  41414141
07ABE6F8  41414141
07ABE6FC  41414141
07ABE700  41414141
07ABE704  41414141
07ABE708  41414141
07ABE70C  41414141
07ABE710  41414141
07ABE714  41414141
07ABE718  41414141
07ABE71C  41414141
07ABE720  41414141
07ABE724  41414141
07ABE728  41414141
```

Codesys 内核研究成果

CVE-ID	Description	CVSS V3.1
CVE-2022-31805	The passwords between the communication clients and servers among the affected products are transmitted unprotected. This allows attackers to guess passwords if they are able to sniff the communication.	9.8
CVE-2022-31806	Password protection is not enabled by default and there is no information or prompt to enable password protection at login in case no password is set at the controller	9.8
CVE-2022-1965	An invalid crafted request is not properly processed by the error handling of the affected CODESYS products. As a result, the file referenced by the malicious request could be deleted if it exists on the controller.	6.5
CVE-2022-32136	A crafted request may cause an internal read access to an uninitialized pointer in the affected CODESYS products, resulting in a denial-of-service condition.	6.5
CVE-2022-32137	A crafted request may cause a heap-based buffer overflow in the affected CODESYS products, resulting in a denial-of-service condition or memory overwrite.	8.8
CVE-2022-32138	A crafted request with may cause an unexpected sign extension in the affected CODESYS products, resulting in a denial-of-service condition or memory overwrite.	8.8
CVE-2022-32139	A crafted request may cause an internal out-of-bounds read in the affected CODESYS products, resulting in a denial-of-service condition.	6.5
CVE-2022-32140	A crafted request may contain an incorrect data length for the associated structured data of the request. Since the affected CODESYS products do not handle the length correctly, this can lead to an internal buffer over-read causing a denial-of-service condition.	6.5
CVE-2022-32141	A crafted request with invalid offsets may cause an internal buffer over-read in the affected CODESYS products, resulting in a denial-of-service condition.	6.5
CVE-2022-32142	A crafted request with invalid offsets may cause an internal out-of-bounds read or write access in the affected CODESYS products, resulting in a denial-of-service condition or local memory overwrite.	8.1
CVE-2022-32143	The CODESYS V2 file download and upload function also allows read and potentially write access to internal files in the working directory, e.g. firmware files of the PLC, since no filtering is performed.	8.8

工控安全

Codesys V3协议漏洞挖掘方法

ic3blac4 2023-08-07 21:12:54 537060

12

原创 本文由 ic3blac4 创作，已纳入「FreeBuf原创奖励计划」，未授权禁止转载

发件人 CODESYS Security <security@codesys.com>

收件人

Hello,

meanwhile we have released the CODESYS security advisories 2022-11 and 2022-12 on our website:
https://www.codesys.com/security/security-reports.html.
The advisories can also be accessed by a direct link:

2022-11: https://customers.codesys.com/index.php?eID=dumpFile&t=f&f=17139&token=ec67d15a433b61c77154166c20c78036540cacb0&download=
2022-12: https://customers.codesys.com/index.php?eID=dumpFile&t=f&f=17140&token=6aa2c5c4a8b83b8b09936fefed5b0b11f9d2cc6c&download=

Thank you again for reporting the vulnerability following coordinated disclosure.

Mit freundlichen Grüßen / Best regards

Matthias Maier

CODESYS Group
We *software* Automation.

CODESYS

CODESYS Control Win V3 Version 3.5.19.0

6 Acknowledgments

These issues were reported by Gao Jian

CODESYS GmbH thanks for reporting following coordinated disclosure. This helps us to improve our products and to protect customers and users.

Codesys 供应链攻击流程及模型

- **扫描/搜索**

通过UDP端口1740、1743和TCP端口11740、11743、1201等具备Codesys明显特征的端口搜索

- **枚举**

利用rtsinfo或者识别发现服务获取Codesys内核版本信息，枚举出该版本具备的漏洞

- **扰乱**

根据攻击意图，执行简单的拒绝服务攻击即可扰乱生产过程

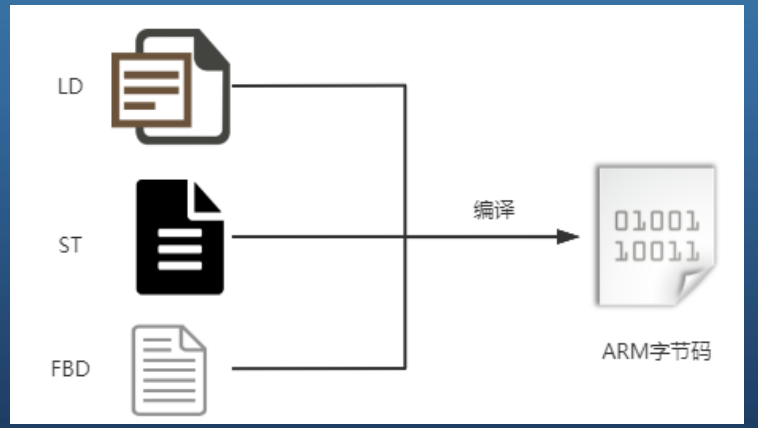
- **控制**

如若想长期驻留或者做更具备价值的攻击，需要深入研究工程逻辑，执行工程修改攻击、修改点值攻击或者更高级别的远程代码执行攻击，使工业控制器变为“武器系统”

攻击点	PLC Runtime	文件系统	管理系统	OS	Firmware
攻击目的及意图	1.读取工程文件 2.运行/停止工程逻辑 3.上载工程文件 4.下装工程文件 5.查看工程文件源码 6.改变工程文件代码 7.读写总线 8.读写寄存器值 9.强制插入程序逻辑或值	1.读写文件 2.读写PLC配置文件 3.读写PLC Runtime系统文件 4.删除文件 5.格式化文件系统 6.改变文件权限	1.重启PLC 2.恢复默认设置 3.停止PLC 4.配置I/O模块	1.改变系统调用 2.底层通信协议栈缺陷 3.代码执行	1.上传固件 2.下载固件 3.改变固件 4.篡改为恶意固件开放某些服务或者端口

Codesys 供应链攻击-远程代码执行

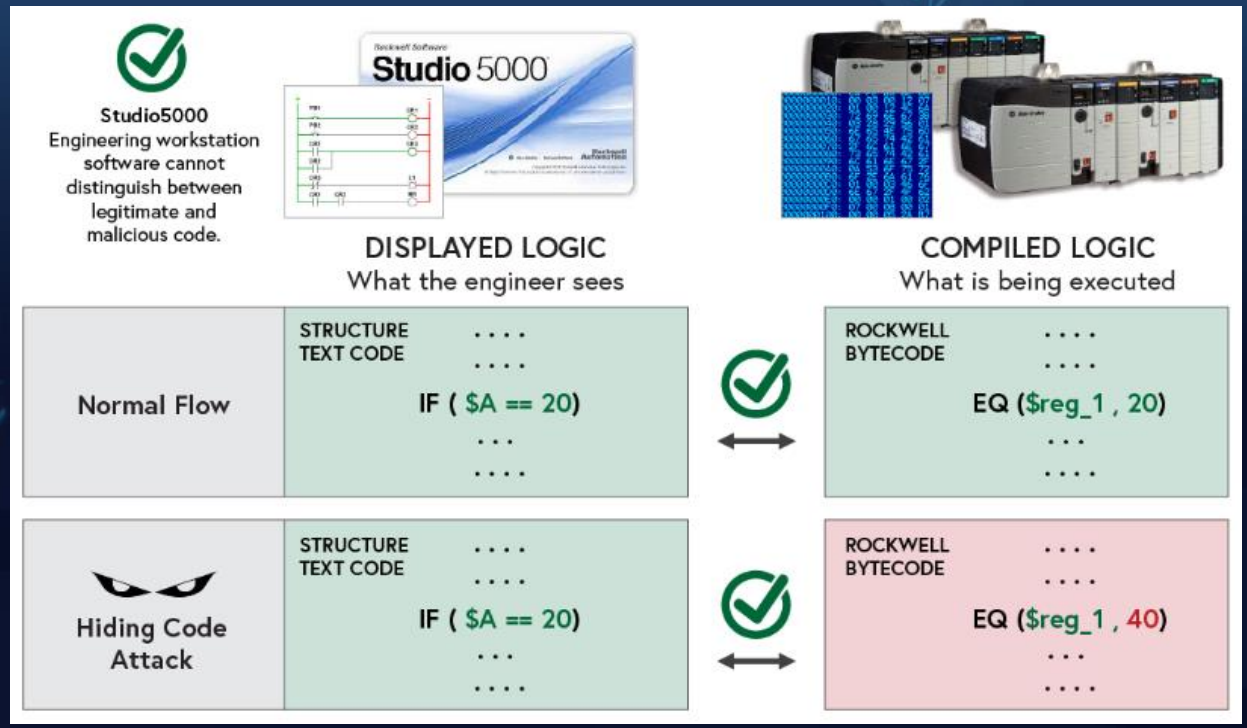
- 不论是IEC-61131-3标准中的何种编程语言，在编译型PLC的IDE中均会被编译成机器码，并通过通信协议下装至PLC的特定区域执行；
- 远程代码执行思路：设计恶意代码payload—插入至编译后的机器码片段中/劫持控制流至某个特定调用处—写入payload至PLC——运行PLC；
- 恶意代码功能：根据需求可以有很多功能，比如添加后门账号、添加socks代理、定期回传特定区域数据、开启远控通道等等；
- 本质的问题：编译后的**本地机器码**可以不受限制的在工控设备的处理器中执行；



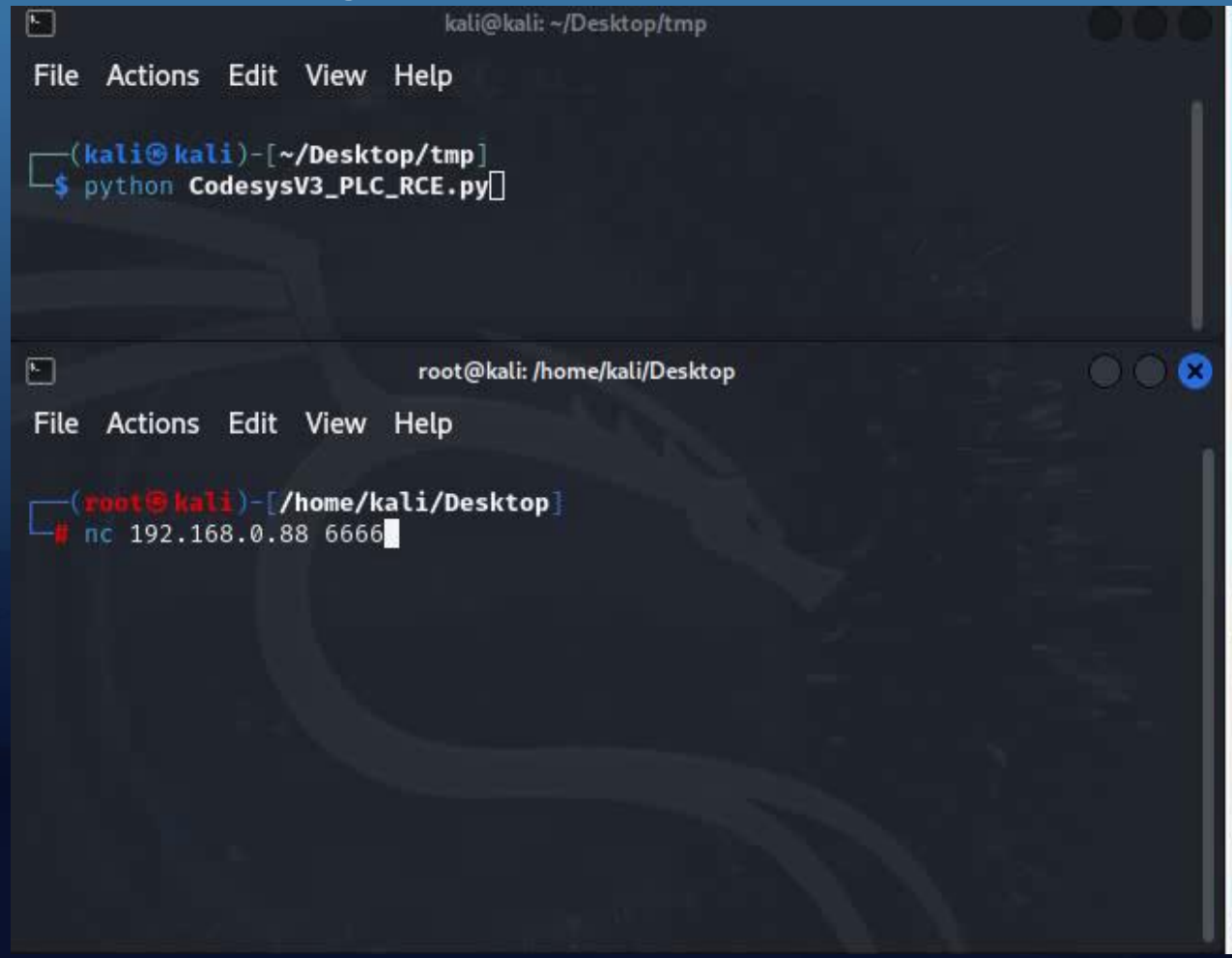
ARM机器码

OS Kernel

ARM处理器



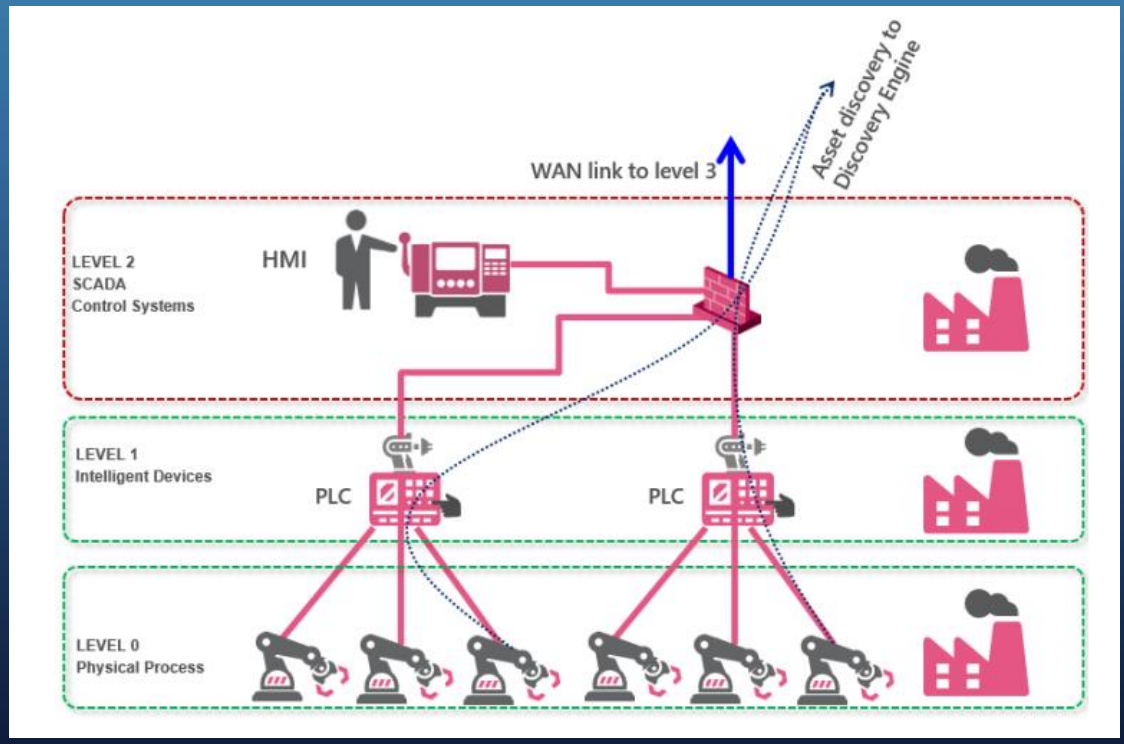
Codesys 供应链攻击-远程代码执行

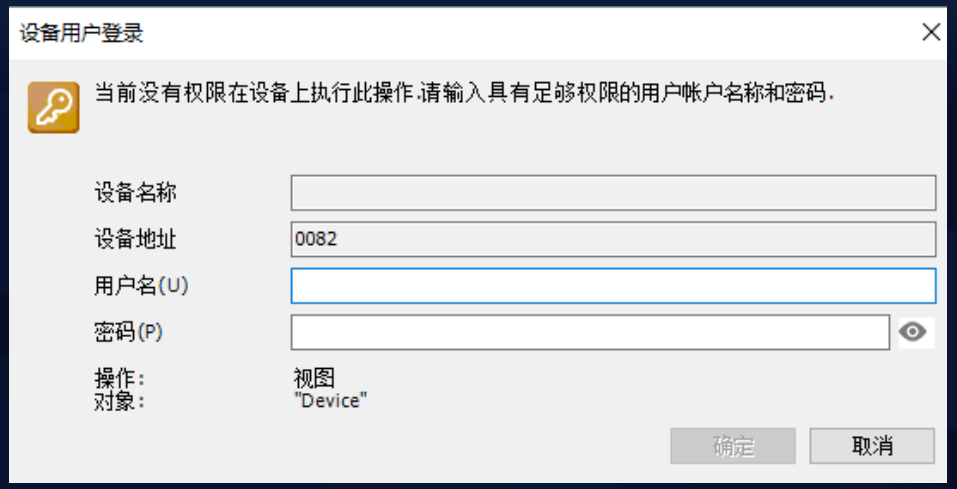


CodesysV3内核PLC：编写恶意payload机器码写入PLC——效果：创建监听端口，外部nc链接端口进行远控

基于现状的防护措施

1. 将受影响的产品放置于安全防护设备之后，做好纵深防御策略；
2. 当需要进行远程访问时，尽量采用安全的VPN网络，并且做好访问审计；
3. 关注受影响产品厂商的安全补丁，经过测试后升级受影响产品以使其免受威胁；
4. 尽量减少受影响设备的私有通信端口暴露，可根据业务场景选择关闭1201/1740/1741/1742/1743/11740/11741等端口；
5. 尽量给受影响的控制器设置用户名和密码保护，提升攻击难度；
6. 建议使用Codesys内核的工控厂商及时自查，并且积极修复，发布修复版本的固件；
7. 长远来看建议使用Codesys内核的工控厂商寻找更安全、更灵活的合作方式，同步更新内核安全补丁；





工业控制器安全设计建议-构建自主可控的生态

RTOS及实时
总线技术自主可控

编程软件环境
自主可控

关键电子元器件自主可控

PLC Runtime 技术自主可控

监控软件环境自主可控

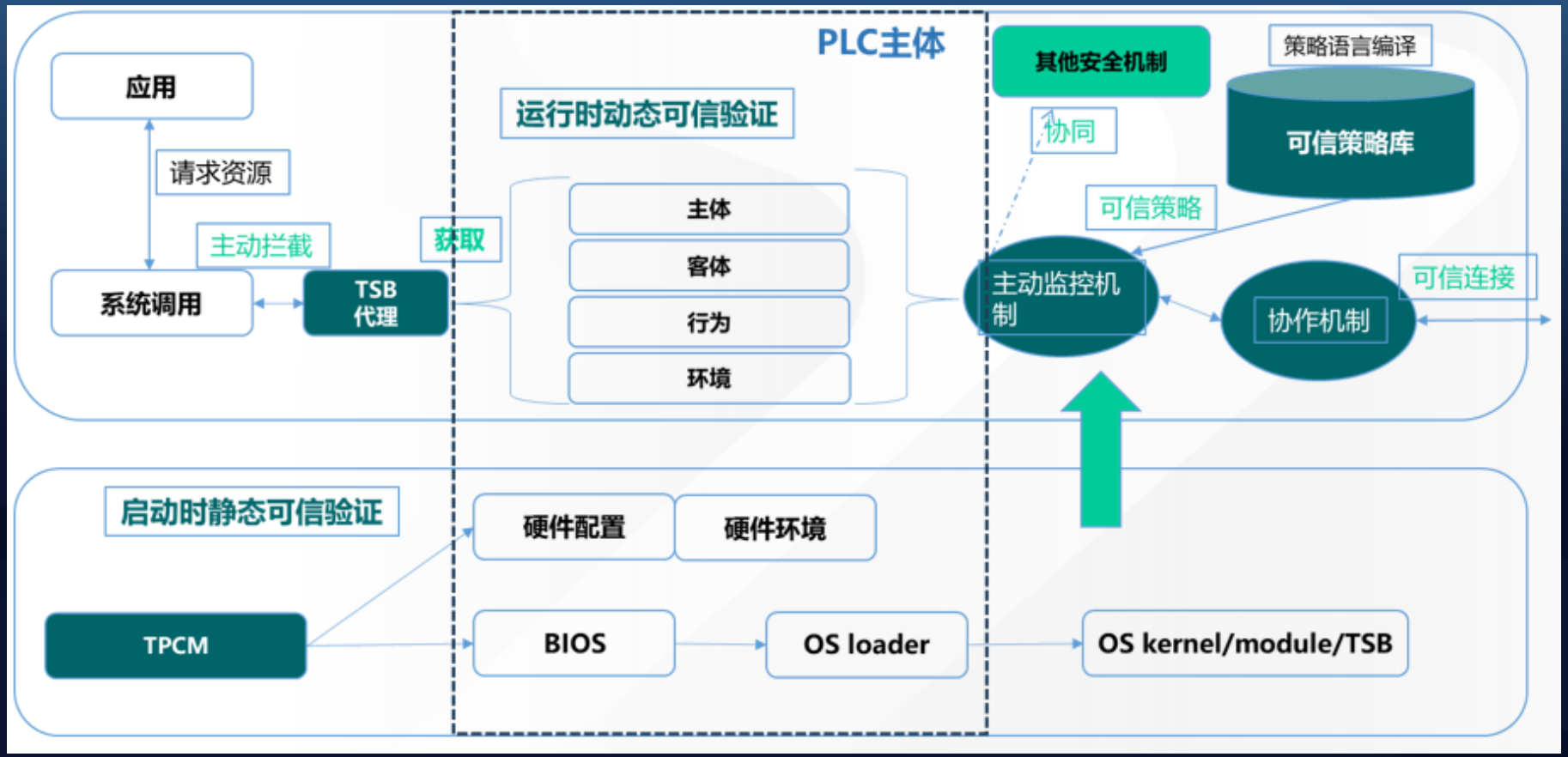
工业控制器安全设计建议-安全可信&国密技术应用

系统设计

以可信计算3.0技术为基础，从安全可信PLC系统设计出发，深入工控系统设计内部，融合安全和可信技术，从根本上解决工业控制系统本质的网络安全问题。

可信计算：

- 基于可信计算的双体系架构设计
- 基于可信计算的全生命周期可信度量技术
- 基于国密算法的通信加解密技术
- 基于TPCM的密钥管理技术



总结

- 从供应链攻击事件入手引出工控系统中存在的供应链风险
- 以广泛使用Codesys解决方案的工业控制器为研究对象展开研究
- 从易受攻击的通讯协议出发深入研究Codesys V2和Codesys V3私有协议漏洞挖掘及分析技术方法
- 展现漏洞对现场运行的工业控制器（PLC）的影响及通用类远程代码执行DEMO
- 就目前工业领域广泛使用Codesys解决方案的现状提出相应建议
- 长远来看，使用自主可控安全可信工业控制器可抵御大规模供应链攻击

感谢聆听