

Ghost in the PLC Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack

ALI ABBASI SYSSEC GROUP, RUHR UNIVERSITY BOCHUM, GERMANY & SCS GROUP UNIVERSITY OF TWENTE, NETHERLANDS

MAJID HASHEMI QUARKSLAB, PARIS

Who we are

 Ali Abbasi, visiting researcher at chair of system security of Ruhr University Bochum and PhD student at Distributed and Embedded Systems Security Group, University of Twente, The Netherlands.

 Majid Hashemi, R&D researcher at Quarkslab, involved in this research as an independent researcher.

Plan of talk

- Background on existing attacks and defenses for embedded systems
- Applicable Defenses for PLCs
- Background on Pin Control
- The Problem
- Rootkit variant
- Non-rootkit variant
- Demo
- Discussions

What this talk is not about?

- The talk is trying to uncover existing design flaw in PLCs.
- The attack can be used in future by attackers.
- Today, there are far easier attacking techniques but it is trivial to defeat them.
- We are not unveiling fully functional rootkit for PLCs.
- NO exploitation techniques, no Oday leak

black hat europe 2016

• We are not going to mention any vendor name.

Electrical





• Water









Military

Steuerten Hacker Raketenstationen der Bundeswehr?

Hacker haben womöglich das Flugabwehrsystem Patriot geknackt: In der Türkei stationierte Raketenstationen der Bundeswehr hätten "unerklärliche" Befehle ausgeführt, berichtet eine Fachpublikation.



DIE WELT APPS



 Elements of infrastructure that, if lost, could pose a significant threat to needed supplies, services, and communication¹.

1. Church, R. L., Scaparra, M. P., & Middleton, R. S. (2004). Identifying critical infrastructure: the median and covering facility interdiction problems.



Who are the attackers

- Mostly State Sponsored
- Use Odays
- Use Sophisticated evasion techniques
- Best Example? Stux...







How to attack critical infrastructures?

- Get into the network without being detected.
 - Defeating Emulation Based Network Intrusion Detection Systems
 - APTs Way: Evading Your EBNIDS, Black Hat Europe, 2014.
- Manipulate the PLCs
 - The PLC logic
- Manipulate the process
 - Damn vulnerable chemical processes²



2. Krotofil, Marina, et al. "Vulnerabilities of cyber-physical systems to stale data—Determining the optimal time to launch attacks." International Journal of Critical Infrastructure Protection 7.4 (2014): 213-232.



What is a PLC?

An Embedded System with RTOS running logic.



12

What is Logic

Logic is a program PLC executes



How PLC executes the Logic

- Logic
- Physical I/O









Existing Attacks and Defenses for Embedded Systems Applicable to the PLCs

Current attacks against embedded systems

- Firmware modification attacks
 - Attacker upload new firmware to the PLC
- Configuration manipulation attacks
 - Attacker modify the logic
- Control Flow attacks
 - Attacker find a buffer overflow or RCE in the PLC
- Authentication bypass

- Attacker find a backdoor password in the PLC.
- Hooking functions for ICS malwares (e.g. Stuxnet)

Current defenses for embedded systems

- Attestation
 - memory attestation
- Firmware integrity verification
 - Verify the integrity of firmware before its being uploaded
- Hook detection
 - Code hooking detection
 - Detect code hooking
 - Data hooking detection
 - Detect data hooking

Applicable Defenses for PLCs

- Designed for embedded devices running modern OS.
- No hardware modifications.
- Limited CPU overhead.
- No virtualization support required.



Non-trivial System-level protection for PLCs

- Trivial Defenses:
 - Logic Checksum
 - Firmware integrity verification
- Non-trivial software-based HIDS applicable to PLCs
 - Doppelganger (Symbiote Defense): an implementation for software symbiotes for embedded devices
 - Autoscopy JR: A host based intrusion detection which is designed to detect kernel rootkits for embedded control systems



How Doppelganger Works

 Scan the firmware of the device for live code regions and insert symbiotes randomly.



How Autoscopy Jr works

- Tries to Detects function hooking by learning
- Verifies the destination function address and returns with the values and addresses in TLL (Trusted Location List)



Function hooking

Code Hook





System Call Table



22

Solution Found?

Looks like if we deploy both
 Doppelganger and Autoscopy Jr,
 the problem will be solved.

 Autoscopy Jr detects code hooking and doppelganger verify static parts and detect data hooking.



Dynamic Memory (Heap)

Doppelganger can not verify dynamic memory.





Dynamic Memory (Heap)

Doppelganger can not verify dynamic memory.





Limitations

- Static Referencing
 - Both Autoscopy Jr and Doppelganger use static references, similar to signature-based approaches.
- Dynamic Memory
 - Doppelganger only verify static memory
- Function Hooking Definition
 - The function hooking definition of Autoscopy Jr is incomplete







Chapter Two

Background on Pin Control

Background on Pin Control

Pin Control Subsystem

- Pin Configuration
- Pin Multiplexing





BCM2835 and available I/O Functions





29

Pin Configuration

- Input Pin
 - readable but not writeable

 Output 	Pin
----------------------------	-----

readable and writeable

dwln.24 T#5S T#5S	TRUE T#5s T#5s	13 IT	NABLE MELO' MEHIG	blk BLINK W	OUT		dwOut.22	
Parameter	Туре	Current	Pr	Value	Default	Unit	Description	T
👘 🖗 GPIO4	Enumeration of BYTE	not used		not used	not used		configuration of GPIO4	
🔷 < 🖗 🖗	Enumeration of BYTE	not used		not used	not used		configuration of GPIO17	
🔷 < 🖗 🖗	Enumeration of BYTE	not used		not used	not used		configuration of GPIO18	
🔷 🏟 GPIO22	Enumeration of BYTE	Output		Output	not used		configuration of GPIO22	_
🔷 🖗 GPIO23	Enumeration of BYTE	not used		not used	not used		configuration of GPIO23	
🔷 🖗 GPIO24	Enumeration of BYTE	Input		Input	not used		configuration of GPIO24	
🔌 GPIO25	Enumeration of BYTE	not used		not used	not used		configuration of GPIO25	



30



Security concerns regarding Pin Control in PLCs

1. No Interrupt for Pin Configuration

 How the OS knows about the modification of pin configuration?

2. No Interrupt for Pin Multiplexing

• What if somebody multiplex a Pin at runtime?




































49
















































Problem statement

- What if we create an attack using pin control that:
 - Do not do function hooking
 - Do not modify executable contents of the PLC runtime.
 - Do not change the logic file
- Obviously we consider other defenses available (e.g. logic checksum is also there)



Pin Control Attack

- Pin Control Attack:
 - manipulate the I/O configuration (Pin Configuration Attack)
 - manipulate the I/O multiplexing (Pin Multiplexing Attack)
- PLC OS never knows about it.



Two variants of the work and bypassing the protections

• 1. a rootkit which manipulates the Pin Configuration.

• 2. a malicious C code which can do the same.



What do we need for a Pin Control Attack

- First variant (rootkit)
 - Root privilege
 - Knowledge of SoC registers
 - Knowledge of mapping between I/O pins and the logic
- Second variant:

- Equal privilege as PLC runtime
- Knowledge of mapping between I/O pins and the logic

How actually a PLC Controls its I/O (very basic)

• PLC Prepares I/O for the logic:

- Map physical I/O base addresses
- Enable Input/Output Mode of the register and use it (write or read it)



First variant (rootkit), precise I/O manipulation

• Exploiting the I/O configuration.

• Utilizing Processor Debug Registers.



Debug Registers

black hat ∈

- Designed for debugging purpose.
- Function hooking intercept the function call and manipulate the function argument.
- We use debug registers in ARM processors to intercept memory access (No function interception, no function argument manipulation)

==Phrack Inc.==	
Volume 0x0c, Issue 0x41, Phile #0x08 of 0x0f	
=[Mistifying the debugger,]=	
======================================	
==[halfdead@phear.org]==	
[Introduction	
Over the years, there have been a plethore of techniques and methods of	
hiding one's presence in a backed system. Many of them were focused on	
$\square \square $	No the

Devil is in detail...

- Combination of Pin configuration registers and ARM Processor Debug register
 - Put the mapped I/O address to the debug register.
 - Manipulate the Pin Configuration or multiplexing upon I/O memory access.



How Pin Configuration Attack Works?

Manipulate Read	Manipulate Write						
1. Put I/O Address into Debug register	1. Put I/O Address into Debug register						
read(I/O, Pin)	write(I/O, Pin)						
2. Intercept Read Operation from I/O	2. Intercept Write Operation to I/O						
3. Set Pin to Output Mode	3. Set Pin to Input (write-ignore)						
4. Write Desired Value to Output	write() continue						
read() continue							
Pin Control Attack actions							

81

Wait a minute!

- Shouldn't the PLC runtime fail or get terminated because of I/O failure?
 - Nope!

O black hat EUROPE 2016

 Remember no interrupt therefore, everything looks fine to the PLC runtime!



Implementation of the attack

- Before implementation we have to know how the PLC runtime interact with the I/O in physical world!.
- We need to know more about I/O registers
- We actually need to write the driver for target I/O in our PLC.

Simple Logic

Lets test it with a simple Function Block Language Logic.



input : State of In.24output: State of Out.22 Main Logic; while True do read input; while input True do switch_state(output, five seconds); //states are High or Low. end if input False then hold the state of the output; else go to first while; end

Simple Logic 2

Second Logic for a real PLC



O black hat EUROPE 2016

85





Lets look at it.

Demo

Digital



Codesys Dynamic and Static Analysis

I/O Mapping

ł

Look for Base Addresses of I/O

Library function 📃 Data 🧧 Regular function 📕 Unexplored 📕 Instruction	Externa	l symbol				
	DCD	0x72322A06,	0x8003FB5D,	ØxD182CBBC,	0x575B9332,	0x836A03F9
	DCD	0xD1F2679A,	0xD1B89713,	0xD9BAA704,	0xB6C6F738,	0x3FA85B8C
	DCD	0xB78538F5,	0x3D4C2D10,	0x282FF5B0,	0x2B081994,	0x1848E56D
[b6e47f54] open("/etc/3S.dat", O_RDONLY) = 8 <0.001979>	DCD	0xAA8C7B82,	0xDE23AF80,	0xE6144FFD,	0xFE82BA1B,	0x18604BA9
[b6df334c] close(8) = 0 <0.001878>	DCD	0x223D3D45,	0x1A00B106,	0x825AC9E5,	0xF425FFE6,	0xB19B375B
[b6e47f54] open("/proc/cpuinfo", 0 RDONLY) = 8 <0.001354>	DCD	0xEF878EA7,	0x172C1C83,	0x40E54D04,	0x588CDBC8,	0x1B19AC0F
[b6df334c] close(2) www.b//>	DCD	0x7ED50852,	0xE0C950C8,	0x9C67C354,	0x3DA8F807,	0x421FBB11
	DCD	0x2C816A17,	0xFB3E4375,	0x60DB663, 🗆	0xF15C6122, (0x64514032 👘
[DoT2C114] open("/dev/mem", U_KUWK) = 8 <0.001182>	DCD	0xDA75AF94,	0x9929D1B3,	0x3D910885,	0x8984059F,	0x4F66A58
<pre>[b6es2998] mmap2(NULL, 4096, PROT_READ PROT_WRITE, MAP_SHARED, 8, 0x2020])</pre>	DCD	0x97F2C7D9,	0x4808685D,	0xB24602AE,	0x75828FCA,	0x734C7E16
[b6f2bdo1] close(8) = 0 <0.001246>	DCD	0xD39E5C65,	0x4BF3B903,	0x952C30A7,	0x2F92553B,	0xD2FBF6E7
[b6f2c7e4] open("/ucs/12 11 0 ppwp) - 8 <0 001240	DCD	0xD2AD2C07,	0x47B449B9,	0x46CF816A,	0x581A980D,	0x9B61780
$[bf2c7c4]$ open("/dev/spidev(0.0" 0.00\\D) = 0 < 0.001005	DCD	0xE7864462,	0xA5DA033E,	0x4B3A8C38,	0xA57A4FD0,	0x235575B9
[0012C/C4] 0pen("/0ev/Spidevo.o", 0_KDWK) = 9 <0.001000>		ByR2F7AFC8	0x0077010F	8yh607200	AVARE267AC	AVR01822D4
						E01:E70
			1 3	2 2		
UDIACKNATEUROPE 2016 🔰 🔰			and states	7		89

AP TA

Ш

A

I/O Attack: Rootkit

- Rootkit needs root user to install its code as a Loadable Kernel Module (LKM).
- vmalloc() allocates our LKM. It bypasses Doppelganger.
- Do not do any kind of function hooking, bypasses Autoscopy Jr.
- Can change the logic regardless of logic operation.



I/O Performance of rootkit variant



91

CPU Overhead

Write Manipulation: ~ 5%

Read Manipulation: ~ 23%



Real-Time Features

- Priority Inheritance Mutex support
 - no priority inversion problem in the rootkit!
- Implementation without using any Kernel API with generic spinlocks.
- Using IRQF_NODELAY for ARM debug registers.
- Works in Linux RT, QNX, Lynx, VxWorks (with MMU) RTOSes.



Second Variant of the Attack – No Rootkit ! No Root!

- No need to have rootkit!
- We can do the same with the PLC runtime privilege.
- Overhead below 1%.
- We can either remap the I/O or use already mapped I/O address.



Second variant



Second Variant



What about Analog Control?

- Analog signals are basically aggregation of digital signals.
- Two ways to do it:
 - 1. If part of or entire analog memory can get multiplexed to digital pins attacker can multiplex the pin and write digital bits and basically control the values in the analog memory
 - 2. Using the technique which we can PC+1, we tell the interrupt handler to return the control to the next instruction within the PLC runtime, basically avoiding write operation occur



Analog I/O Manipulation





Lets look at it.



Analog



Other Future Possibilities!

- Attacking pull-up and pull-down resistors in I/O interfaces
- What if we disable them?

O black hat EUROPE 2016

Remotely manipulate the I/O via a powerful electromagnetic field!



Discussions

- For now attacker can:
 - Simply change the logic
 - Modify PLC Runtime executable
- Fixing these attacks are trivial:
 - Proper Authentication
 - Proper Logic Checksum
 - PLC Runtime integrity verification
- Next Step for attackers:
 - Achieve its goal without actually modifying the Logic or Runtime or hooking functions

Conclusions

- Need to focus on system level security of control devices In future more sophisticated techniques come that evade defenses.
 - Pin Control attack is an example of such attacks.

• Pin Control Attack:

- lack of interrupt for I/O configuration registers
- Significant consequences on protected PLCs and other control devices such as IEDs.

• Solution:

- It is hard to handle I/O interrupts with existing real-time constraints.
- Monitoring I/O Configuration Pins for anomalies.
- User/Kernel space separation for I/O memory.

Questions?

Everything that has a beginning has an end.

The Matrix Revolutions.

Contact:

a.abbasi@utwente.nl

mhashemi@quarkslab.com

