Do as I Say not as I Do
Stealth Modification of Programmable Logic Controllers I/O by Pin Control Attack

ALI ABBASI

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

MAJID HASHEMI

PARIS, FRANCE

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.

  (@bl4ckic3)

- Majid Hashemi, R&D researcher (@m4ji_d).

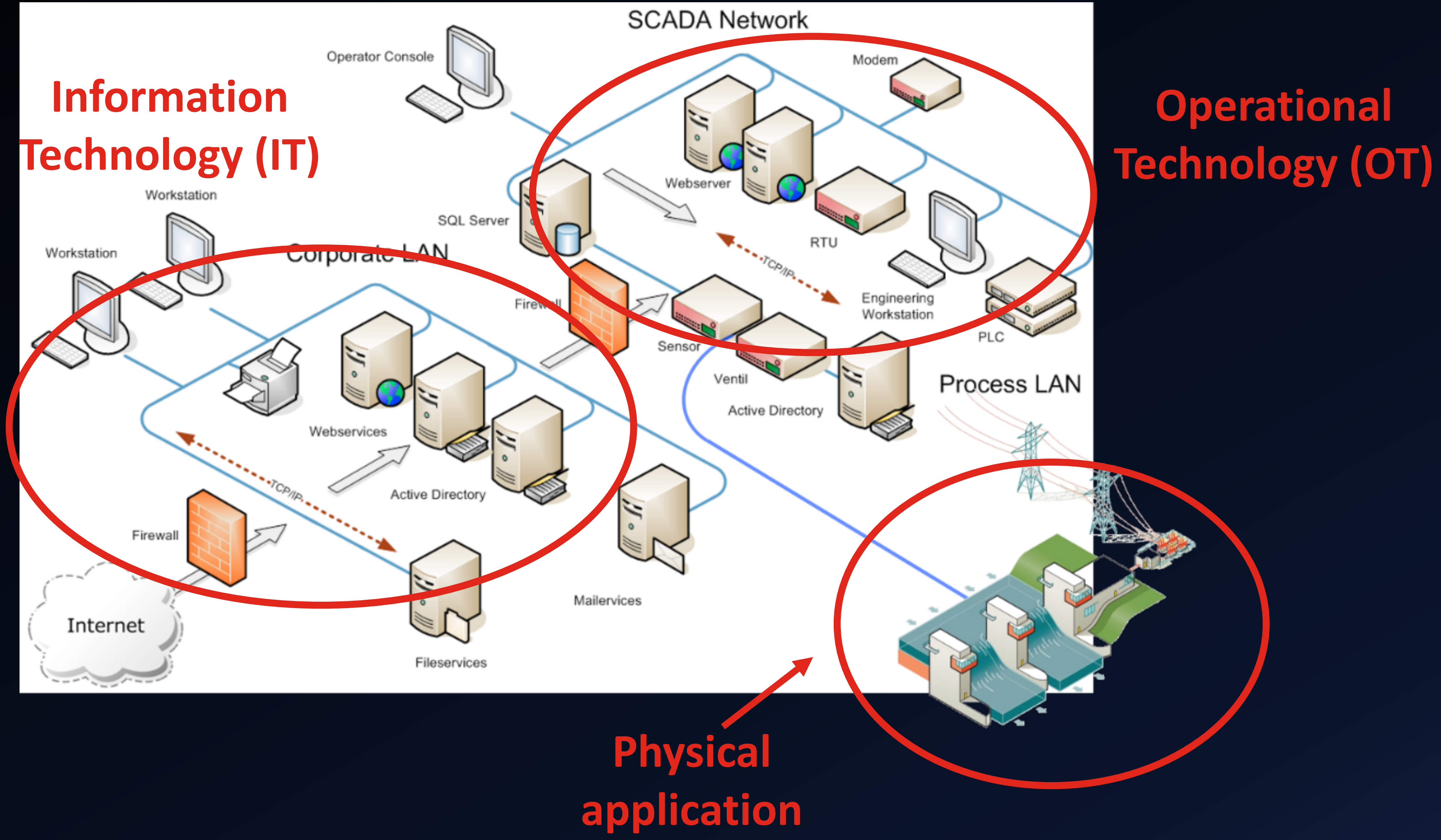
Agenda

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

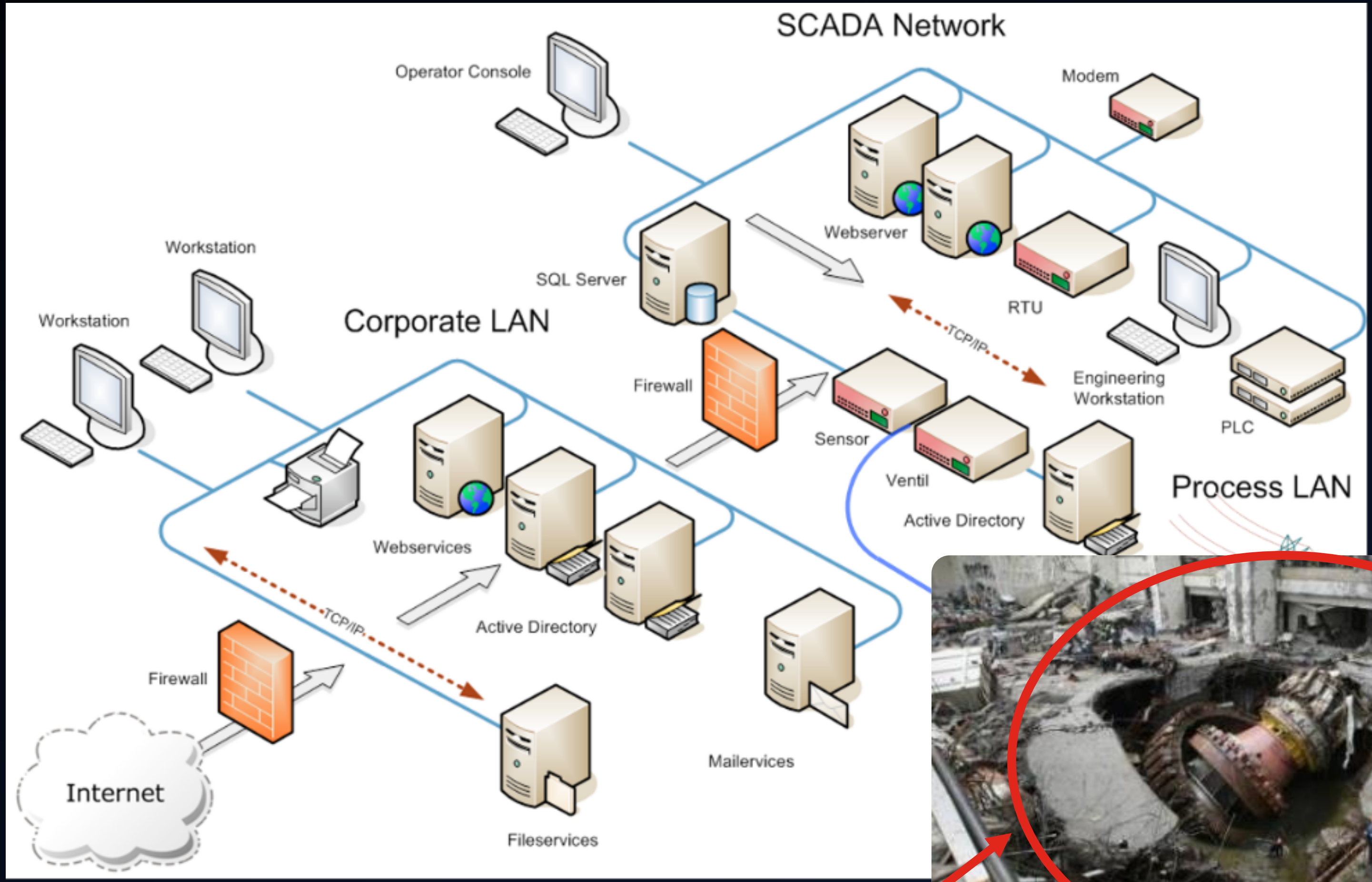
What this talk is about?

- The talk is trying to uncover existing design flaw in PLCs.
- The attack can be used in future by attackers.
- We are not unveiling fully functional malware for PLCs.
- No exploitation techniques, no 0day leak
- We are not going to mention any vendor name.

Industrial Control System



Industrial Control System hacking

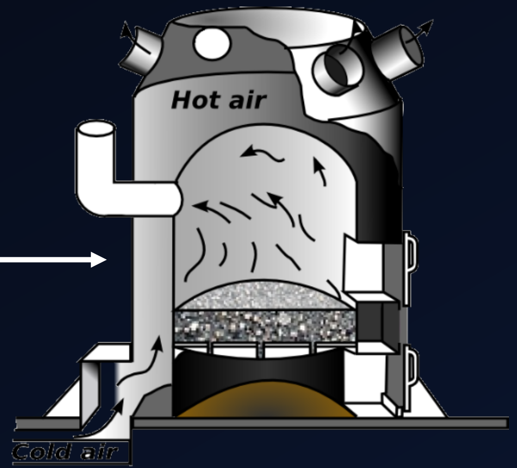


Physical application

Process control 101

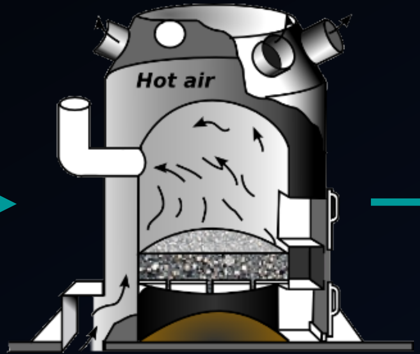
Process control

Set point



Running upstairs to turn on your furnace every time it gets cold gets tiring after a while so you automate it with a thermostat

Control loop

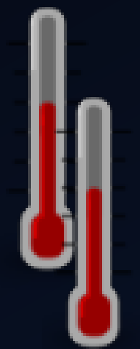


Physical process

Actuators

Adjust themselves to influence process behavior

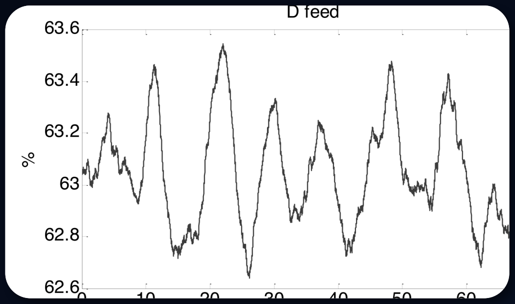
Sensors

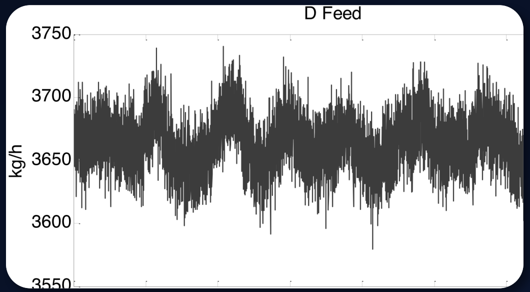


Measure process state

Control system

Computes control commands for actuators



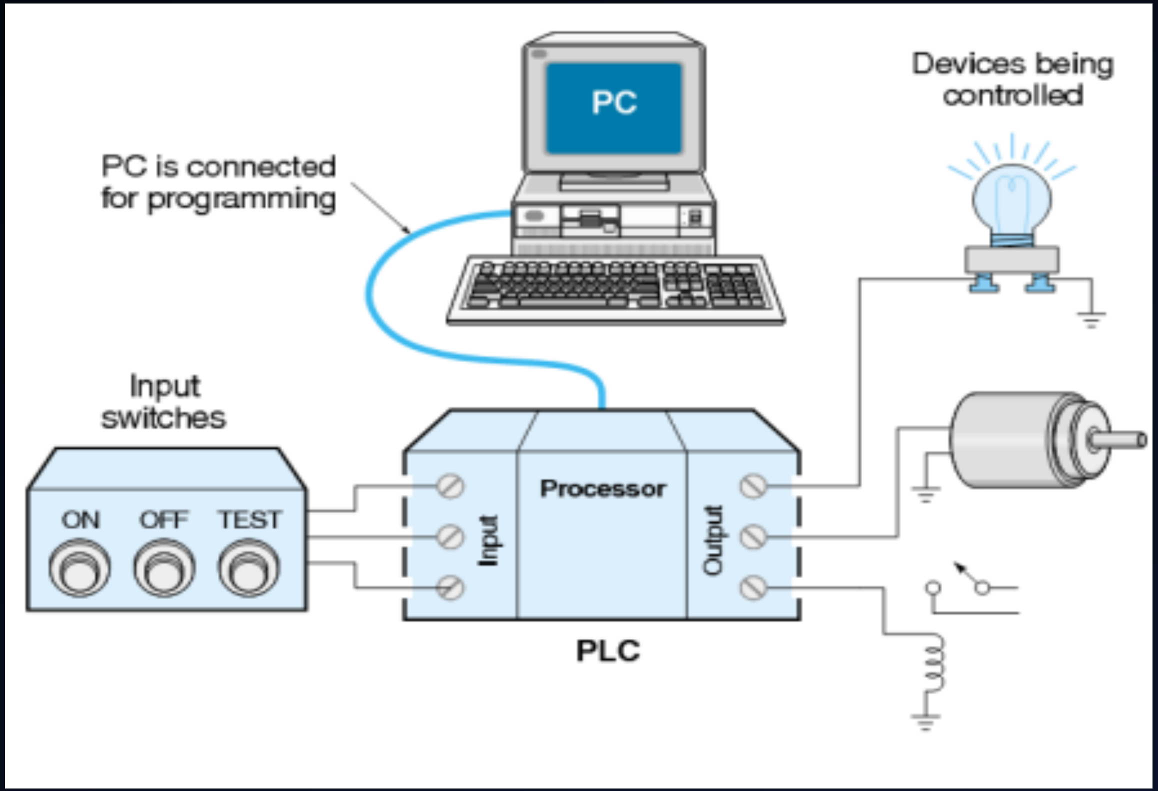


Control equipment

- In large –scale operations control logic gets more complex than a thermostat
- One would need something bigger than a thermostat to handle it
- Most of the time this is a programmable logic controller (PLC)

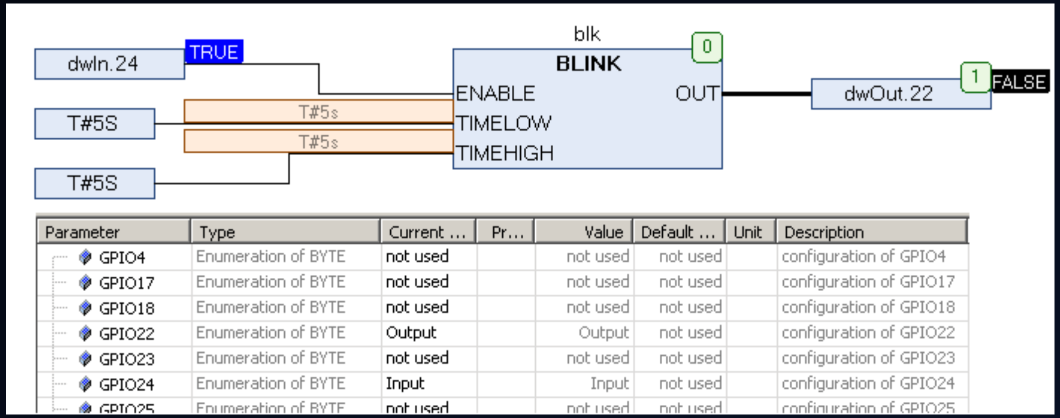
What is a PLC?

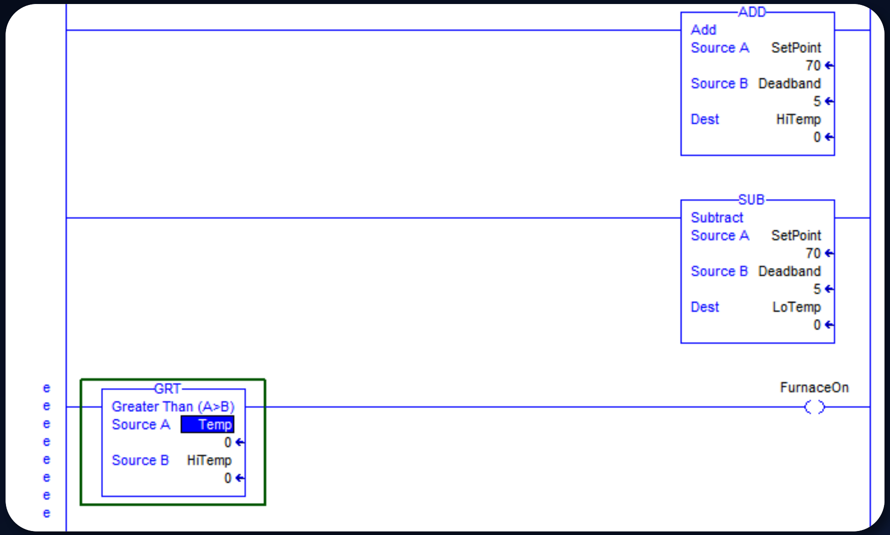
- An Embedded System with RTOS running logic.



Control logic

- It is programmed graphically most of the time
- Defines what should/should not happen
  - Under which conditions
  - At what time
  - Yes or No proposition





[if input 1] AND [input 2 or input 11] -> [do something in output 6]

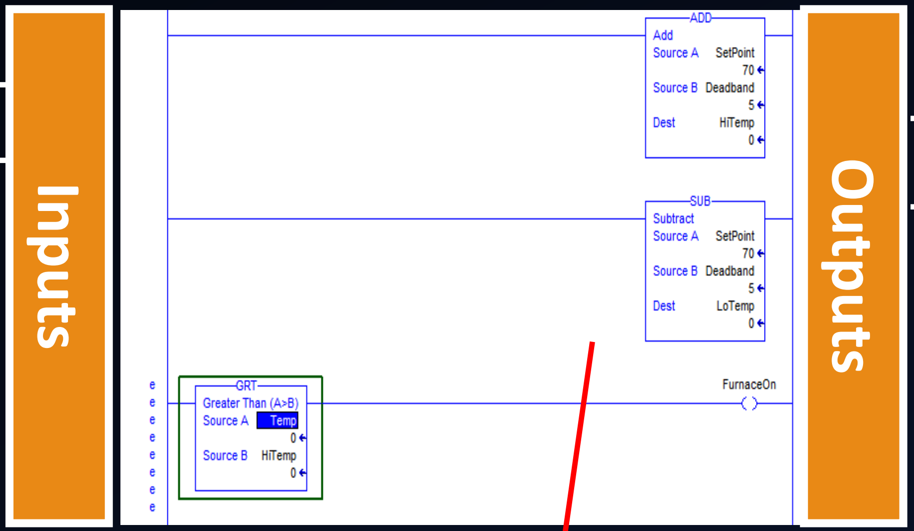
If tank pressure in PLC 1 > 1800 reduce inflow in PLC 3

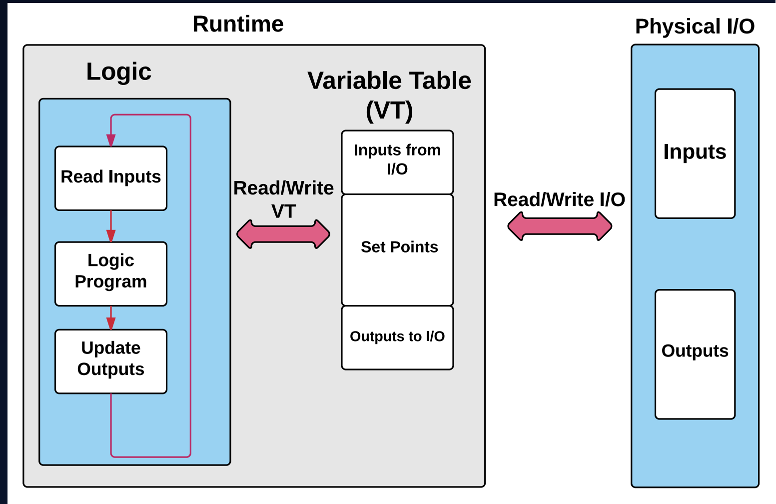
How PLC Works

1. Copy data from inputs to temporary storage
2. Run the logic
3. Copy from temporary storage to outputs

Sensors

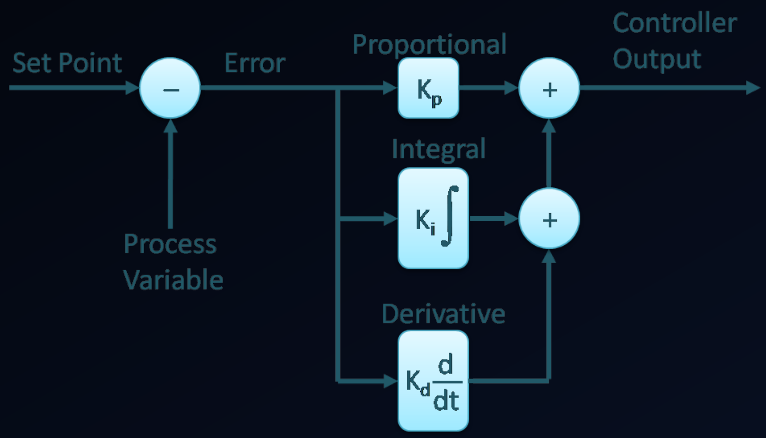
Actuators



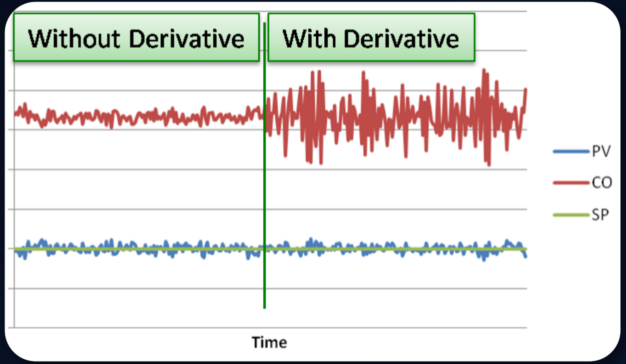


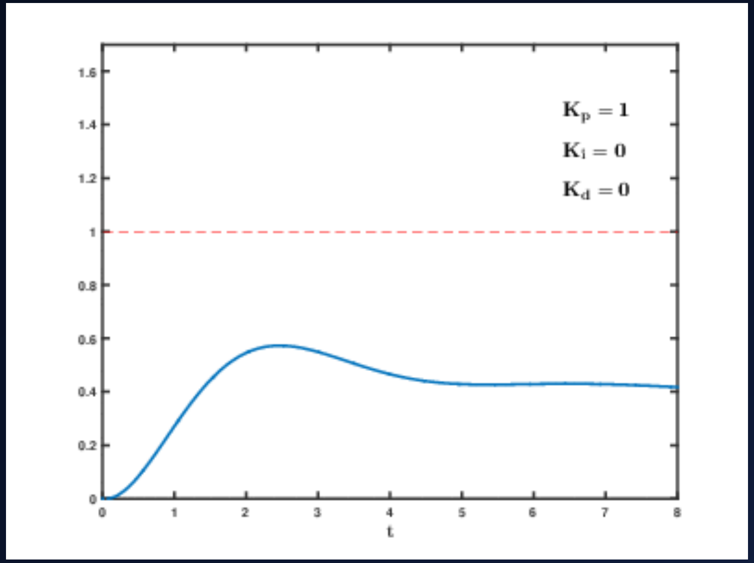
Control algorithm

- Used to compute output based on inputs received from control logic
- **PID: proportional, integral, derivative –** most widely used control algorithm on the planet
- PI controllers are most often used



$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right)$$





Jacques Smuts „Process Control for Practitioners“

Existing Attacks and Defenses for Embedded Systems

Applicable to the PLCs

Current attacks against embedded systems

- Authentication bypass
  - Attacker find a backdoor password in the PLC.
- 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
- Hooking functions for ICS malwares

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

Requirement for Applicable Defenses for PLCs

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

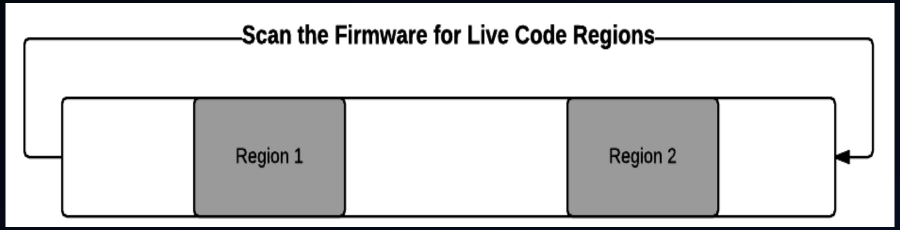
System-level protection for PLCs

- 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

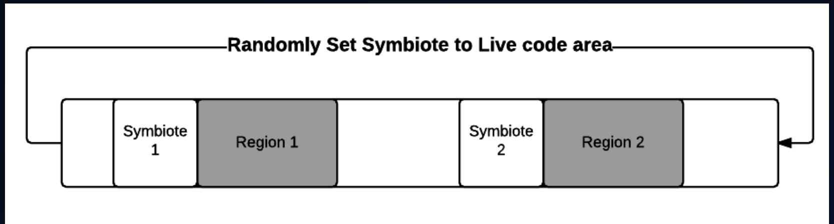
How Doppelganger Works

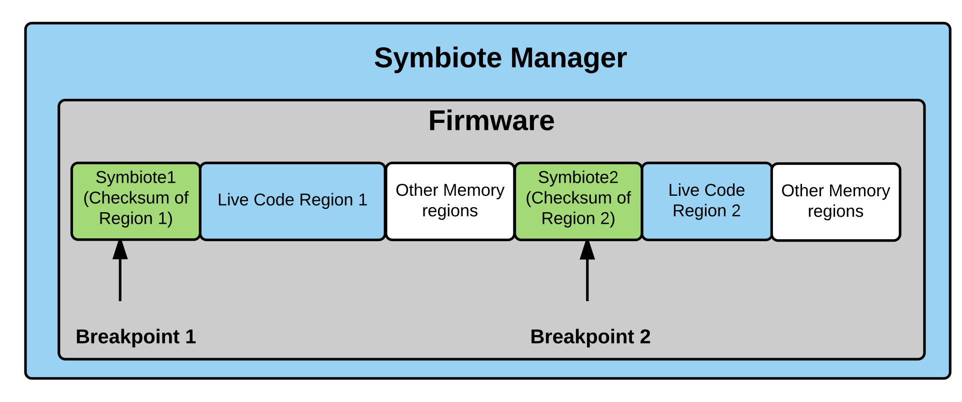
- Scan the firmware of the device for live code regions and insert symbiotes randomly.

1



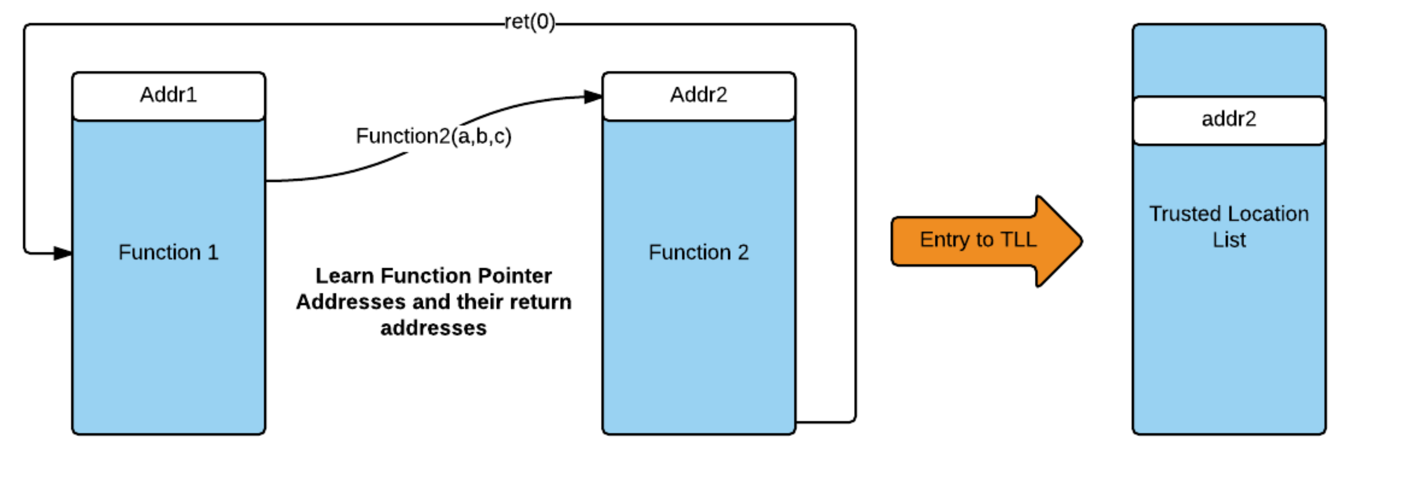
2





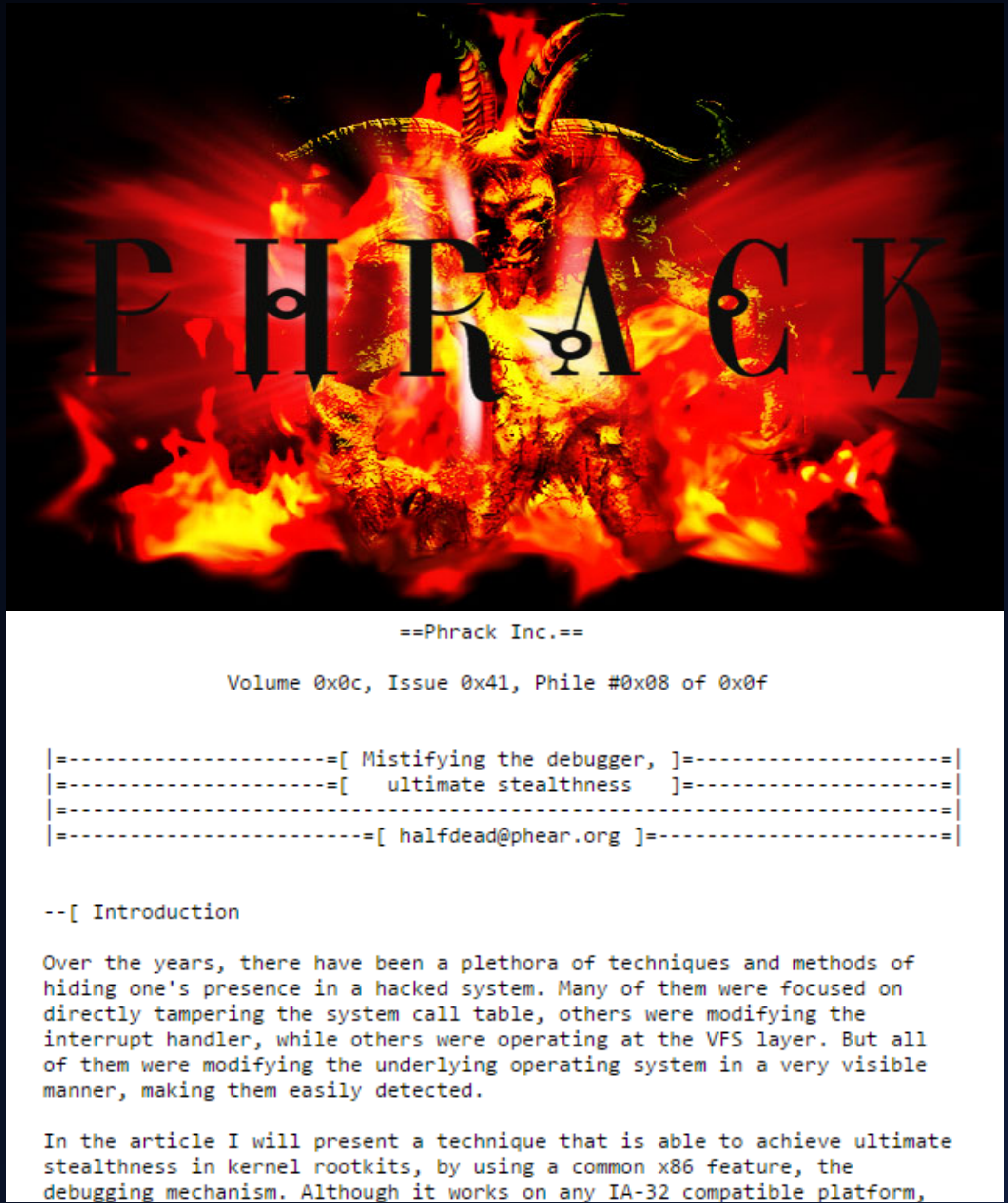
How Autoscopy Jr works

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



Debug Registers

- 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, ]=--------------------=|
|=--------------------=[    ultimate stealthness   ]=--------------------=|
|=-----------------------------------------------------------------------=|
|=-----------------------=[ halfdead@phear.org ]=-----------------------=|
```

--[Introduction

Over the years, there have been a plethora of techniques and methods of hiding one's presence in a hacked system. Many of them were focused on directly tampering the system call table, others were modifying the interrupt handler, while others were operating at the VFS layer. But all of them were modifying the underlying operating system in a very visible manner, making them easily detected.

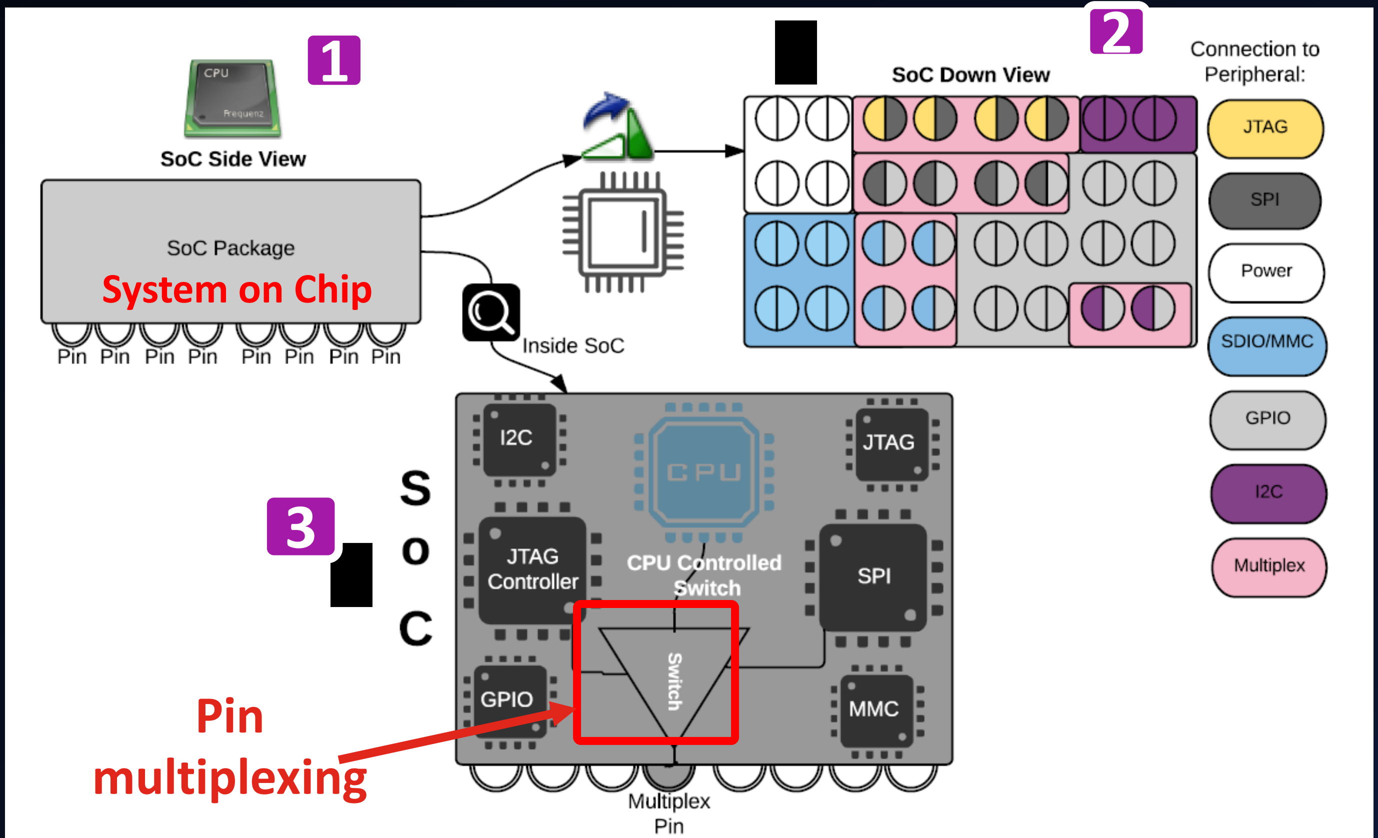
In the article I will present a technique that is able to achieve ultimate stealthness in kernel rootkits, by using a common x86 feature, the debugging mechanism. Although it works on any IA-32 compatible platform,

Pin Control

Background on Pin Control

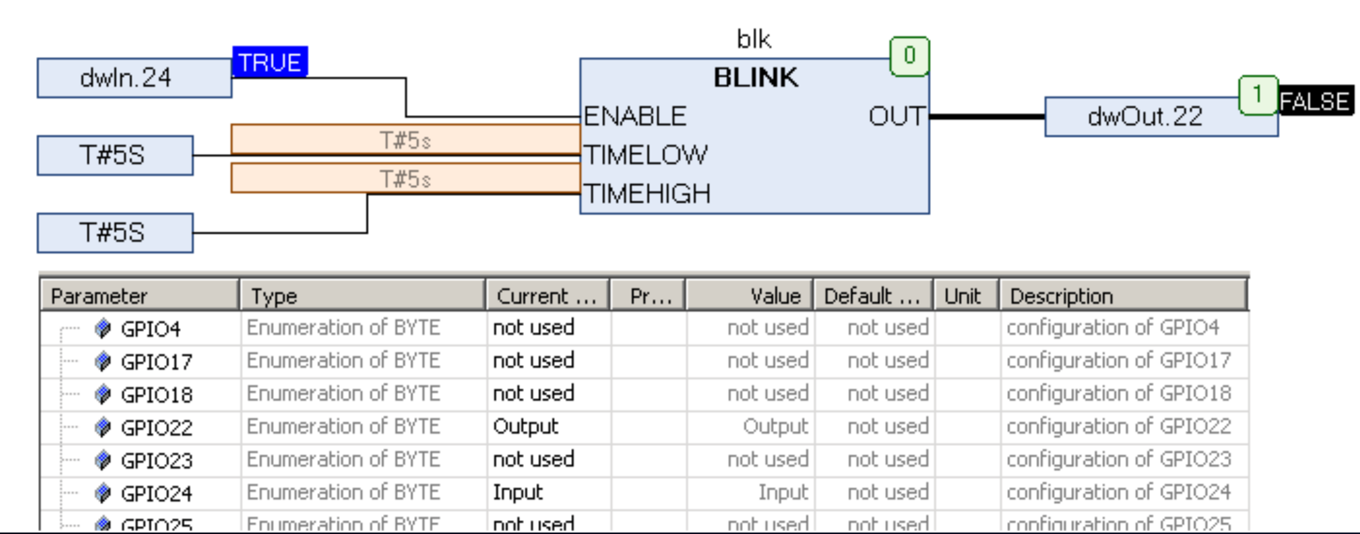
Pin Control subsystem

- Pin multiplexing (type)
- Pin configuration (in/out)

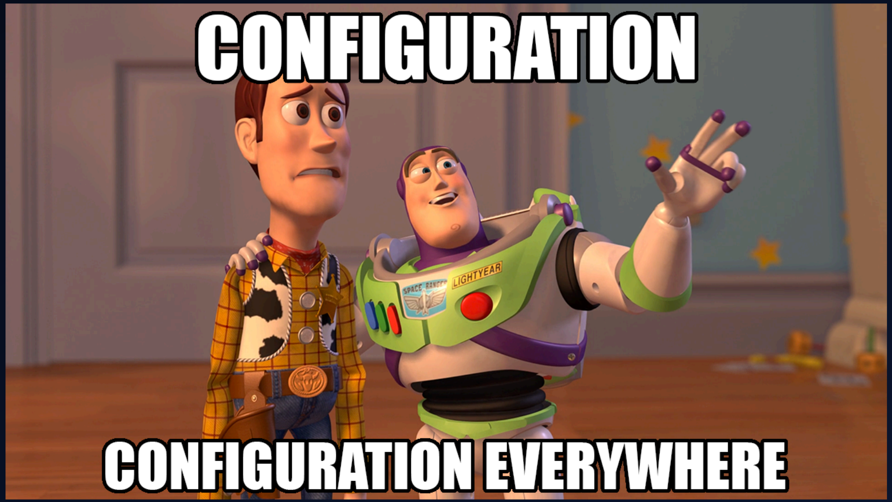


Pin Configuration

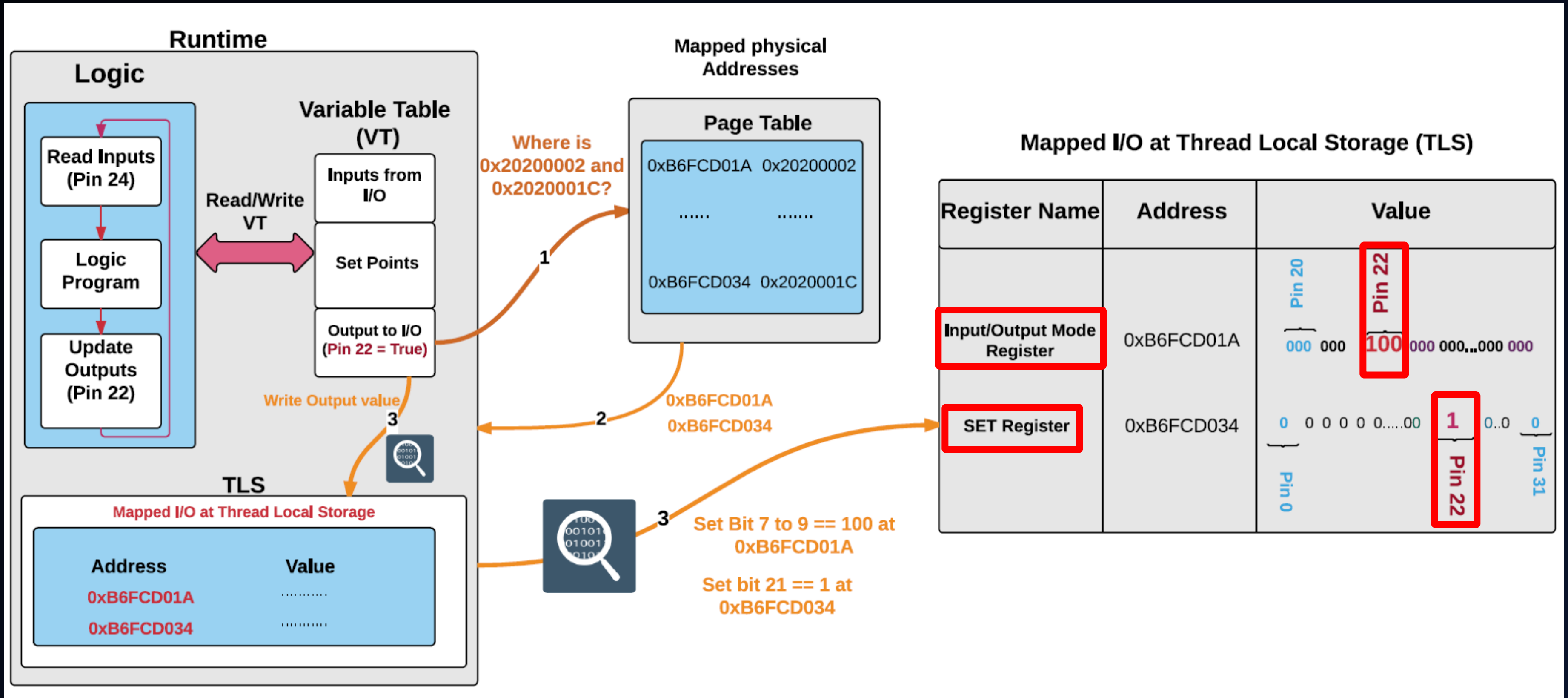
- Input Pin
  - readable but not writeable
- Output Pin
  - readable and writeable



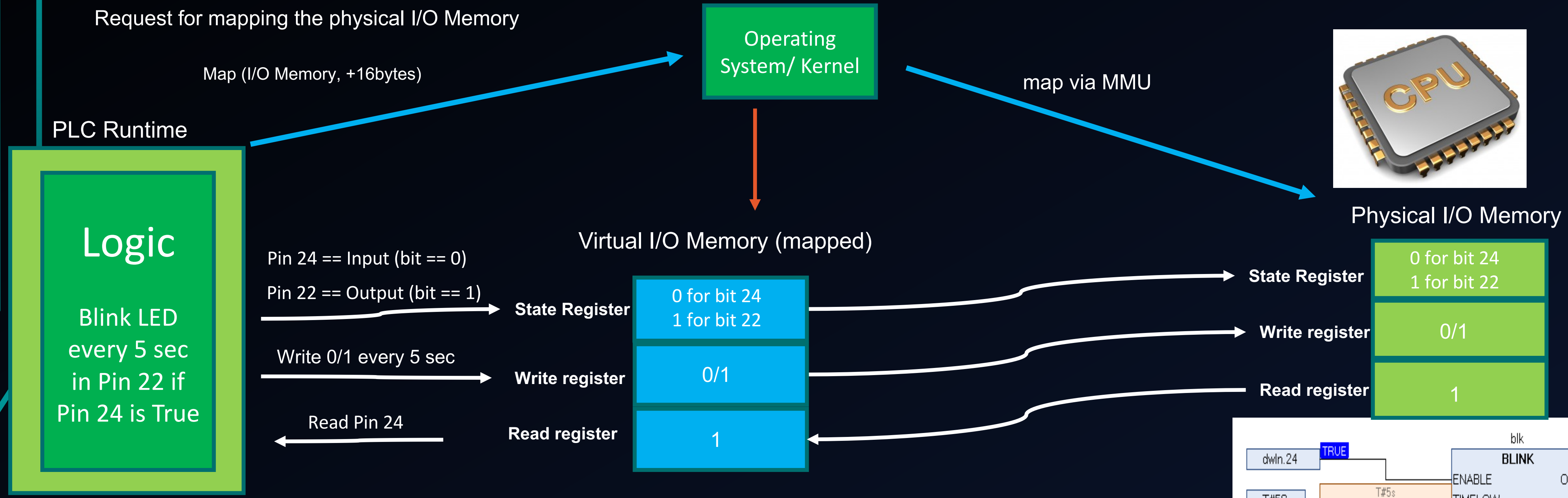
Parameter	Type	Current ...	Pr...	Value	Default ...	Unit	Description
GPIO4	Enumeration of BYTE	not used		not used	not used		configuration of GPIO4
GPIO17	Enumeration of BYTE	not used		not used	not used		configuration of GPIO17
GPIO18	Enumeration of BYTE	not used		not used	not used		configuration of GPIO18
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

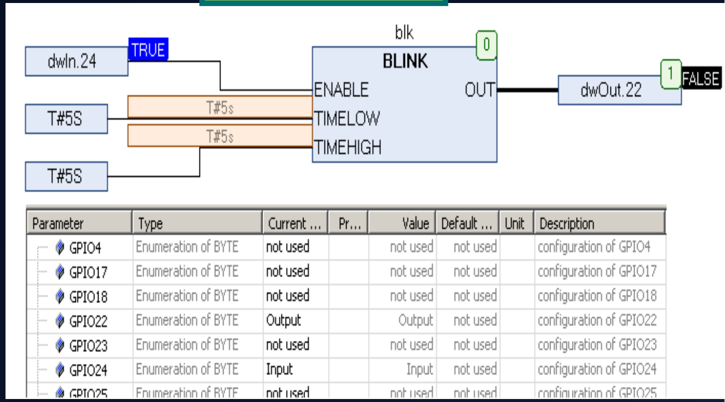


How PLC controls I/O



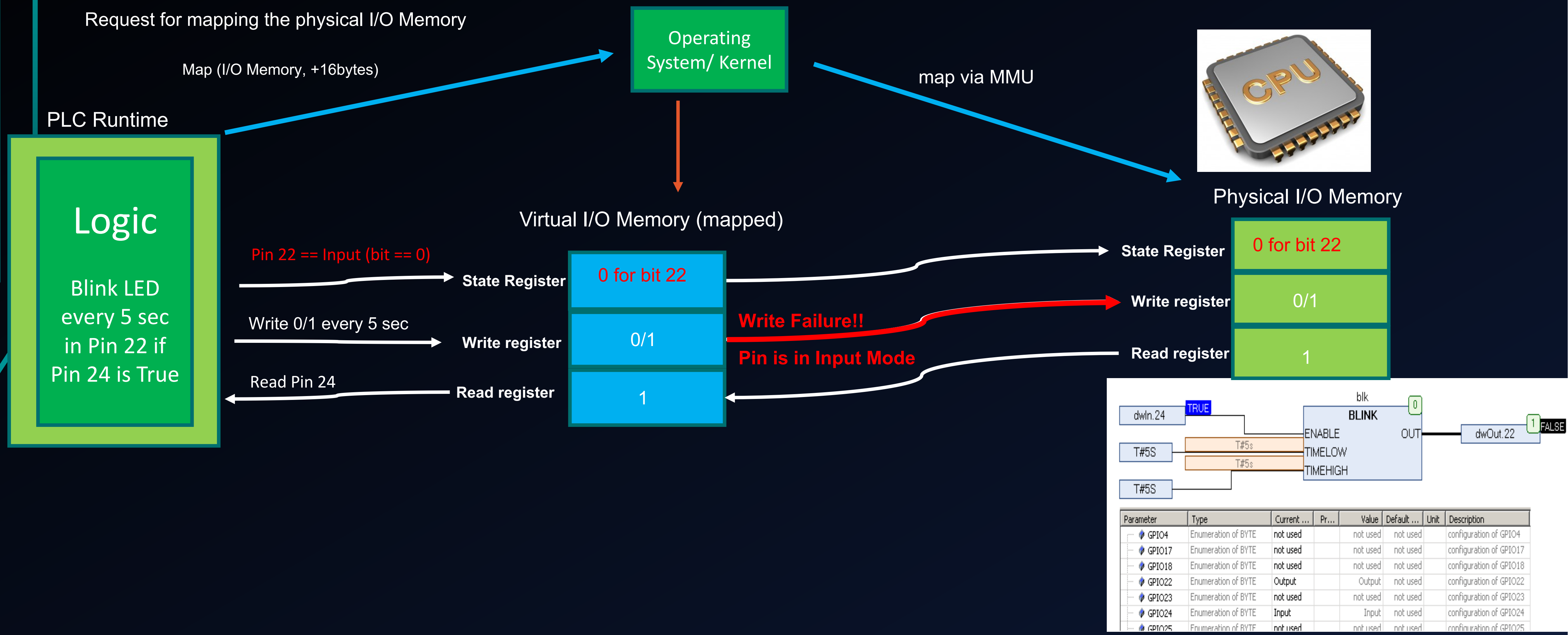
Introducing Pin Control Attack: A Memory Illusion





Parameter	Type	Current ...	Pr...	Value	Default ...	Unit	Description
GPIO4	Enumeration of BYTE	not used		not used	not used		configuration of GPIO4
GPIO17	Enumeration of BYTE	not used		not used	not used		configuration of GPIO17
GPIO18	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

Introducing Pin Control Attack: A Memory Illusion



Parameter	Type	Current ...	Pr...	Value	Default ...	Unit	Description
GPIO4	Enumeration of BYTE	not used		not used	not used		configuration of GPIO4
GPIO17	Enumeration of BYTE	not used		not used	not used		configuration of GPIO17
GPIO18	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

Think of copying files to USB drive

- Similar mapping between physical and virtual addresses
- If USB drive is removed during copy operation, OS reports a warning back

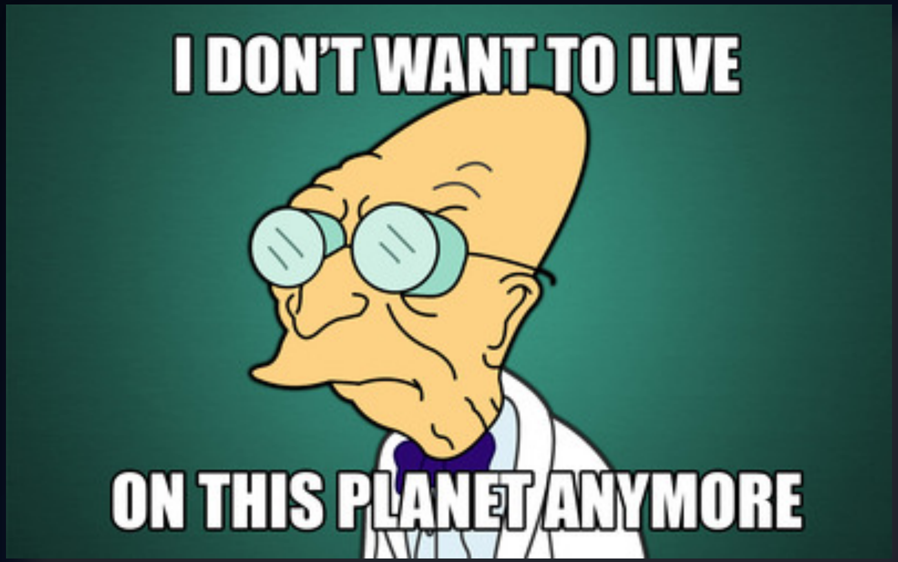
Lets look at it.

Demo 1

Digital

Nobody thought about the same issue for PLCs

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



- PLC design was always about paramount reliability of real-time execution, HIGH up-time and long-term useful life in harsh environmental conditions
- Malicious manipulation of PLC were not part of design considerations :-)

Security concerns regarding pin control

- No interrupt for pin configuration
  - How the OS knows about the modification of pin configuration?
  - What if somebody modifies configuration of a pin at runtime?
  - By switching input pin into output pin, it is possible to write arbitrary value into its physical address
- No Interrupt for pin multiplexing
  - How OS knows about modification of pin multiplexing?
  - What if somebody multiplex a pin at runtime?
  - By multiplexing pin it is possible to prevent runtime from writing value into output pin

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

- Pin Control Attack:
  - manipulate the I/O configuration (Pin Configuration Attack)
  - manipulate the I/O multiplexing (Pin Multiplexing Attack)

- PLC OS will never knows about it.

Two options to achieve the same

1 First version: rootkit

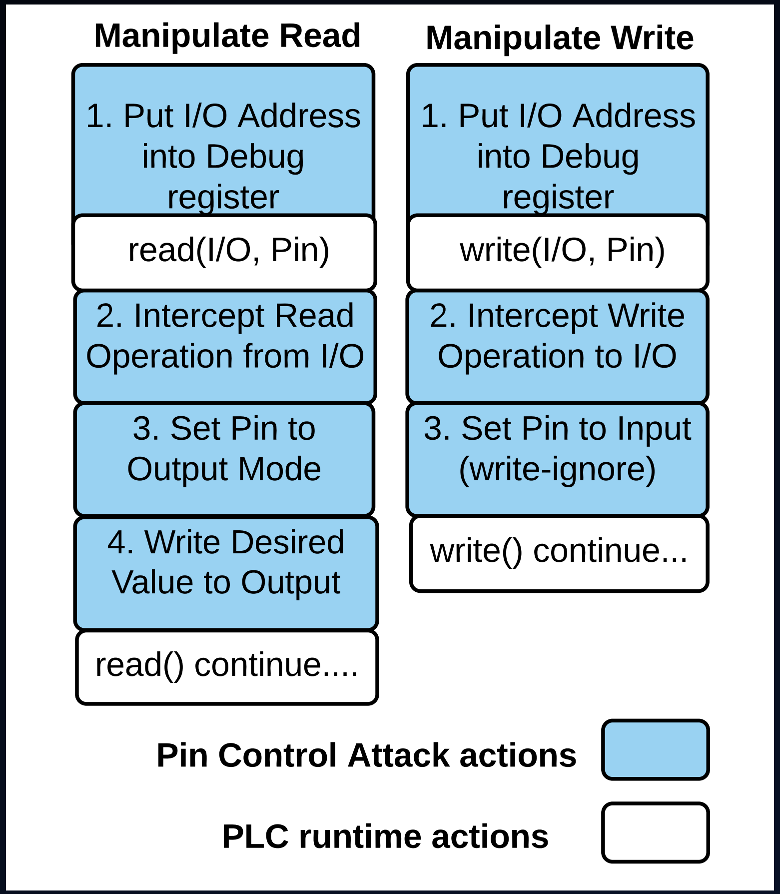
- Root privilege
- Knowledge of SoC registers
- Knowledge of mapping between I/O pins and the logic

2 Second version: C-code (shell code)

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

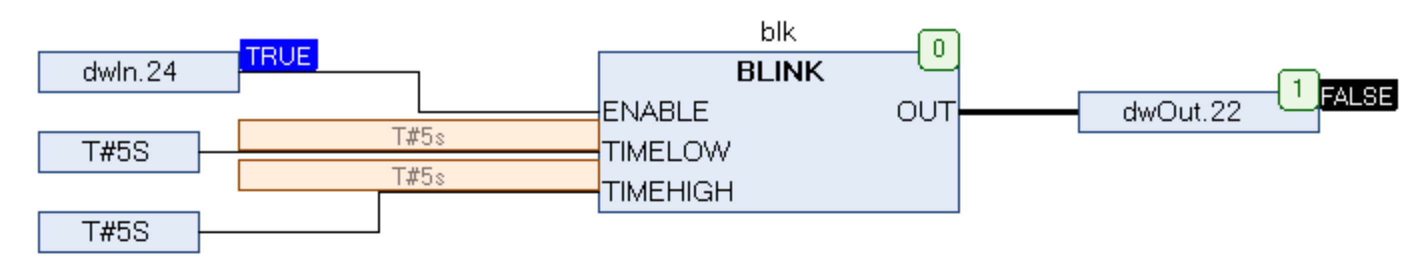
- No function hooking
- No modification of PLC runtime executable content
- No change to logic file

How Pin Configuration Attack Works?



Simple Logic

Lets test it with a simple Function Block Language Logic.

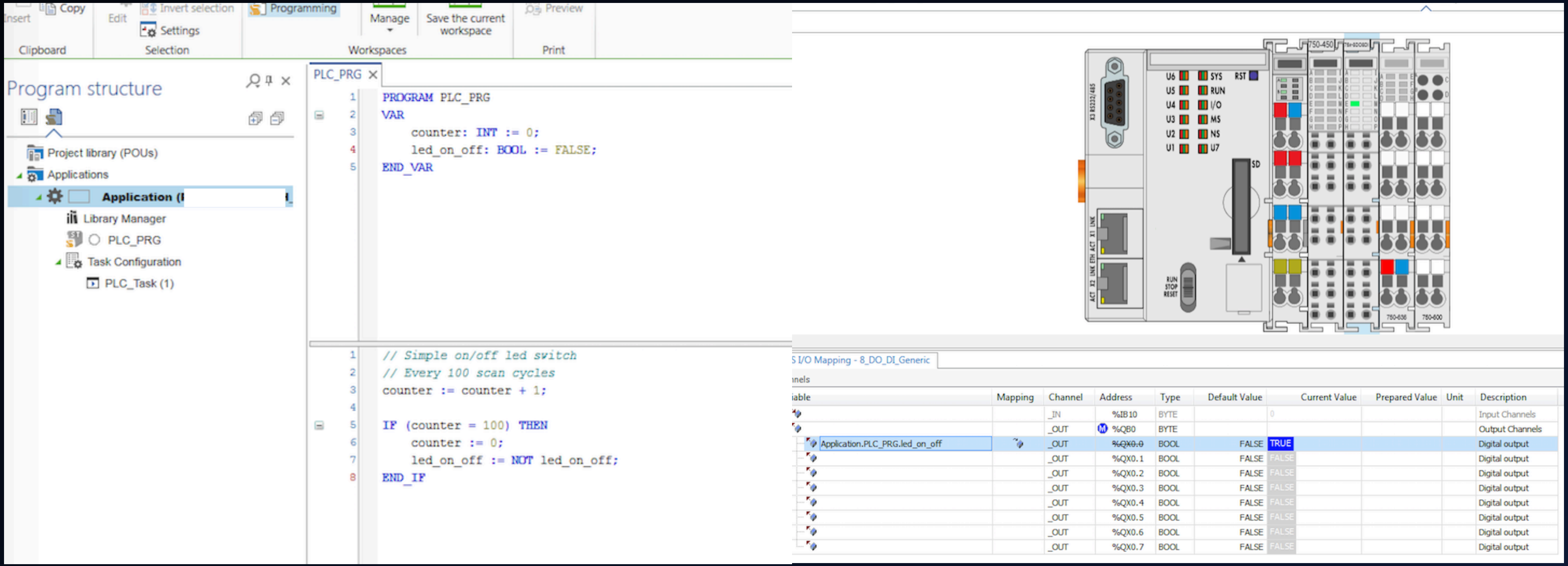


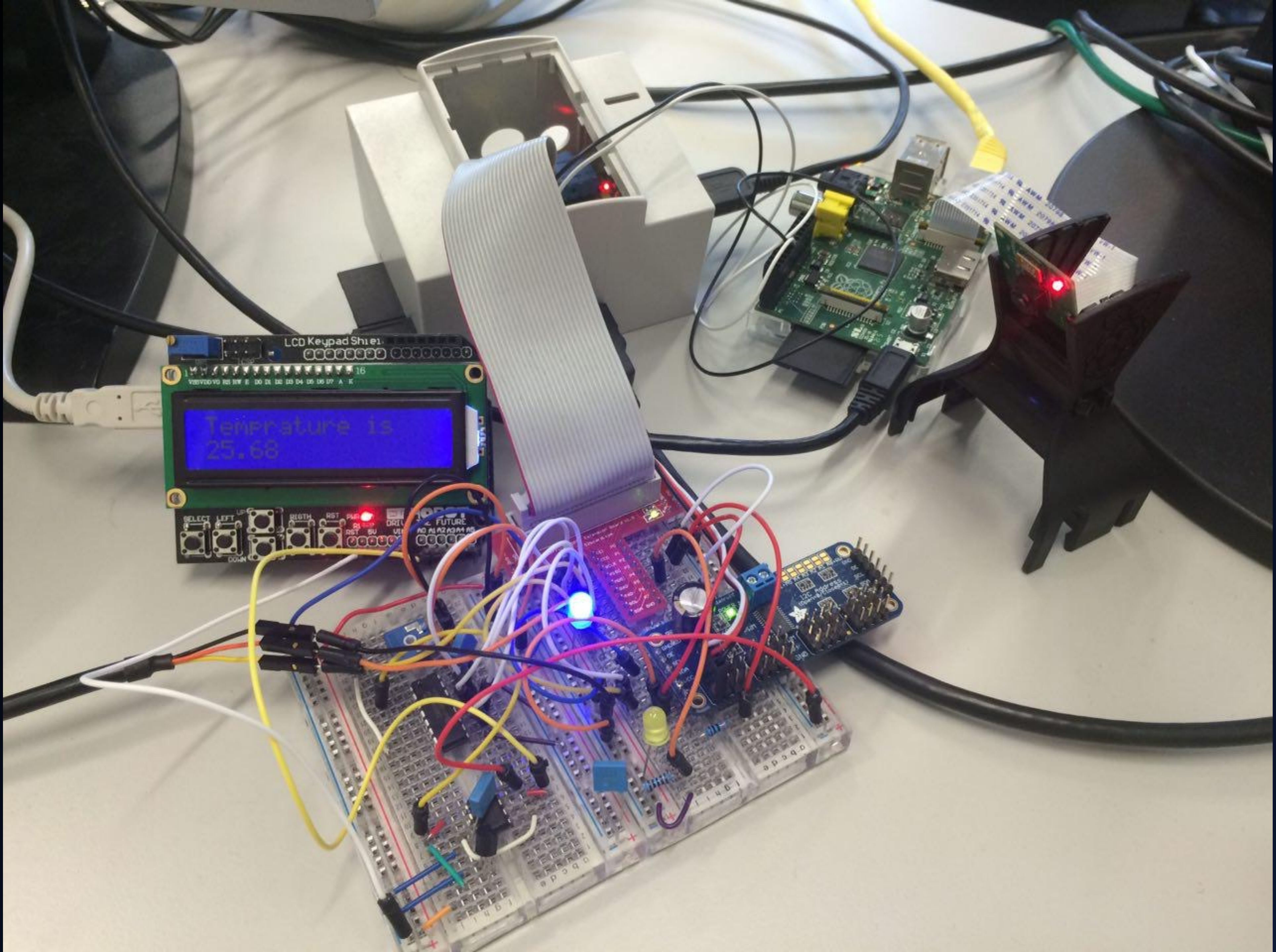
Parameter	Type	Current ...	Pr...	Value	Default ...	Unit	Description
GPIO4	Enumeration of BYTE	not used		not used	not used		configuration of GPIO4
GPIO17	Enumeration of BYTE	not used		not used	not used		configuration of GPIO17
GPIO18	Enumeration of BYTE	not used		not used	not used		configuration of GPIO18
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

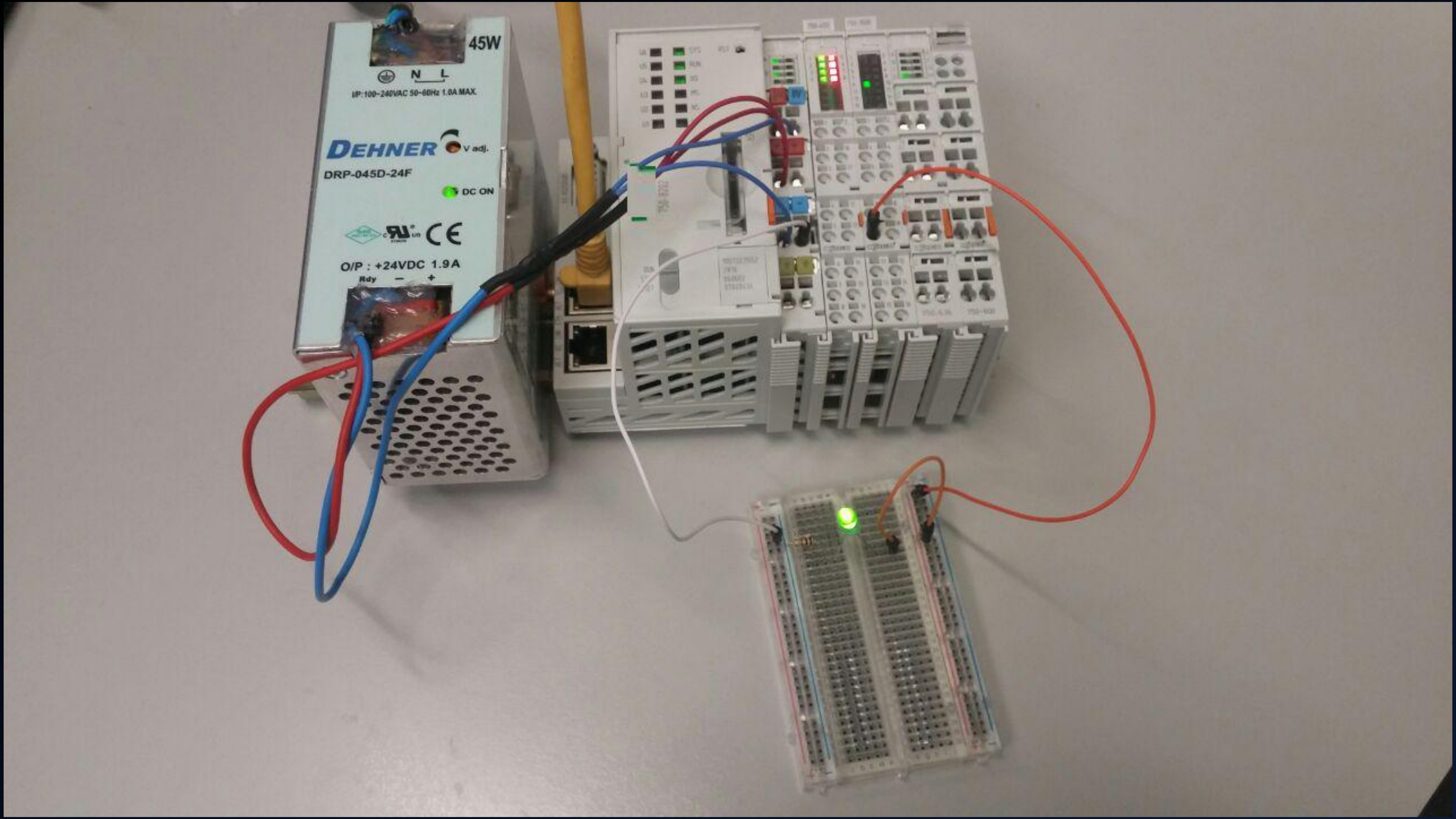
```
input  : State of In.24
output: State of Out.22

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
end
```

Simple Logic 2

- Second Logic for a real PLC




LCD Keypad Shiel
Temprature is
25.68


45W
N L
I/P: 100~240VAC 50~60Hz 1.0A MAX.
DEHNER
V adj.
DRP-045D-24F
DC ON
O/P : +24VDC 1.9A
Rdy – +

Lets look at it.

Demo 2

Digital

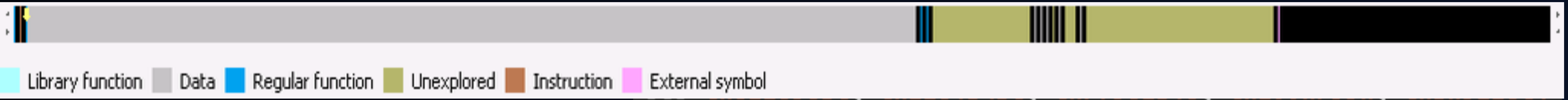
Lets look at it.

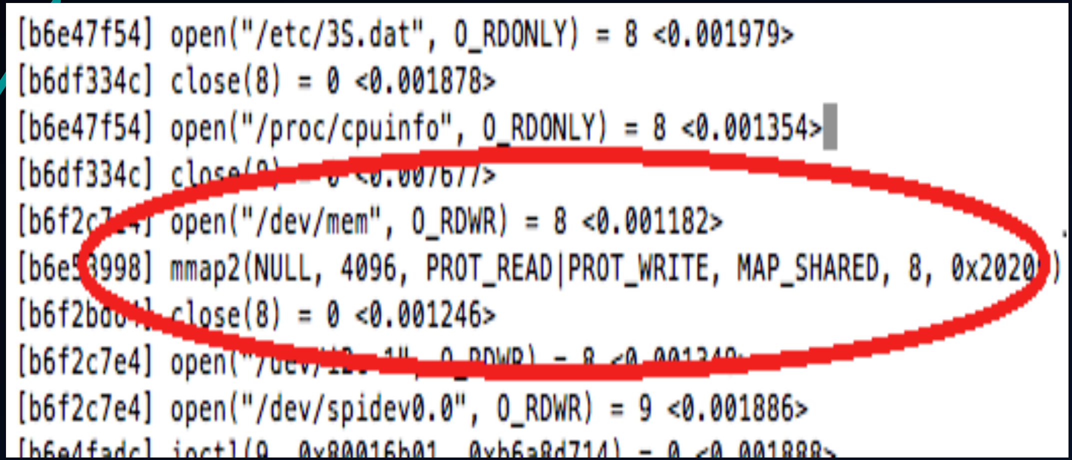
Demo 3

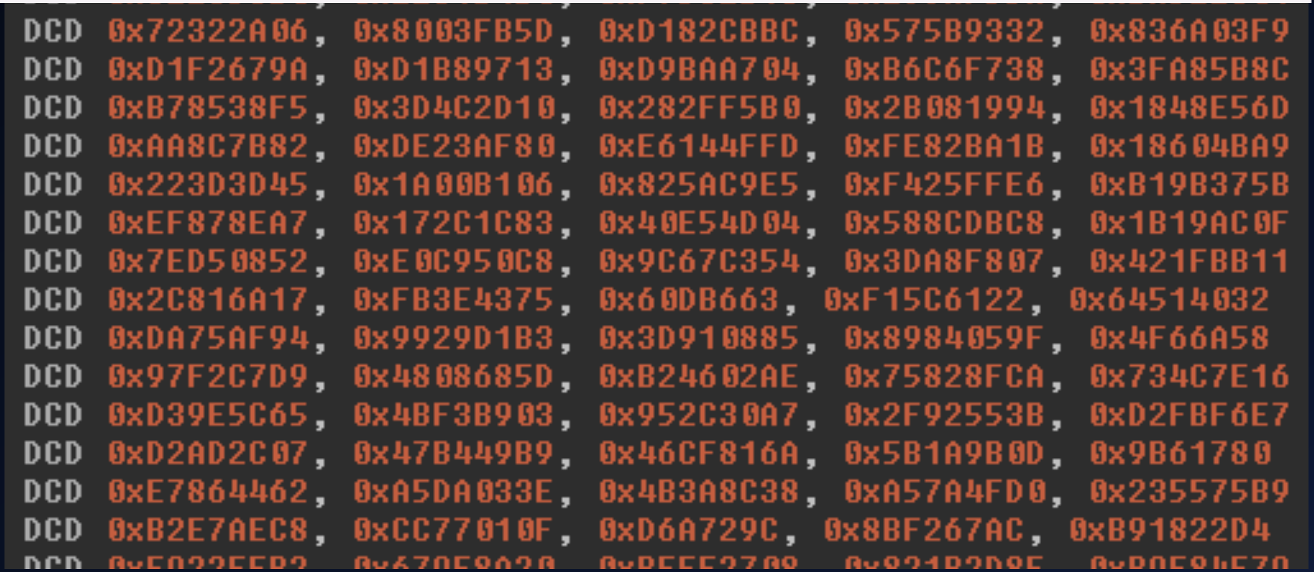
Digital

A PLC runtime Dynamic and Static Analysis

- I/O Mapping
- Look for Base Addresses of I/O

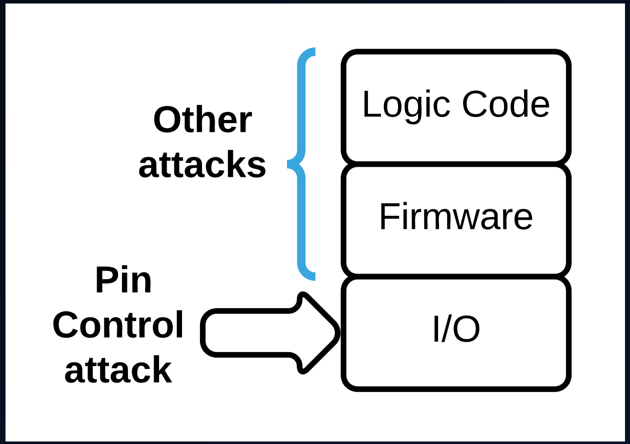




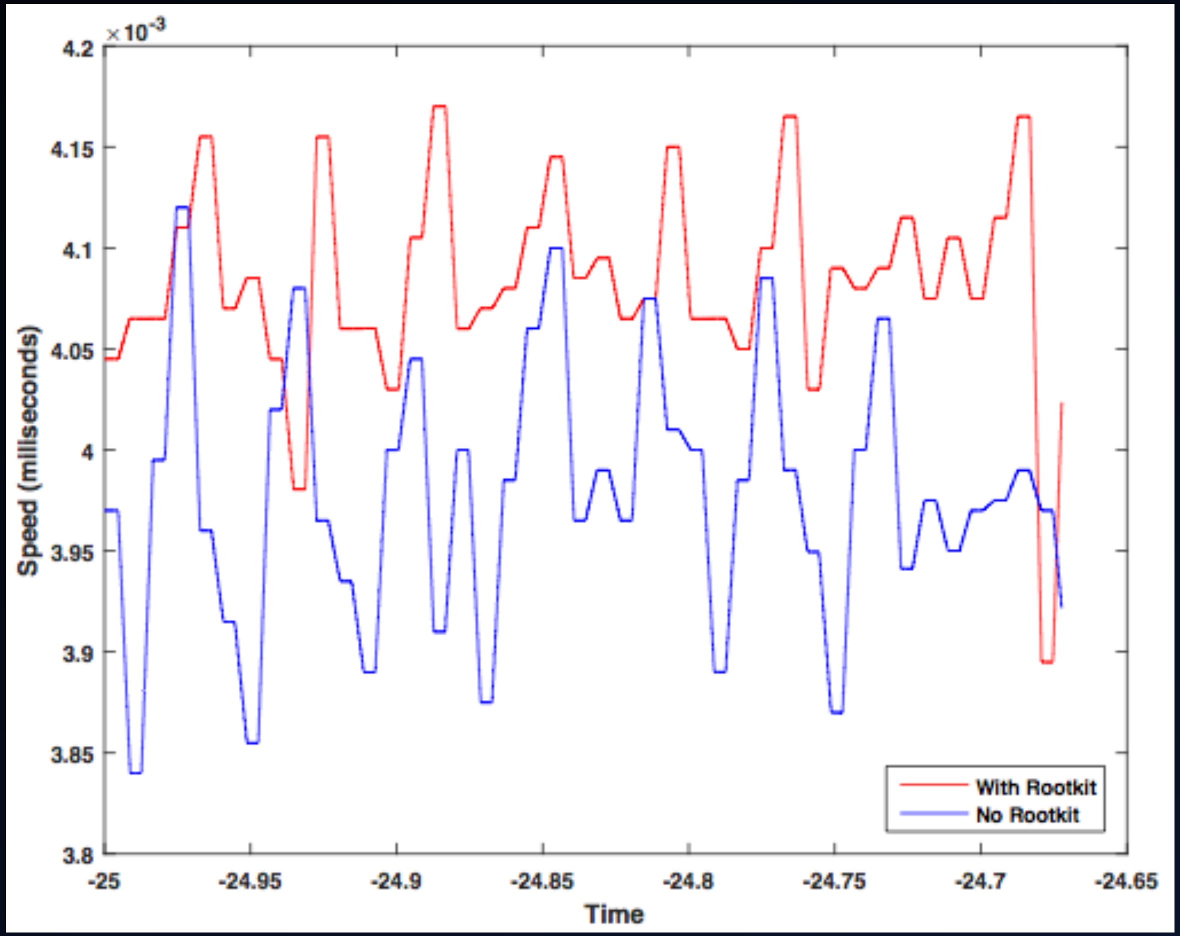


I/O Attack: Rootkit

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



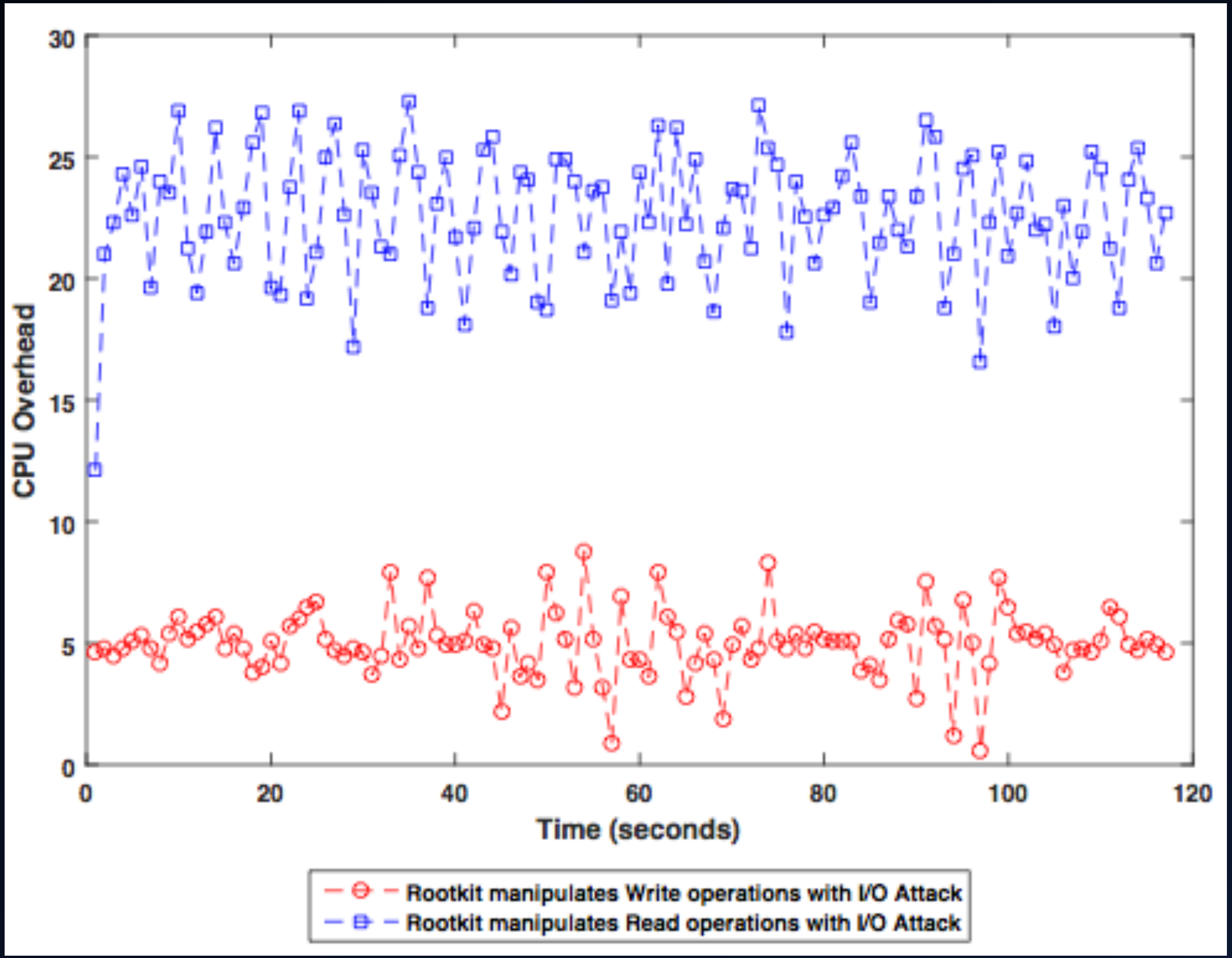
I/O response time fluctuation in rootkit variant



CPU Overhead

Write Manipulation: ~ 5%

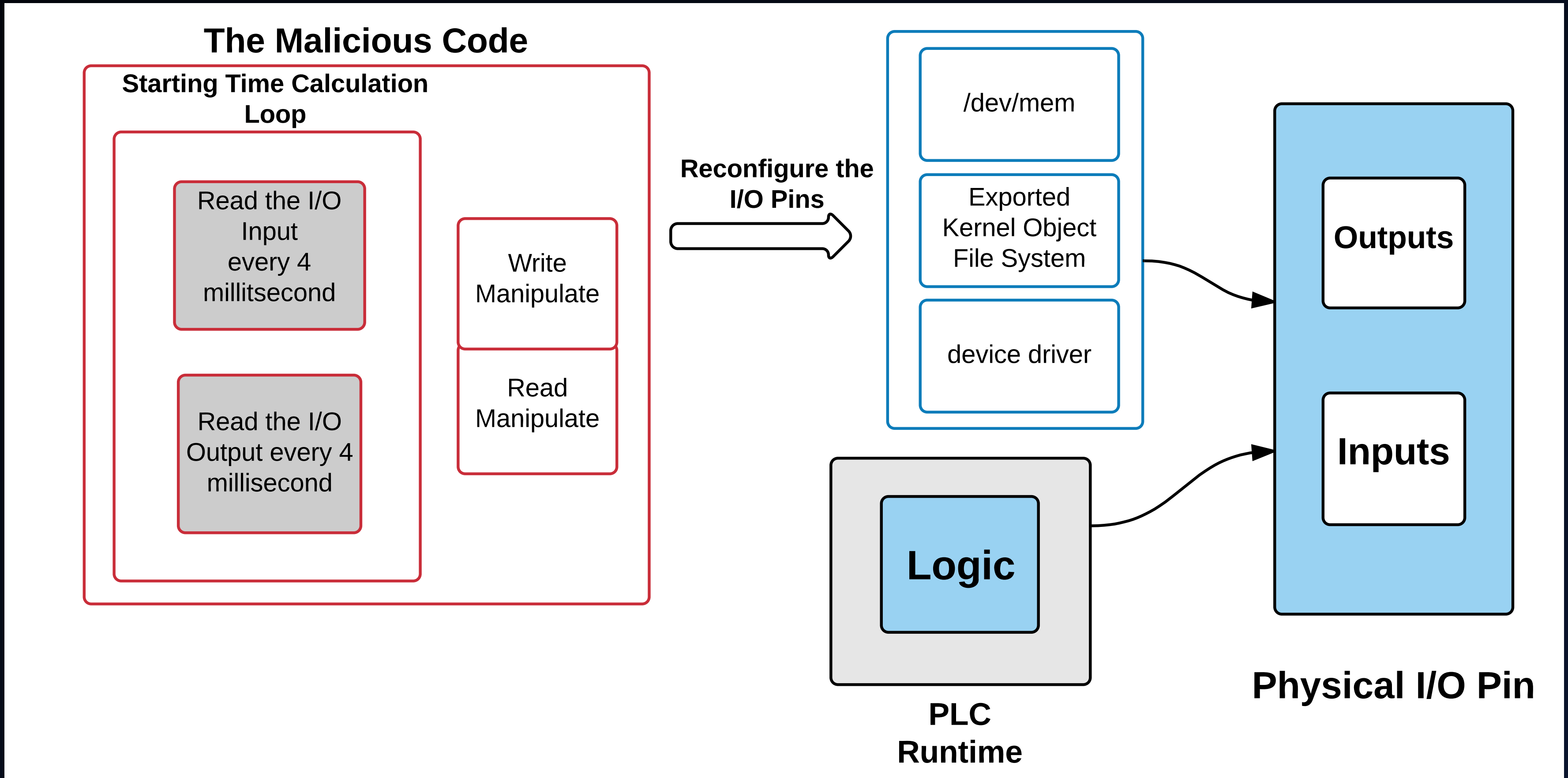
Read Manipulation: ~ 23%



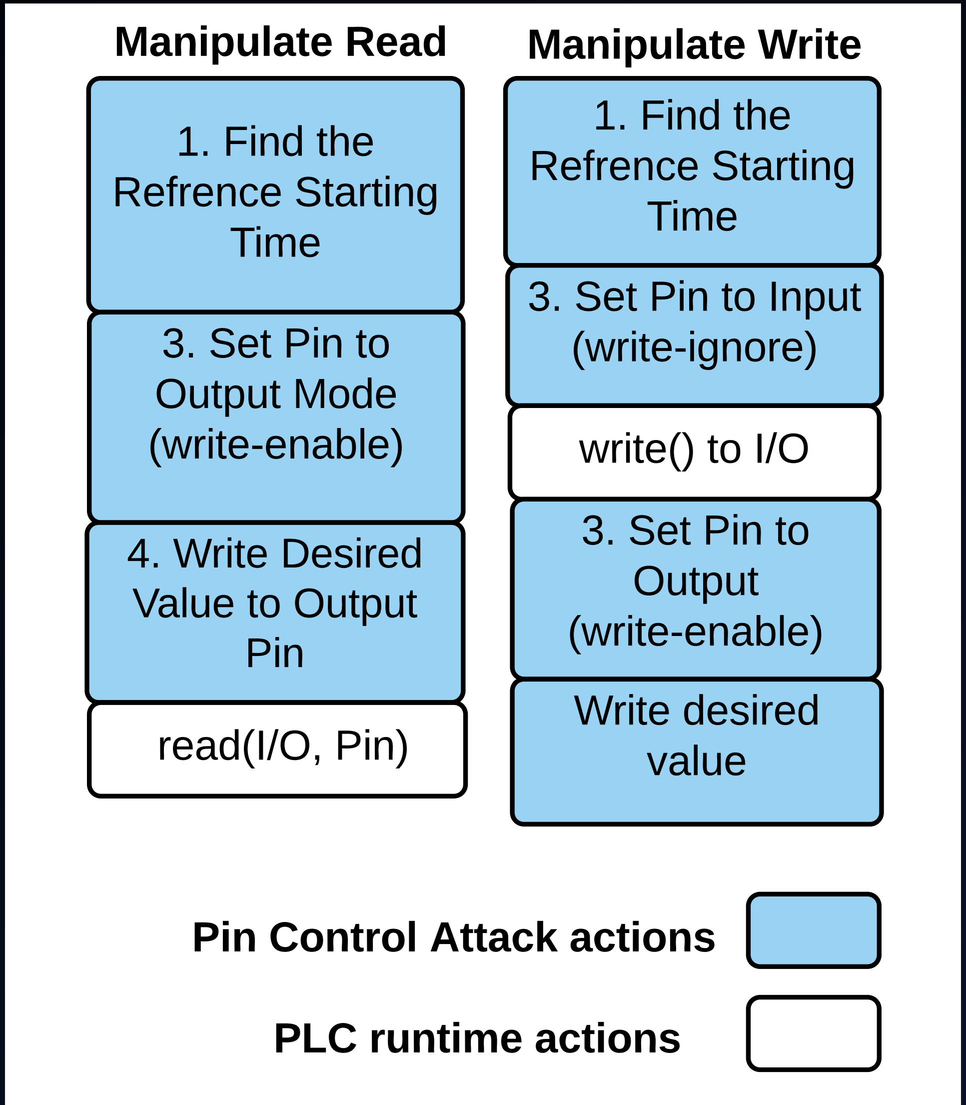
Second Variant of the Attack – No Rootkit !

- 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.
- As shellcode

Second variant



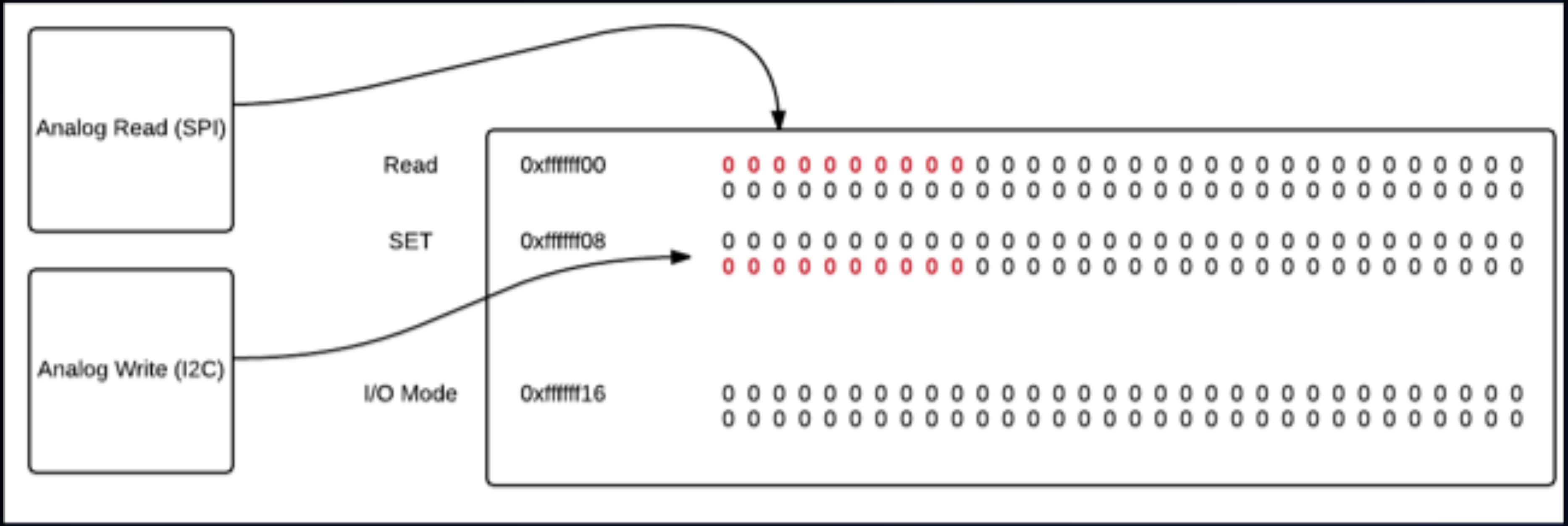
Second Variant



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



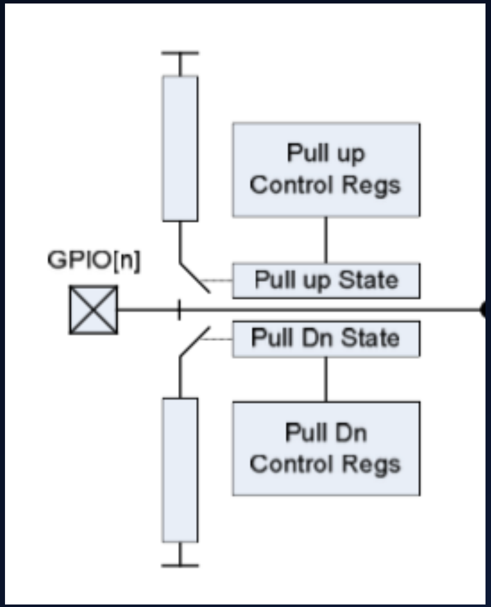
Lets look at it.

Demo

Analog

Other Future Possibilities!

- Attacking pull-up and pull-down resistors in I/O interfaces
- What if we disable them?
- Remotely manipulate the I/O via a powerful electromagnetic field!



Never trust your inputs!

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

Race to the Bottom

As soon as security is introduced at some layer of computer or network architecture abstraction, the attackers are going one layer down.

In the hacking community it is called Race-to-the-Bottom

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?

Looking for more…

Attend our talk at DigitalBond S4x17, Miami, USA

Everything that has a beginning has an end.

The Matrix Revolutions.

Contact:

a.abbasi@utwente.nl @bl4ckic3

@m4ji_d