Stuxnet-in-a-Box: In-Field Emulation and Fuzzing of PLCs to Uncover the Next Zero-Day Threat in Industrial Control Systems

Dimitrios Tychalas **Michail Maniatakos**

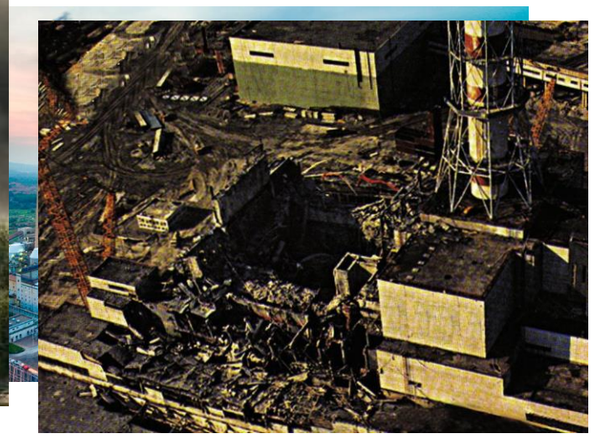
Industrial Control Systems - Intro

- Industrial Control Systems
  - What are they??
  - What do they do??

- ICS categories
  - Supervisory Control and Data Acquisition (SCADA)
  - Distributed Control Systems (DCS)
  - Programmable Logic Controllers (PLC)

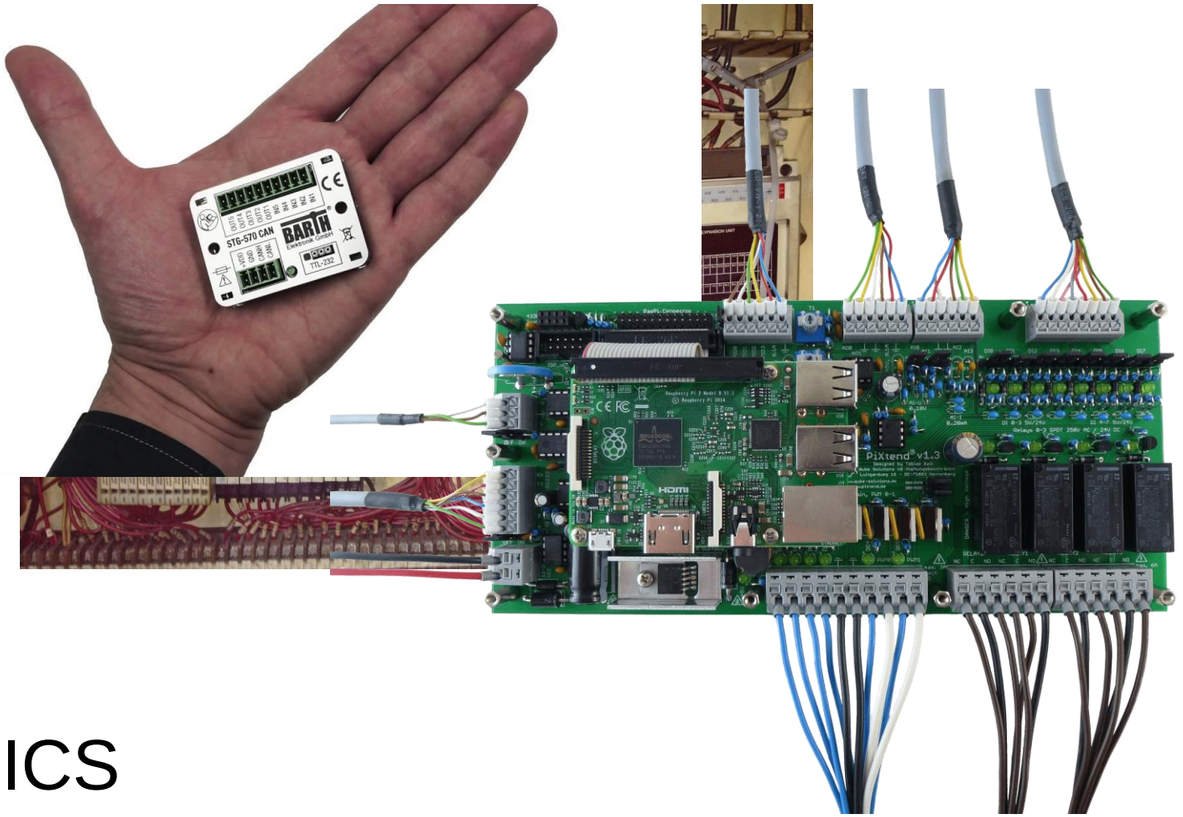
Industrial Control Systems - Criticality

- Deployment in critical infrastructures
  - Power grids
  - Various industrial plants
  - Nuclear facilities
- Paramount criticality
  - Serious disruptions
  - Loss of revenue
  - Loss of lives
- Safety depends on security



Industrial Control Systems - Evolution

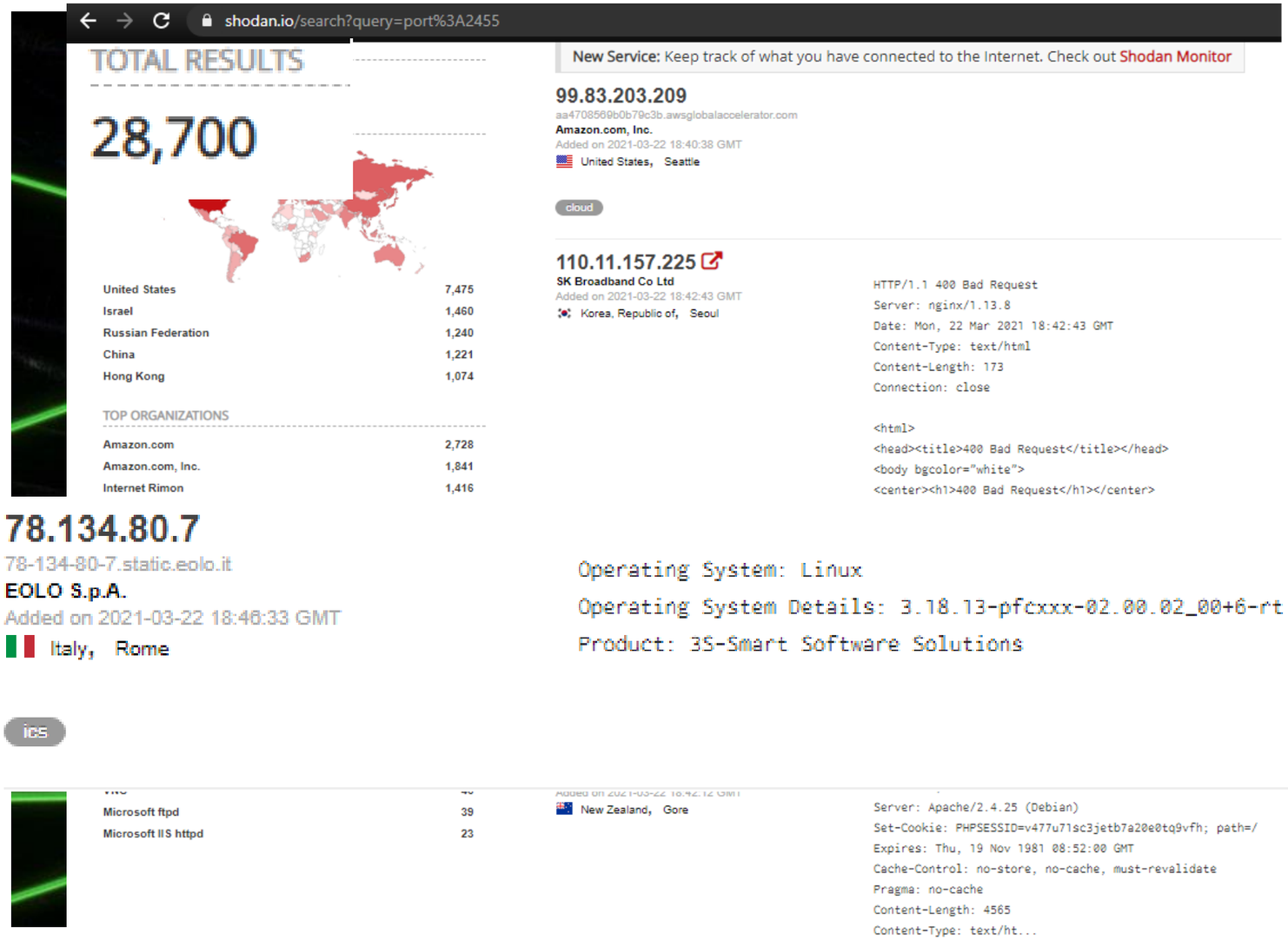
- Industry 4.0 and IIoT
  - Ditch this, get this, or this



- ICS evolve into typical computers
  - Generic third-party SoCs
  - General-purpose OS
  - Internet connection

- Typical computer threats jump over to ICS
  - Control flow hijacking, privilege escalation, network spoofing…
  - ICS can be indexed by a search engine (!) -> Shodan

Industrial Control Systems - Evolution



black hat ASIA 2021 Industrial Control Systems - Security

The State of Security
NEWS. TRENDS. INSIGHTS.

tripwire

ransomware a
dustrial cont
s ransomware is designed to tar
ition" in malware.

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

CRUDE BUT CONCERNING —

New ransomware doesn't just encrypt data. It also meddles with critical infrastructure

Ekans represents a "new and deeply concerning" evolution in malware targeting control

ITProPortal

News Best VPN Reviews Features Resources Security Ebook

ITProPortal is supported by its audience. When you purchase through links on our site, we ma

Home > Features

The world-changing 2015 cyberattack on Ukraine's power grid

By Roman Marshanski a month ago

In 2015, Ukrainian power plant operators fell victim to a sophisticated cyberattack.

Oct 7, 2010, 06:00am EDT

The Story Behind The Stuxnet Virus

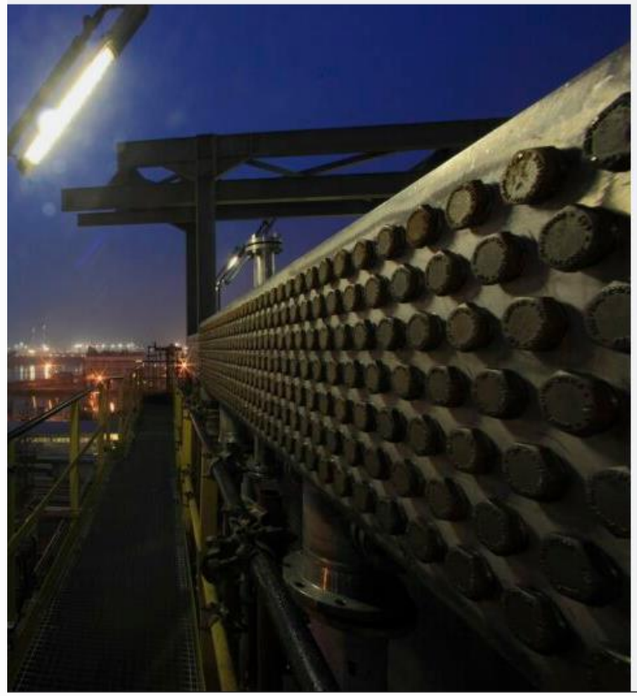
Bruce Schneier Former Contributor
I am the CTO of Resilient Systems, Inc.

This article is more than 10 years old.

Computer security experts are often surprised at which stories get picked up by the mainstream media. Sometimes it makes no sense. Why this particular data breach, vulnerability, or worm and not others? Sometimes it's obvious. In the case of Stuxnet, there's a great story.

As the story goes, the Stuxnet worm was designed and released by a government--the U.S. and Israel are the most common suspects--specifically to attack the Bushehr nuclear power plant in Iran. How could anyone not report that? It combines computer attacks, nuclear power, spy agencies and a country that's a pariah to much of the world. The only problem with the story is that it's almost entirely speculation.

Here's what we do know: Stuxnet is an Internet worm that infects Windows computers. It primarily spreads via USB sticks, which allows it to get into computers and networks not normally connected to the Internet. Once inside a network, it uses a variety of mechanisms to propagate to other machines within that network and gain privilege once it has infected those machines. These mechanisms include both known



#BHASIA @BLACKHATEVENTS

Industrial Control Systems – What's next?

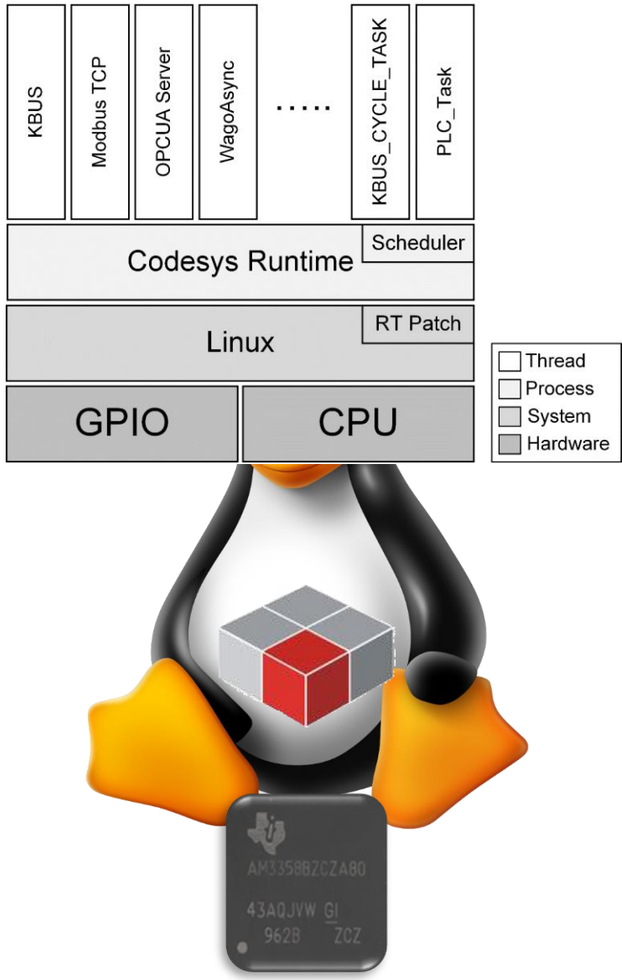
- How can one defend against an unknown threat?
  - Reactive solutions most commonly deployed
  - Can ICS security be proactive?

- Highly sophisticated attacks -> State actors
  - Can smaller teams develop a Stuxnet-level threat?

- Our answer
  - A tool collection to expose underlying vulnerabilities (IFFSET, ICSFuzz)
  - A demonstration of what threat an actor with limited resources can unleash (Stuxnet-in-a-box)

Codesys Runtime

- PLCing evolved
  - Monolithic firmware? Sure, if it's an ELF!
  - 3rd party popular SoC
  - "Firmware" hosted as an application in an open-source OS
  - Fast industry response -> Codesys holds ~25% market share



- Dedicated hardware replaced by software threads
  - HMI Connectivity -> Linux thread
  - PLC I/O (sensors/actuators) -> Linux thread
  - MODBUS communication -> Linux thread
  - Control logic functionality -> Linux thread (and some more)

Codesys Runtime

```
root@PFC100-4414DB:/etc/rc.d cat /proc/730/maps
00008000-00196000 r-xp 00000000 b3:02 19483      /usr/bin/codesys3
0019e000-001a1000 rw-p 0018e000 b3:02 19483      /usr/bin/codesys3
001a1000-0024d000 rw-p 00000000 00:00 0
017f0000-01a4f000 rw-p 00000000 00:00 0          [heap]
b4c77000-b4c78000 ---p 00000000 00:00 0
b4c78000-b4c97000 rwxp 00000000 00:00 0          [stack:3669]
b4c97000-b4c98000 ---p 00000000 00:00 0
b4c98000-b4cb7000 rwxp 00000000 00:00 0          [stack:3670]
b4cb7000-b4cb8000 ---p 00000000 00:00 0
b4cb8000-b4cd7000 rwxp 00000000 00:00 0          [stack:3671]
b4cd7000-b4cd8000 ---p 00000000 00:00 0
b4cd8000-b4cf7000 rwxp 00000000 00:00 0          [stack:3694]
b4cf7000-b4d17000 rw-s 00000000 00:15 3928       /dev/uio0
b4d17000-b4d18000 ---p 00000000 00:00 0
b4d18000-b4d37000 rwxp 00000000 00:00 0          [stack:3672]
```

```
S02_determine_hostname     S21_logforward          S91_virtua
S02_networking             S21_networking-finish   S92_rt-set
S04_auto_firmware_restore  S22_ifplugd             S97_serial
S05_logsystemstart         S22_ipwatchd            S98_runtir
S09_pureftpd               S24dnsmasq              S99_finali
S10_crond                  S48_mounthd2            S99_ssl_pc
S10_lighttpd               S54_keymap
S10syslog-ng               S60_mdmd
root@PFC100-4414DB:/etc/rc.d cat S98_runtime | grep daemon | grep codesys
    start-stop-daemon -K -qx /usr/bin/codesys3
root@PFC100-4414DB:/etc/rc.d
```

```
root@PFC100-4414DB:/etc/rc.d ps -AT | grep codesys
  730   730 ?        00:17:20 codesys3
root@PFC100-4414DB:/etc/rc.d ps -AT | grep 730
  730   730 ?        00:17:20 codesys3
  730   745 ?        00:00:00 DAL evt dispatc
  730   818 ?        00:00:24 KBUS dbus
  730   819 ?        00:00:05 com_DBUS_worker
  730   820 ?        00:00:00 wdbw_TermReg_cO
  730   837 ?        00:00:03 ModbusSlaveTCP
  730   838 ?        00:00:02 ModbusSlaveUDP
  730   872 ?        00:00:00 Oms_Watch_Threa
  730   876 ?        00:00:00 WagoIpcMsgCom
  730   878 ?        00:00:00 CAAEventTask
  730   879 ?        00:00:00 SchedProcessorL
  730   880 ?        00:00:00 SchedException
  730   881 ?        00:23:34 Schedule
  730   926 ?        00:00:47 BlkDrvTcp
  730   927 ?        00:01:14 BlkDrvUdp
  730   928 ?        00:01:12 OPCUAServer
  730  3562 ?        00:00:00 WagoAsyncRtHigh
  730  3669 ?        00:00:01 ProcessorLoadWa
  730  3670 ?        00:02:36 KBUS_CYCLE_TASK
  730  3671 ?        00:00:06 PLC_Task
  730  3672 ?        00:00:17 VISU_TASK
  730  3694 ?        00:00:30 WebServerTask
root@PFC100-4414DB:/etc/rc.d
```

Codesys Runtime - Security

- Codesys runtime
  - Great concept/implementation, still evolving
  - Prolific research target
  - CVE's ramping up throughout the years

	2014	2015	2016	2017	2018	2019	2020
# CVEs	3	3	-	2	6	14	21

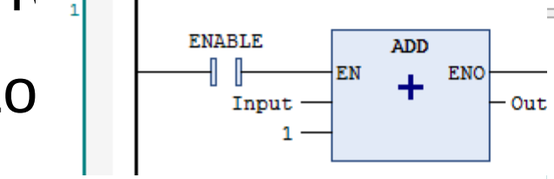
- Examples (2020)
  - CVE-2020-7052 : Uncontrolled memory allocation (6.5/10, medium)
  - CVE-2020-12068: Privilege escalation in visualization modules (6.5/10, medium)
  - CVE-2020-15806: Uncontrolled memory allocation (7.5/10, high)
  - CVE-2020-6081: Exploitable code execution in PLC program loading (8.8/10, critical)
  - CVE-2020-10245: Remote code execution through heap overflow in web server (9.8/10, critical)

PLC Control Application - Intro

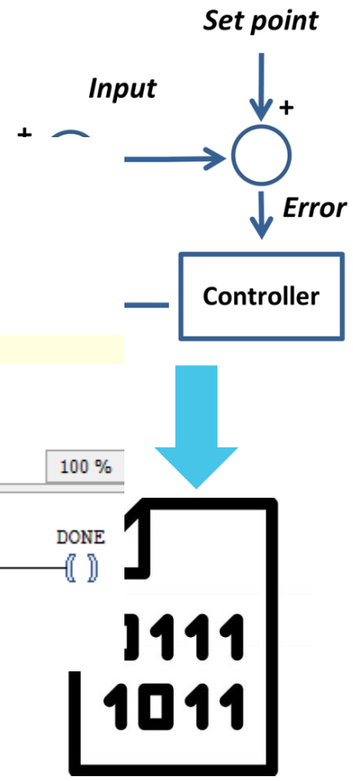
- Control logic made into an application
  - Cc
- Built
  - Pr
  - Lo

```
PROGRAM PLC_PRG
VAR
    Input       AT  %IW1    :WORD; (
    Output      AT  %QW1    :WORD; (
    ENABLE      AT  %IX1.0  :BOOL; (
    DONE        AT  %QX0.0  :BOOL; (
END_VAR
```

```
PROGRAM PLC_PRG
VAR
    Input       AT  %IW1    :WORD; (*Analog Input 0-10V*)
    Output      AT  %QW1    :WORD; (*Analog Output 0-10V*)
    ENABLE      AT  %IX1.0  :BOOL; (*Digital Input (0 or 5V) for Enable button*)
    DONE        AT  %QX0.0  :BOOL; (*DigitaL Output (0 or 5V) for LED indication*)
END_VAR
```




```
IF ENABLE = TRUE THEN
    Output:= Input +1;
    DONE:= TRUE;
ELSE
    DONE:= FALSE;
END_IF
```



- Multiple languages standardized by IEC (IEC-61131)
  - Ladder Logic (simplest, most ubiquitous)
  - Function Block Diagram (moderately complex, like LL but beefier)
  - Structured Text (complex, based on Pascal)

PLC Control Application – Threats

- Compilation
  - Variable input
  - Input used for indexing
  - Compiler cannot predict threat
  - Out-of-bounds read/write
- Third party libraries
  - C-like libs developed
  - Low-level memory management
  - May lack bounds checks
  - Buffer overflow

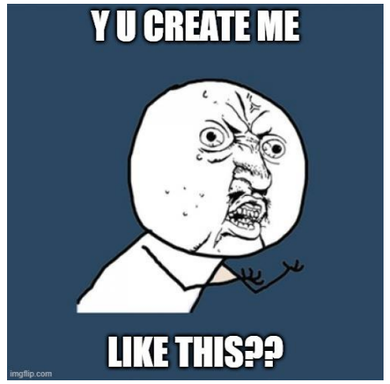
```
PROGRAM PLC_PRG
VAR
    snoop_array: ARRAY[1..10] OF UDINT;
    input1 AT %IW1:WORD;
    output1 AT %QW1:WORD;
```

	C/C++	Codesys 2.x, 3.x		
	Function Name	Function Name	Bounds Check	Crash
String Operations	strcpy()	SysStrCpy()	✗	✗
	strcat()	Concaat()	✓	✗
Memory Operations	memcpy()	SysMemCpy()	✗	✓
	memset()	SysMemSet()	✗	✓
	memmove()	SysMemMove()	✗	✓
	memcmp()	SysMemCmp()	✗	✗

```
x := 16#DEAFBEEF;
snoop_array[WORD_TO_INT(input1)+16#DEADCAFE] := x;
```

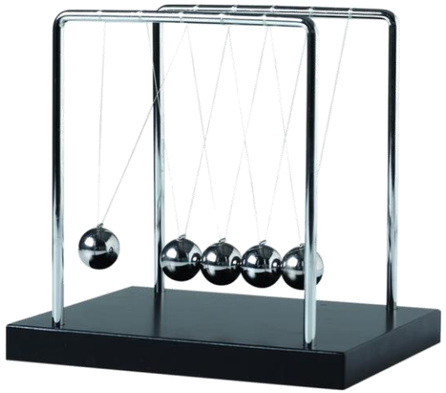
PLC Security – What to do?

- Challenge 1
  - Compiler has inherent weaknesses
  - Untested 3rd party libraries



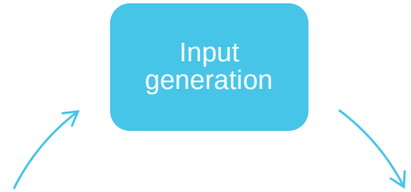
totally_legit.dll

- Challenge 2
  - PLC are expensive
  - PLC are slow
  - PLC must be perpetually engaged



- Solution
  - Challenge 1: Fuzzing
  - Challenge 2 : Emulation

- Fuzzing
  - Input testing with corner cases (the fuzzy outliers)
  - Input mutation
  - Binary instrumentation
- Emulation
  - QEMU
- What about here?
  - Many, many targets!
  - System binaries
  - Codesys runtime
  - PLC application



QEMU

110010110

110110110

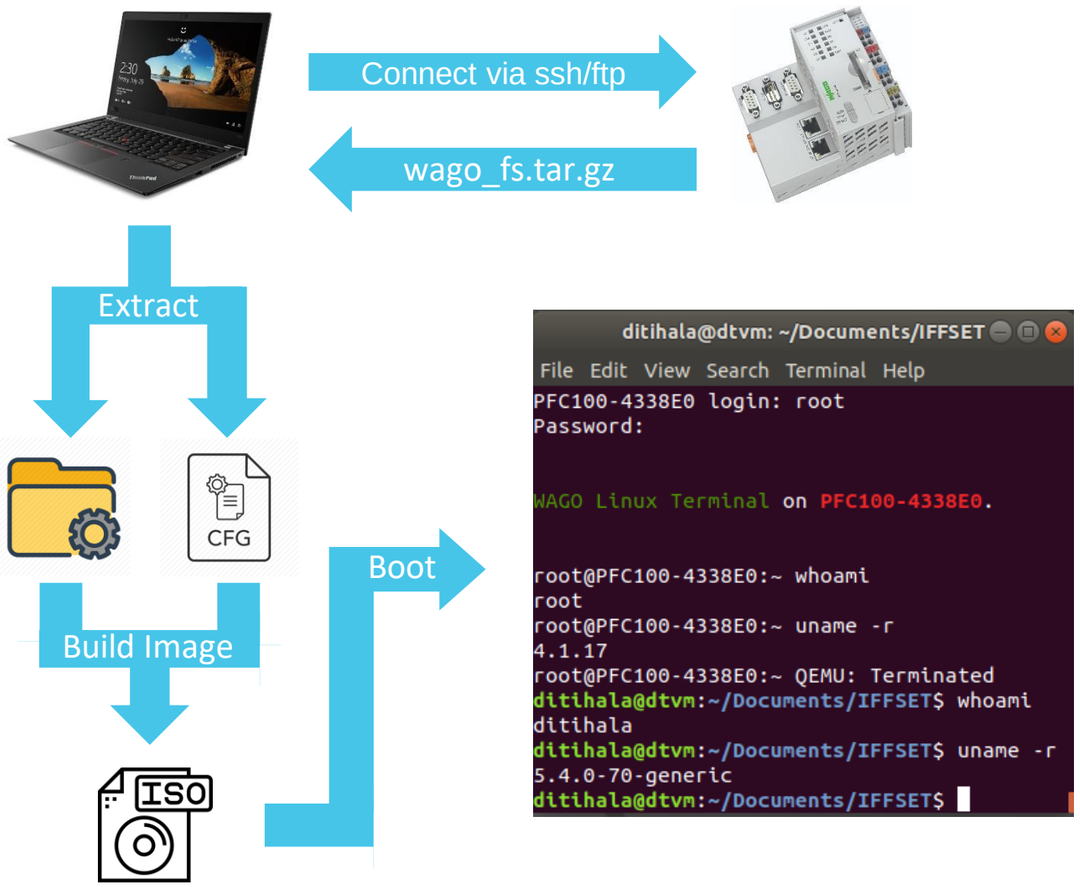
110010110

1100010010100101

```
c000: #compute -0x8(%rbp) and copy it to a buffer
c008: 48 89 7d f8   mov  %rdi, -0x8(%rbp)
c00c: #compute (%rsp) and copy it to a buffer
c014: 5e            pop  %rsp
c015: 75 f8         jne  0xc004
c017: c9            leaveq
c018: c3            retq
```

Fuzzing/Emulation System (IFFSET)

- Connect to PLC (SSH/FTP)
- Extract necessary resources
  - Kernel configuration
  - File system
- Build bootable QEMU disk image
  - Mount image for processing
  - Modify init scripts
  - Add fuzzer
- Boot system
  - Login through default credentials
  - AFL to fuzz system binaries



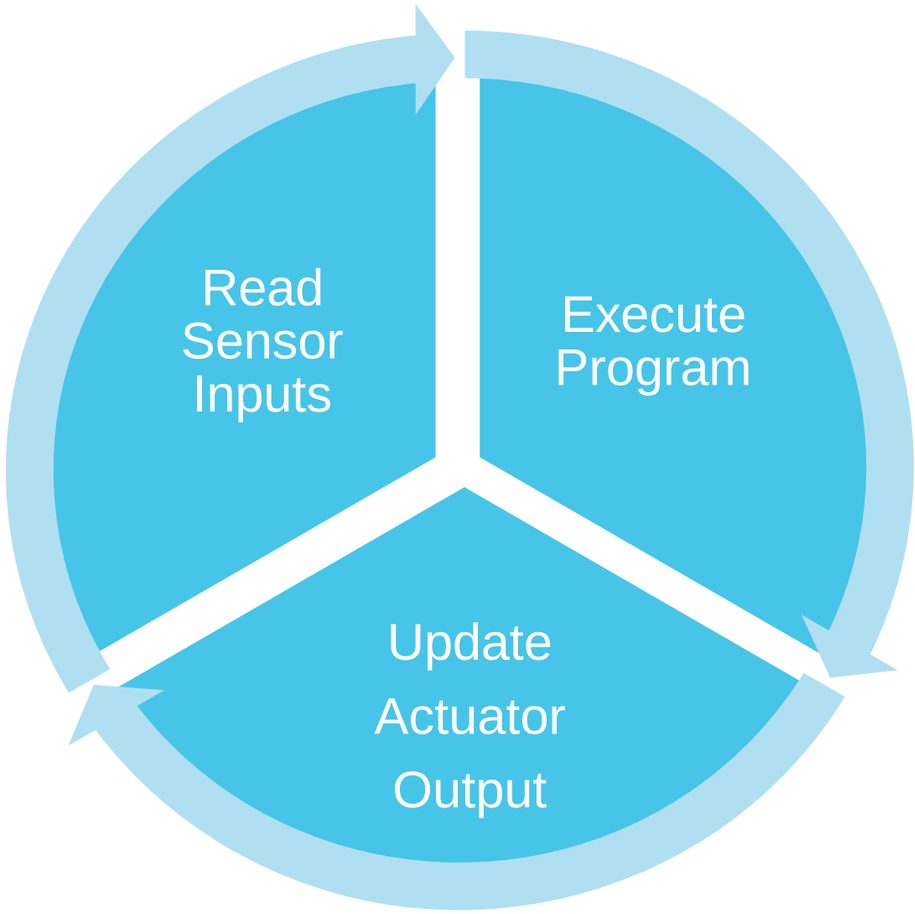
Fuzzing – IFFSET (Demo)

Fuzzing – PLC application (ICSFuzz)

- Binary format not suitable for fuzzing
- Binary must exist in Codesys context
- Fuzzing broken down:
  - Execution control
  - Input control

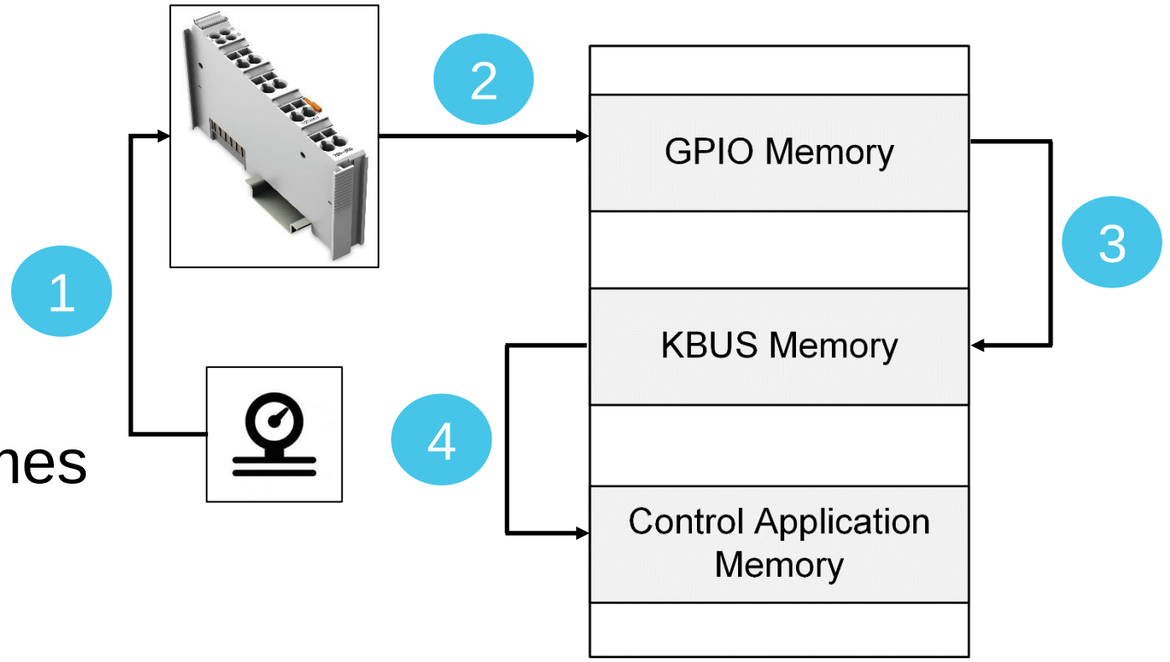
- Execution control
  - PLC applications run on a cycle
  - Leverage this cycle for continuous execution

- Input control
  - Reverse engineer input delivery to the PLC



Fuzzing – PLC application (ICSFuzz)

- Input control (contd.)
  - Follow the input flow
  - Choose the most controllable point
  - Force new values in-memory
  - Mutate values based on established schemes



- Instrumentation
  - No source code – No party
  - Scan for opportunities
  - Why not NOPs?
  - Replace NOPs with controllable code

```
STR   r5 ,[sp ,#0x0]
STR   r4 ,[sp ,#0x8]
STR   r6 ,[sp ,#0xc]
LDR   r11 ,=0xB4F22A8Ch
LDR   r6 ,[r11 ,#0x0 ]
CPY   r0 ,sp
STR   r10 ,[sp ,#0x38 ]!
LDR   r10 ,=0xCDE1F2CDh
STR   r10 ,[sp ,#0x24 ]!
MOV   r10 ,#0x0
MOV   r0 ,r0
MOV   lr ,pc
```

```
STR   r5 ,[sp ,#0x0]
STR   r4 ,[sp ,#0x8]
STR   r6 ,[sp ,#0xc]
LDR   r11 ,=0xB4F22A8Ch
LDR   r6 ,[r11 ,#0x0 ]
CPY   r0 ,sp
STR   r10 ,[sp ,#0x38 ]!
LDR   r10 ,=0xCDE1F2CDh
STR   r10 ,[sp ,#0x24 ]!
MOV   r10 ,#0x0
STR   pc , [r0 ,#0xDEADBEEF]
MOV   lr ,pc
```

Fuzzing – ICSFuzz (demo)

Stuxnet-in-a-box

- Combine all information uncovered through assessment
  1. Reverse-engineer PLC application function
  2. Manipulate functionality
  3. Fuzz for vulnerabilities
  4. Exploit vulnerability
  5. Synthesize new attack vector

- A novel attack methodology that
  - Manipulates a PLC application functionality
  - Inserts a kernel rootkit to spoof correct function

Stuxnet-in-a-box (demo)

Mitigations

- System
  - Updates based on assessment
- Codesys platform
  - Open-up for researchers
  - Control application loading redesign
- Application
  - Hot patching
  - Compiler run-time awareness
  - 3rd party library auditing
  - NX-bit enforcement
  - Coding limitation?

Standardization issues

- Limited software-based standardization in ICS
  - Network communication
  - Language structure
  - Firmware structure?
  - Code production?
- Industry leaders stick to in-house solutions
  - In-house SoC
  - Self-developed firmware
- Codesys a step to the right direction
  - Multi-platform compatibility
  - Centralized security assessment efforts

Modbus

IEC 61131-3

A-B QUALITY Allen-Bradley

SIEMENS

Closing remarks – Main takeaways

- Current PLC programming practices introduce vulnerabilities as easily exploitable as buffer overflows that can hijack a whole industrial process.
- Modern Linux-based soft PLC platforms are exploitable from a simple DoS to a full-blown take-over of an industrial setting or a critical infrastructure.
- The lack of standardization practices across the various ICS vendors limits the potential for a thorough security assessment of industrial devices, such as PLC, which opens the way for more zero-day vulnerabilities.

End of presentation

Thank you!

Questions?