RUTGERS

THE STATE UNIVERSITY
OF NEW JERSEY

Leveraging Physical Models for Attacking and Defending PLCs

Luis Garcia

4N6 Cyber Security & Forensics Research Lab

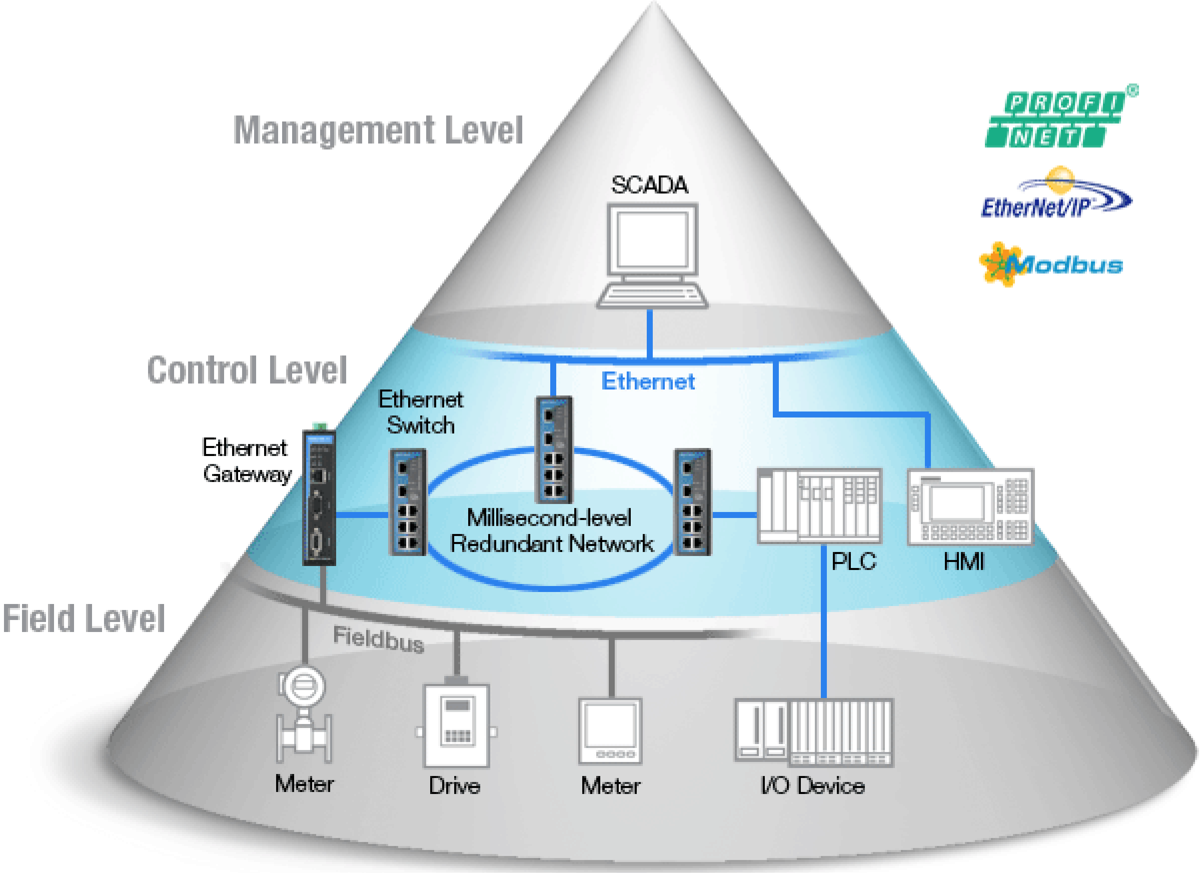
ECE Department

Rutgers University

Outline

- Background
- Harvey: Model-Aware Rootkit
  - System Model
  - Physics-Awareness
  - Implementation and Evaluation
- Device-Oriented Verification of CPS
- Conclusions

Programmable Logic Controllers (PLCs) and Industrial Control Systems (ICSs)



PROFINET

EtherNet/IP

Modbus

Credit: MOXA

What is a Programmable Logic Controller(PLC)?

- The interface between cyber and physical components in many CPS applications

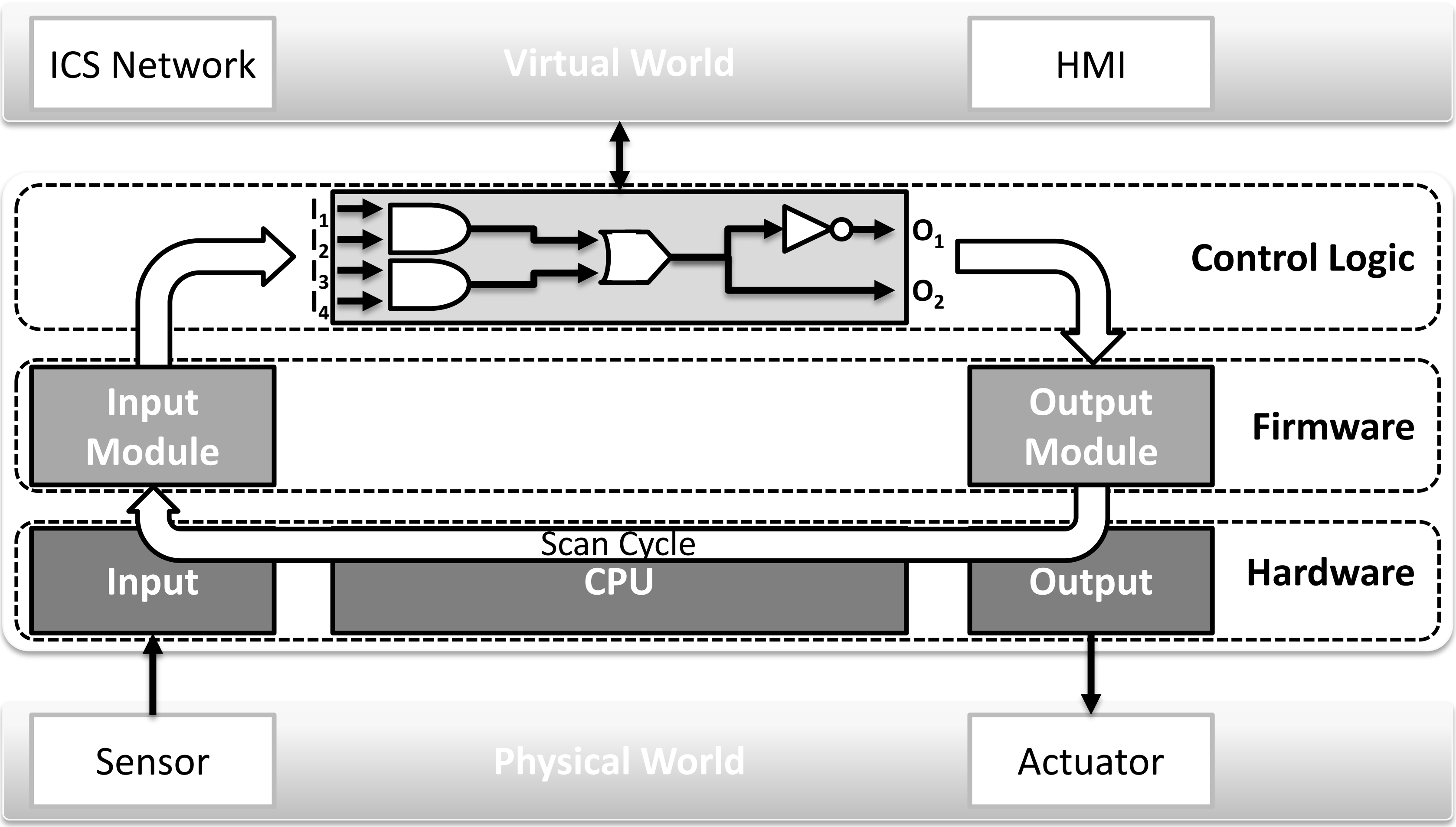
What is a Programmable Logic Controller(PLC)?

- The interface between cyber and physical components in many CPS applications
- Contain simple logic code that is easy to verify

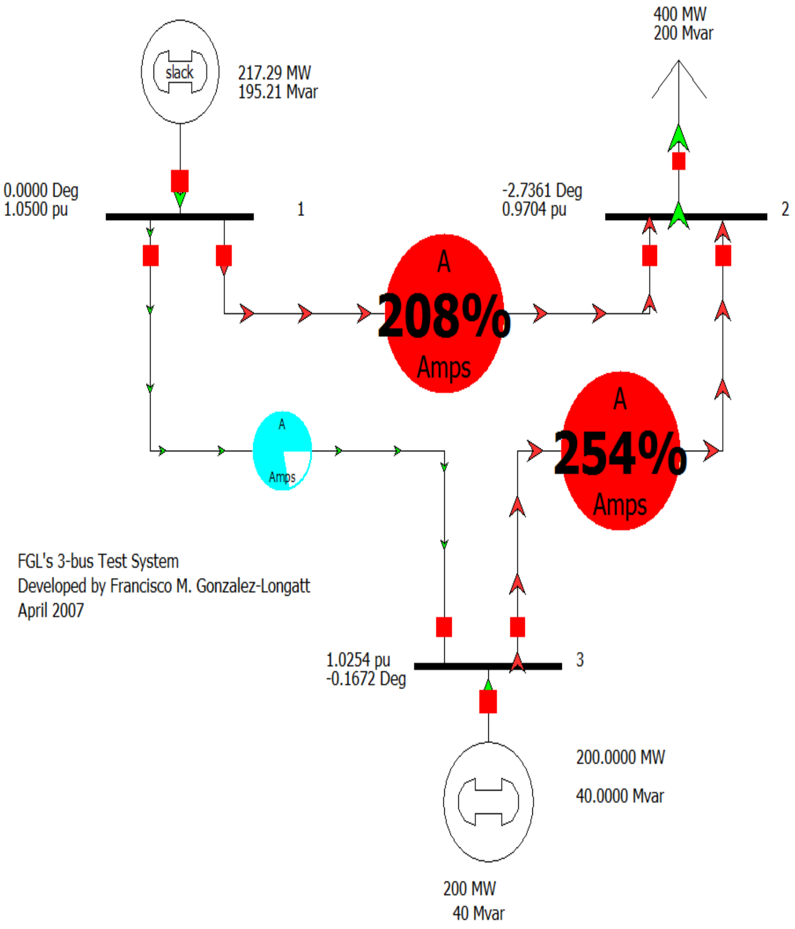
What is a Programmable Logic Controller(PLC)?

- The interface between cyber and physical components in many CPS applications
- Contain simple logic code that is easy to verify
- Typically the target in CPS attacks
  - E.g., Stuxnet

PLC Architecture

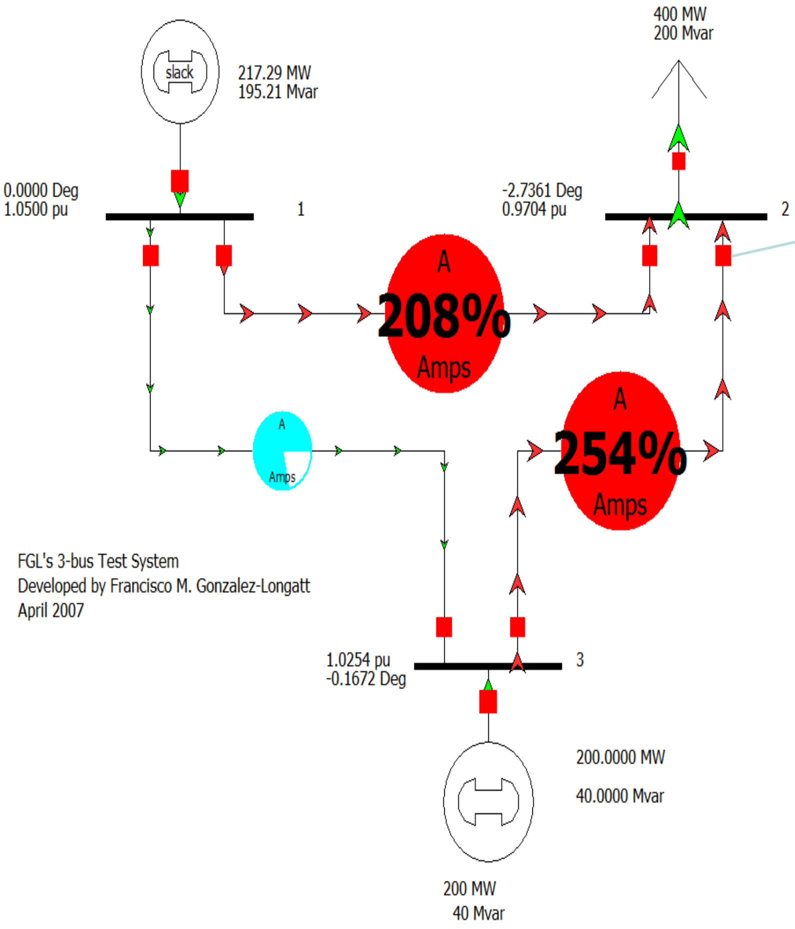


Example Industrial Control System



Physical System: Power Grid Network

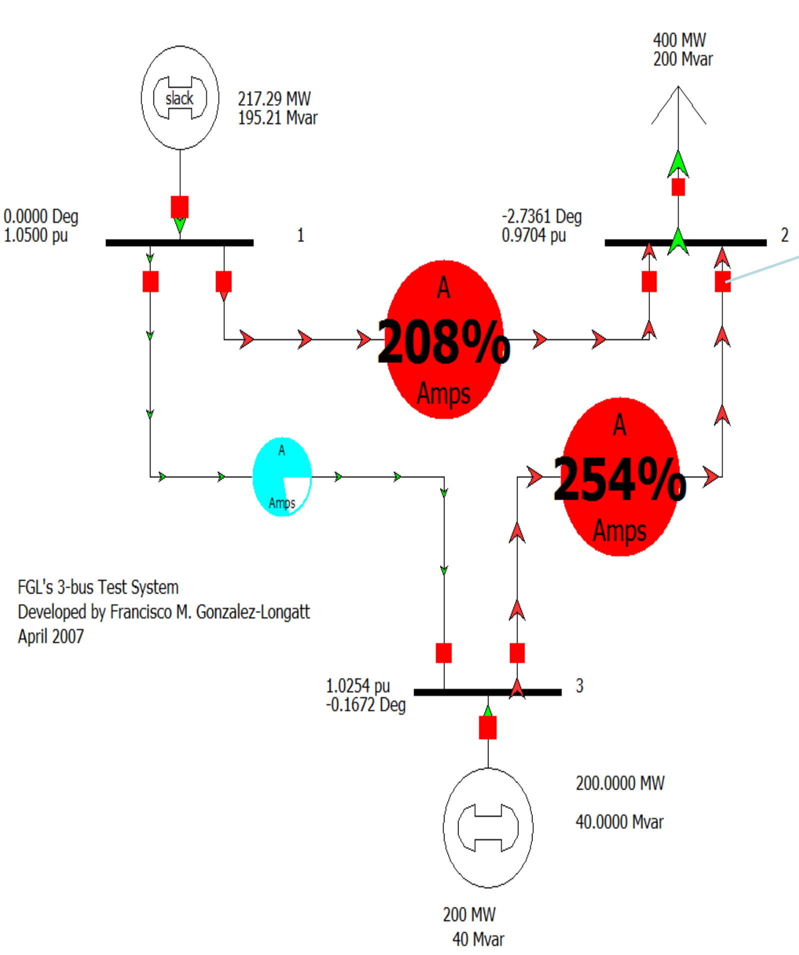
Example Industrial Control System



Physical System: Power Grid Network

In this example, the opening/closing of a circuit breaker in this scenario is controlled by a PLC

Example Industrial Control System

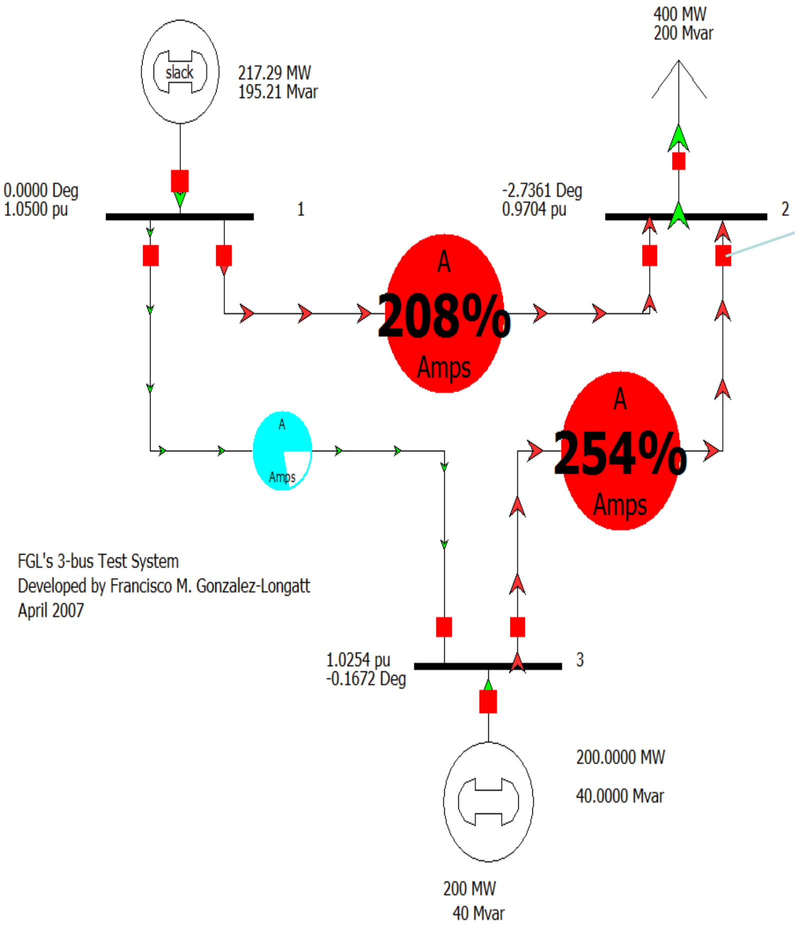


FGL's 3-bus Test System
Developed by Francisco M. Gonzalez-Longatt
April 2007

Physical System: Power Grid Network

An HMI System (in this case, a SCADA center)
May monitor the PLC values and send commands Accordingly.

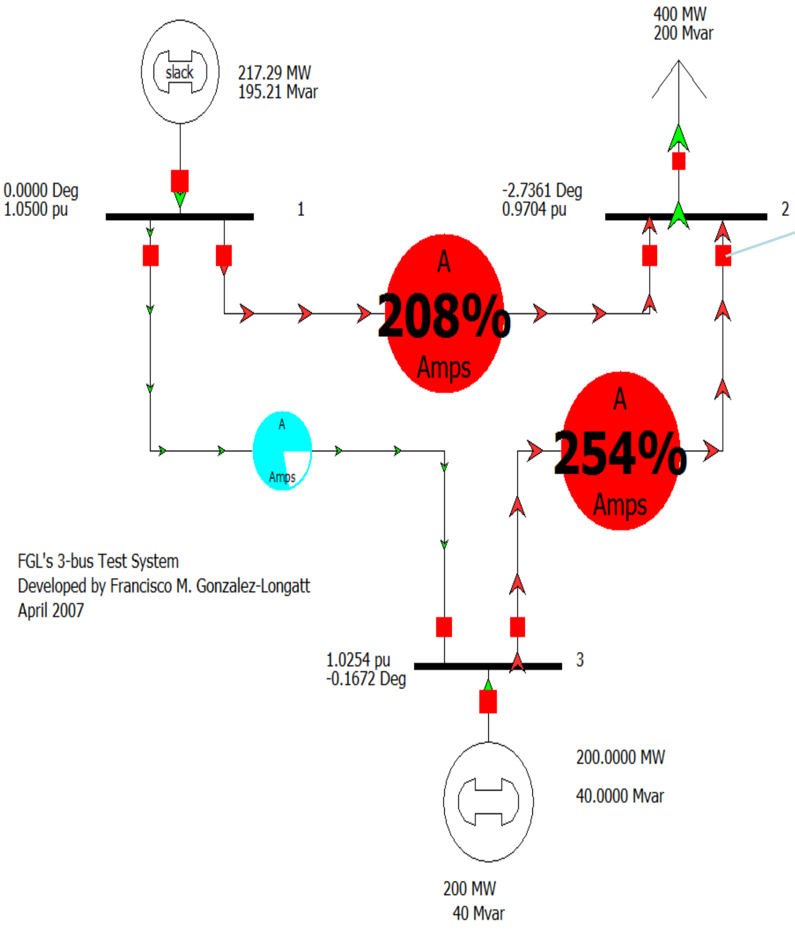
Example Industrial Control System



Physical System: Power Grid Network

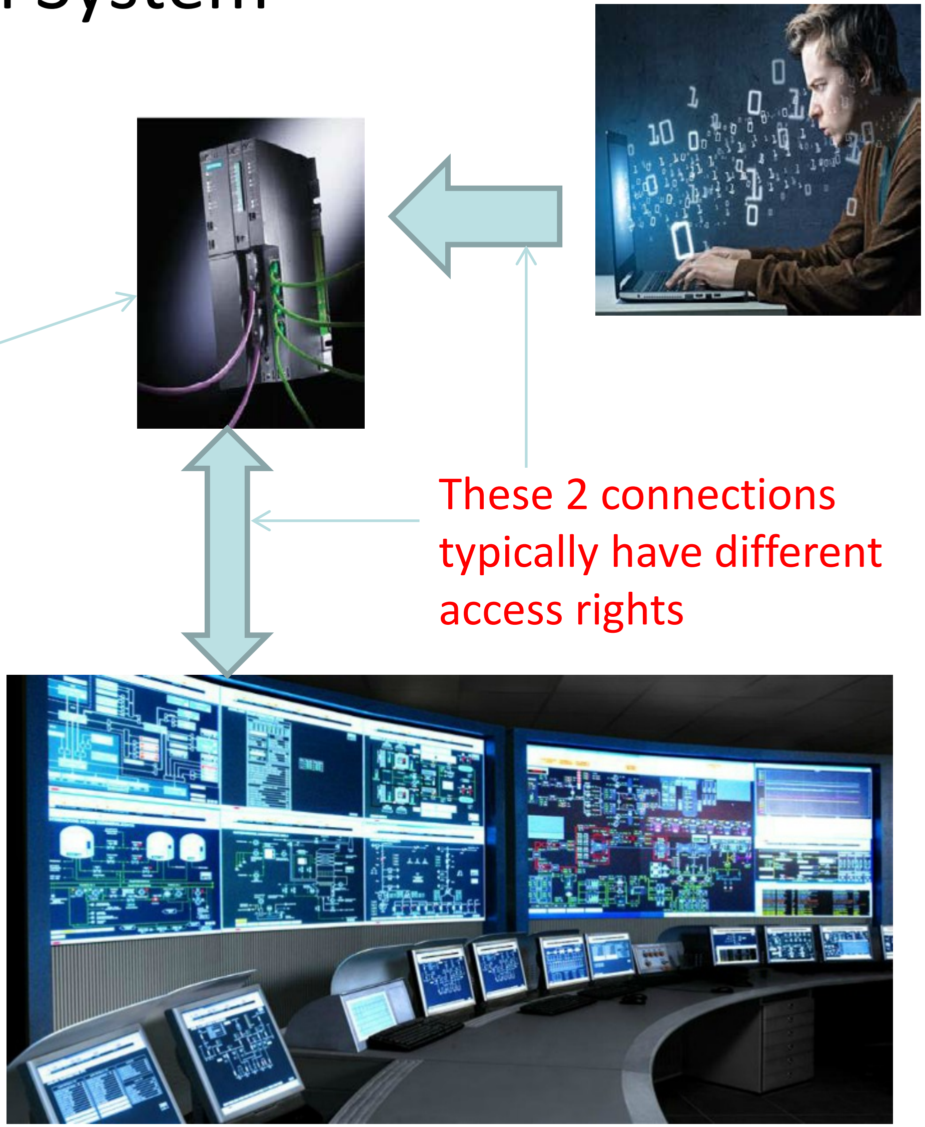
A programmer will be allowed to change The PLC configuration as well as the Control logic of the system

Example Industrial Control System



FGL's 3-bus Test System
Developed by Francisco M. Gonzalez-Longatt
April 2007

Physical System: Power Grid Network

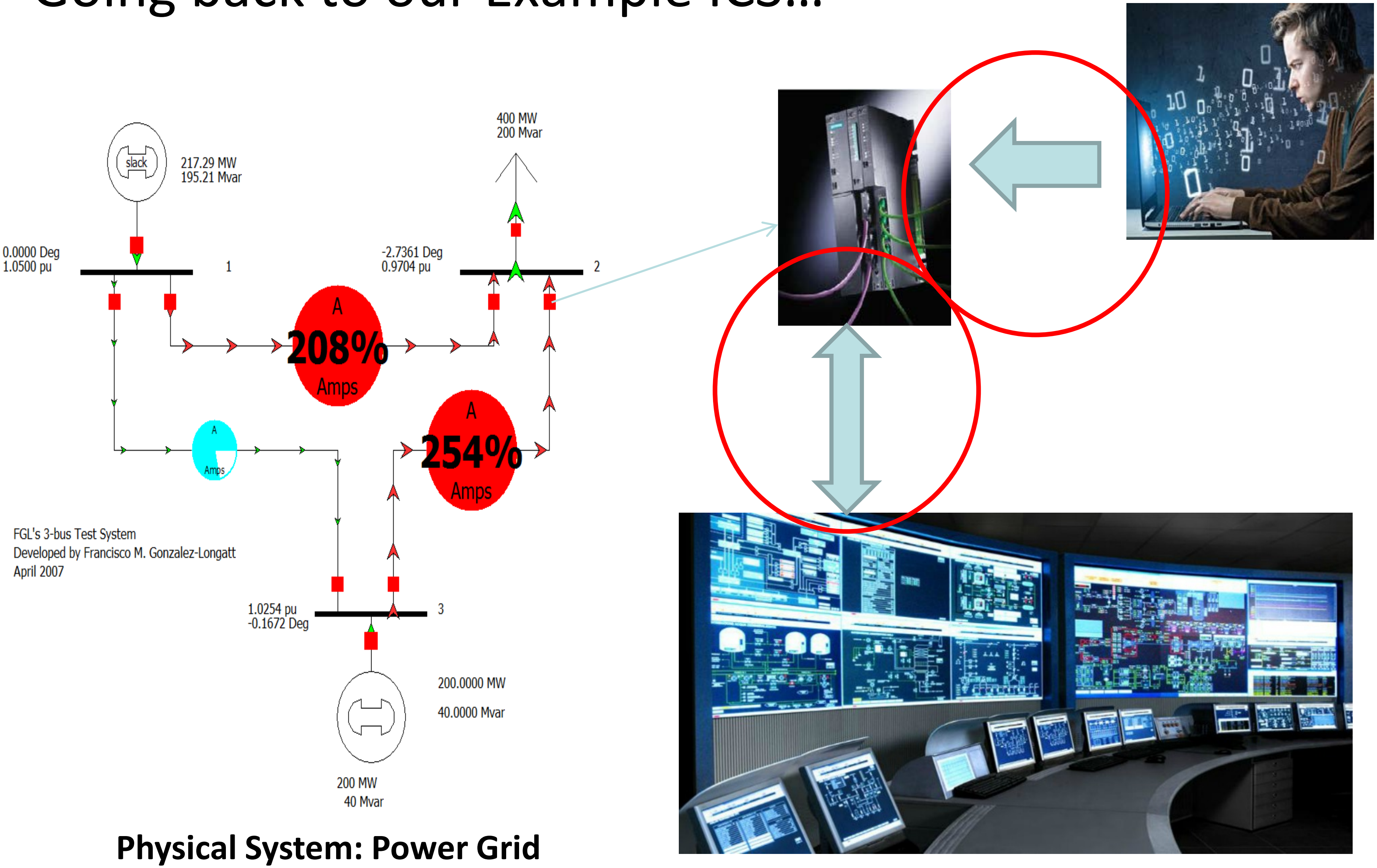


These 2 connections typically have different access rights

Previous Attacks on PLC’s: Stuxnet

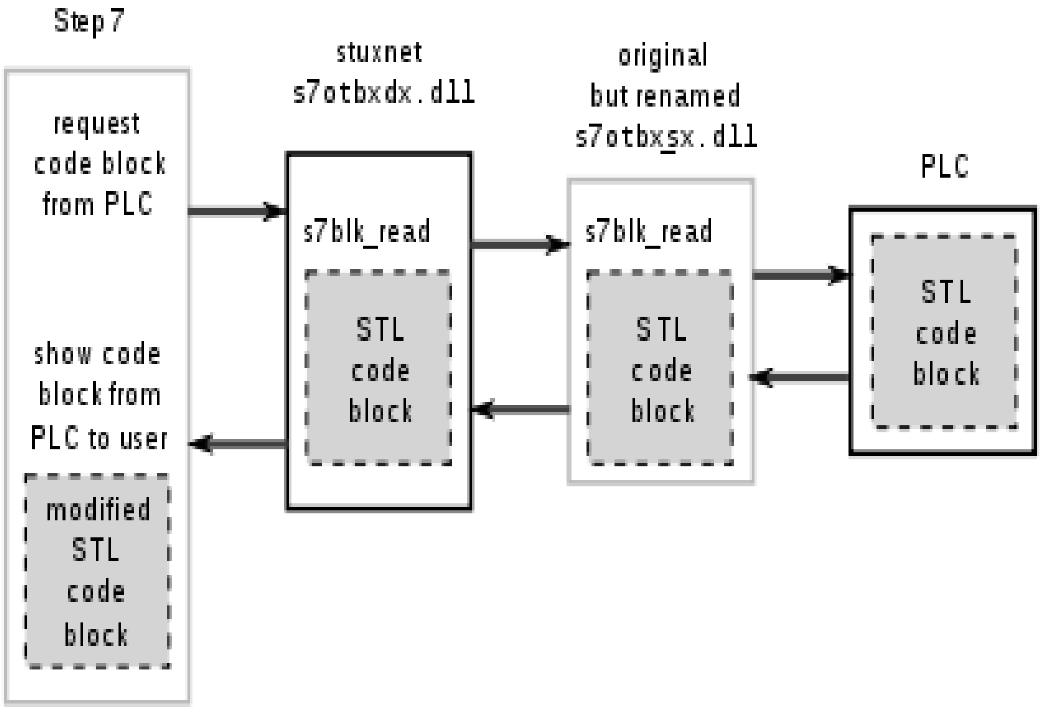
- Advanced malware worm that attacked Siemens S7 PLC’s and WinCC systems
- Targeted high frequency drives controlling centrifuges
- Caused billions of dollars in damages

Going back to our Example ICS…

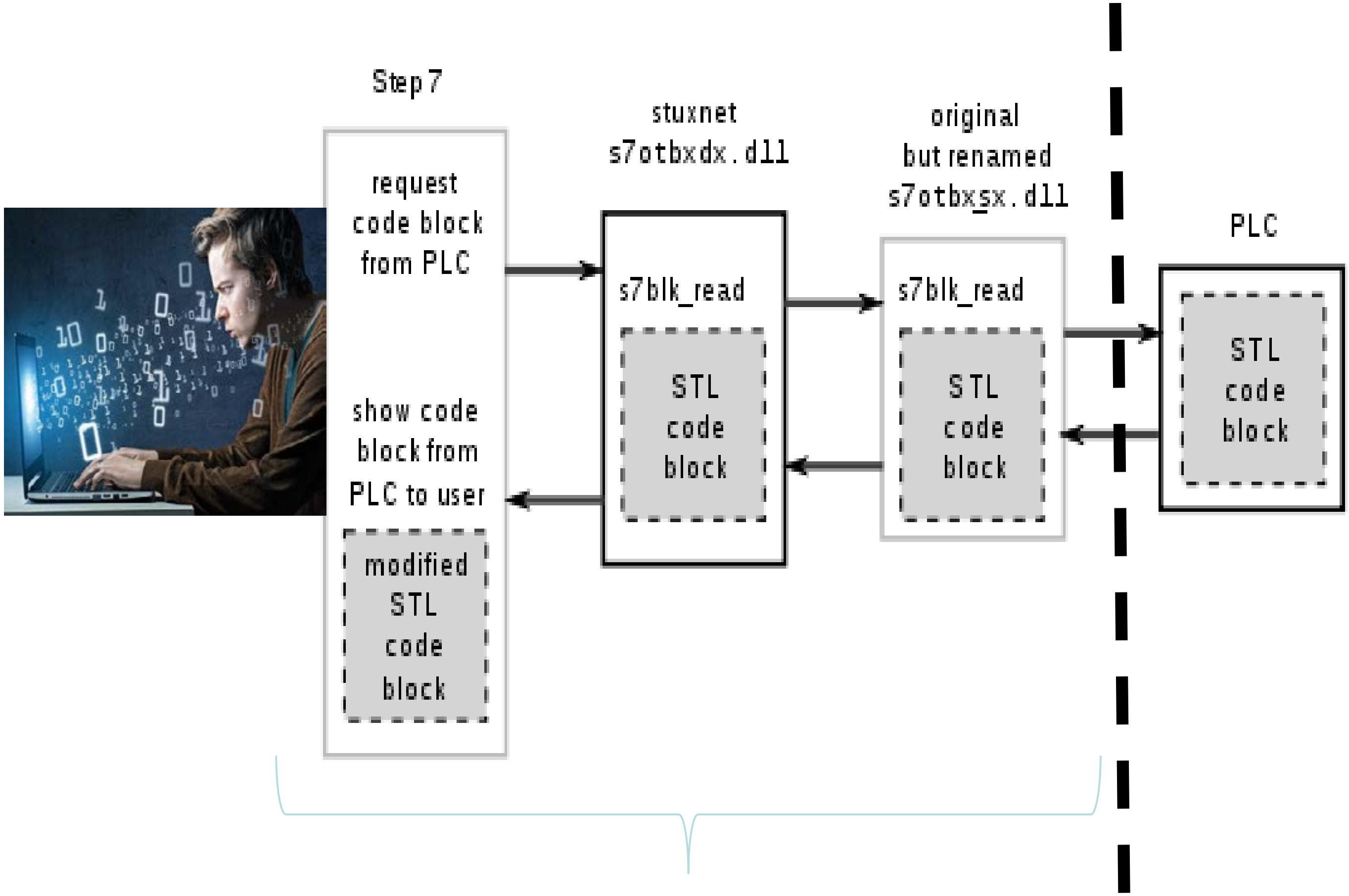


Physical System: Power Grid Network

Stuxnet's PLC Attack Overview

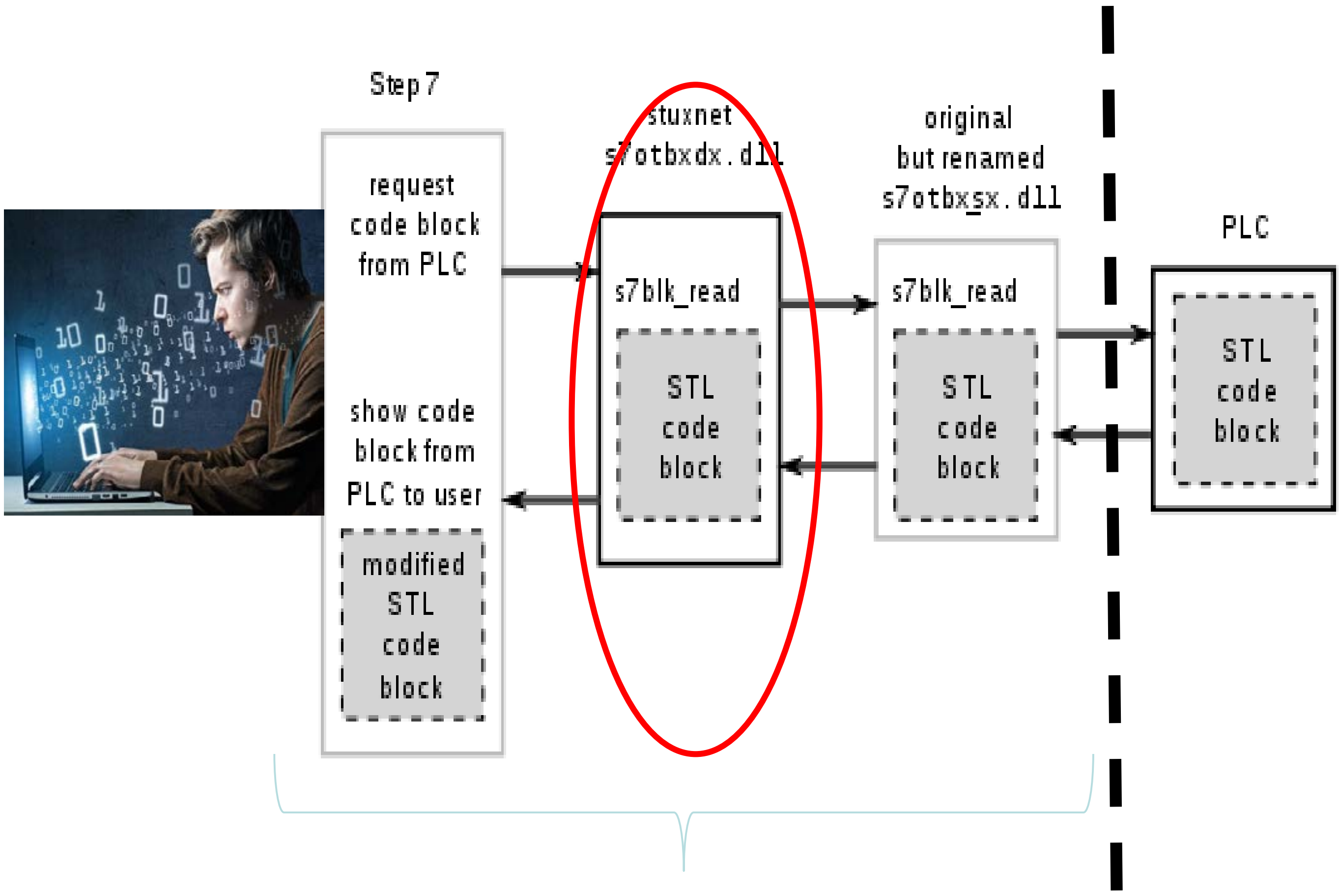


Stuxnet's PLC Attack Overview



Programmer's PC

Stuxnet's PLC Attack Overview



Step 7

request code block from PLC

show code block from PLC to user

modified STL code block

stuxnet s7otbxdx.dll

s7blk_read

STL code block

original but renamed s7otbxsx.dll

s7blk_read

STL code block

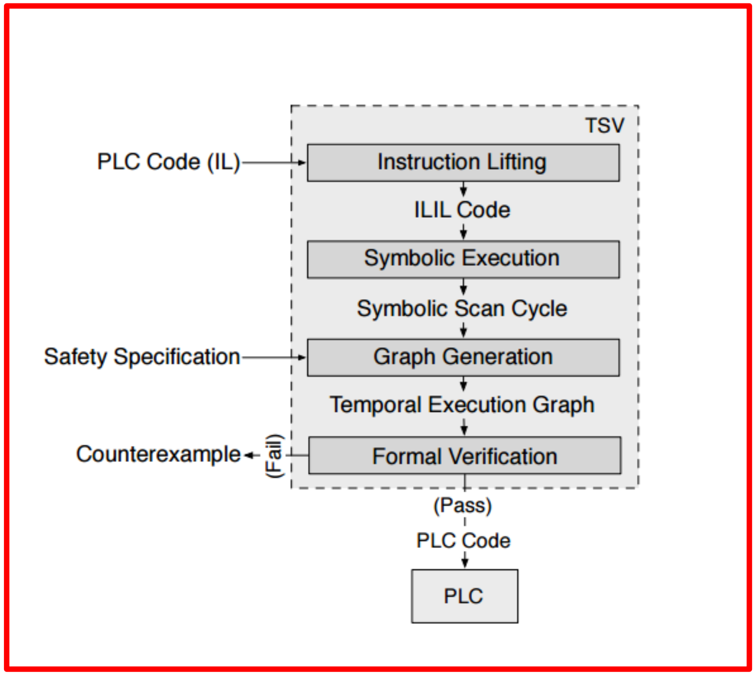
PLC

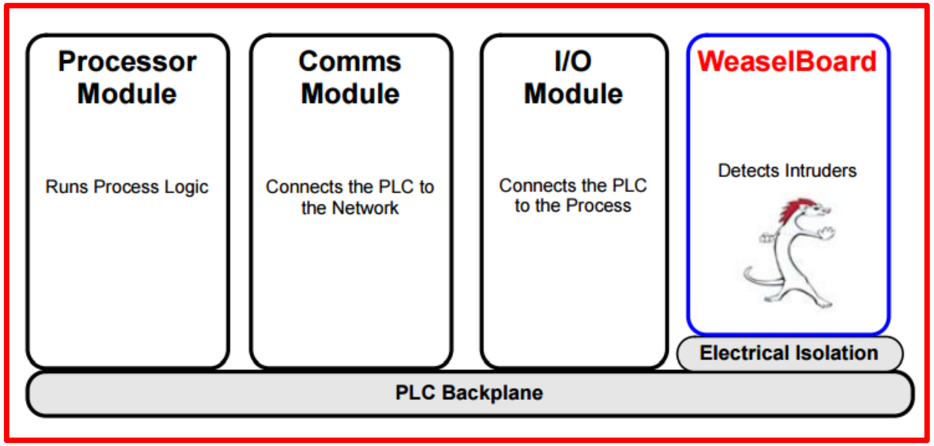
STL code block

Programmer's PC

Prior Efforts to Mitigate Attacks like Stuxnet

- Typically offline, passive solutions
- External solutions for PLCs





RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit

Luis Garcia, Saman Zonouz
ECE Department
Rutgers University

Ferdinand Brasser, Ahmad-Reza Sadeghi
System Security Lab
Technische Universität Darmstadt

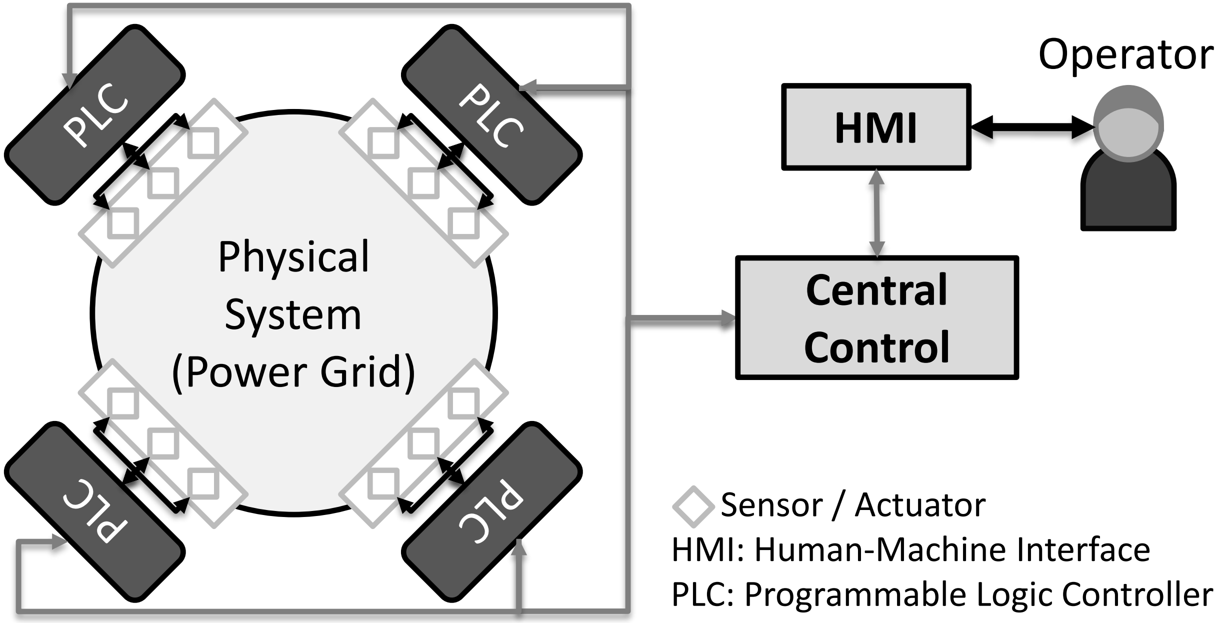
Mehmet H. Cintuglu, Osama Mohammed
ECE Department
Florida International University

NDSS 2017

Harvey: Model-Aware Rootkit

- A rootkit that takes into account the physical topology of the ICS
- Model
  - Uses physical models to optimize control commands for an adversarial objective function
- PLC infection: compromising the PLC’s firmware
  - Utilize the firmware update mechanism to replace firmware over the network
  - Local firmware modifications, e.g., SD card or JTAG implantation
  - Run-time attacks, e.g., network exploits or remote code execution vulnerabilities (FrostyURL)

System Model



◇ Sensor / Actuator
HMI: Human-Machine Interface
PLC: Programmable Logic Controller

Adversary Model

- Stealthiness

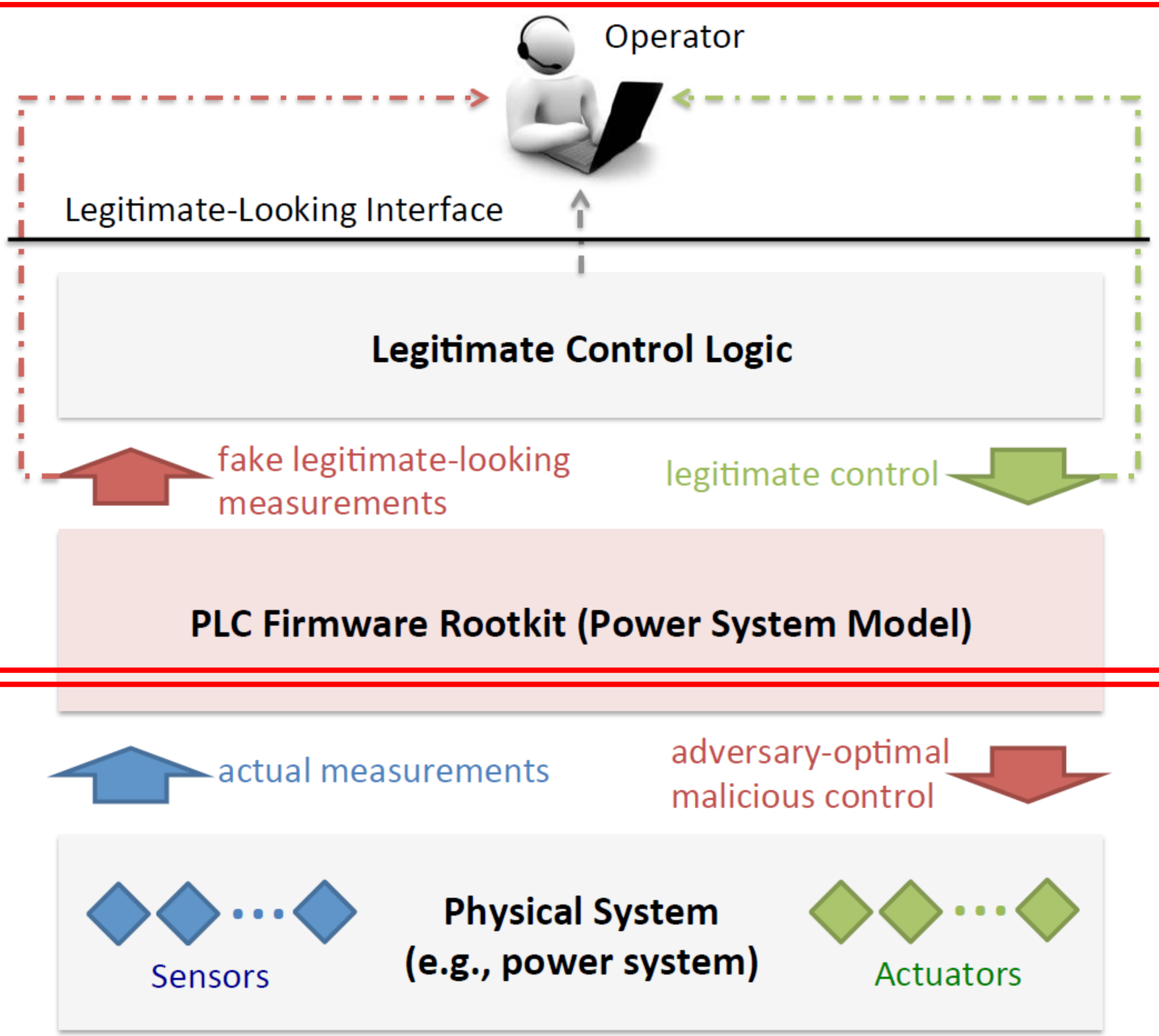
Adversary Model

- Stealthiness
- PLC-only attack

Adversary Model

- Stealthiness
- PLC-only attack
- Physical model extraction

Physics-Awareness: 2-Way Data Manipulation



Operator

Legitimate-Looking Interface

Legitimate Control Logic

fake legitimate-looking measurements

legitimate control

PLC Firmware Rootkit (Power System Model)

actual measurements

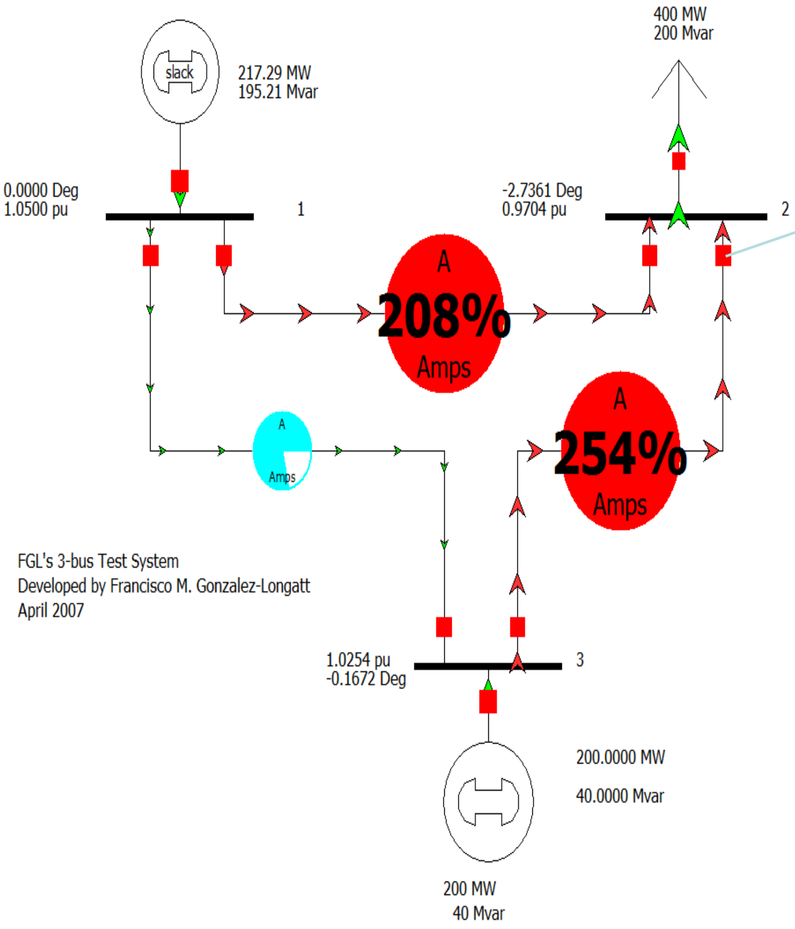
adversary-optimal malicious control

Physical System (e.g., power system)

Sensors

Actuators

Back to ICS Example…

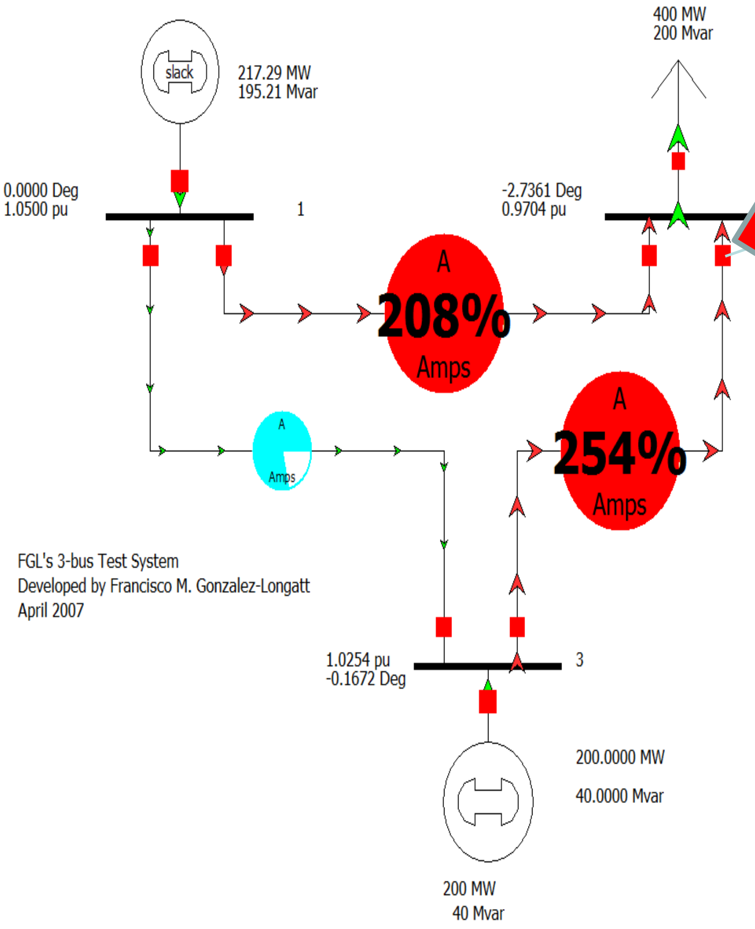


Physical System: Power Grid Network

Stuxnet attacked these two communication channels

Back to ICS Example…

Our attack focuses on the interface Between the PLC and it's own I/O Modules (i.e., the interface between The PLC and the underly physical System)



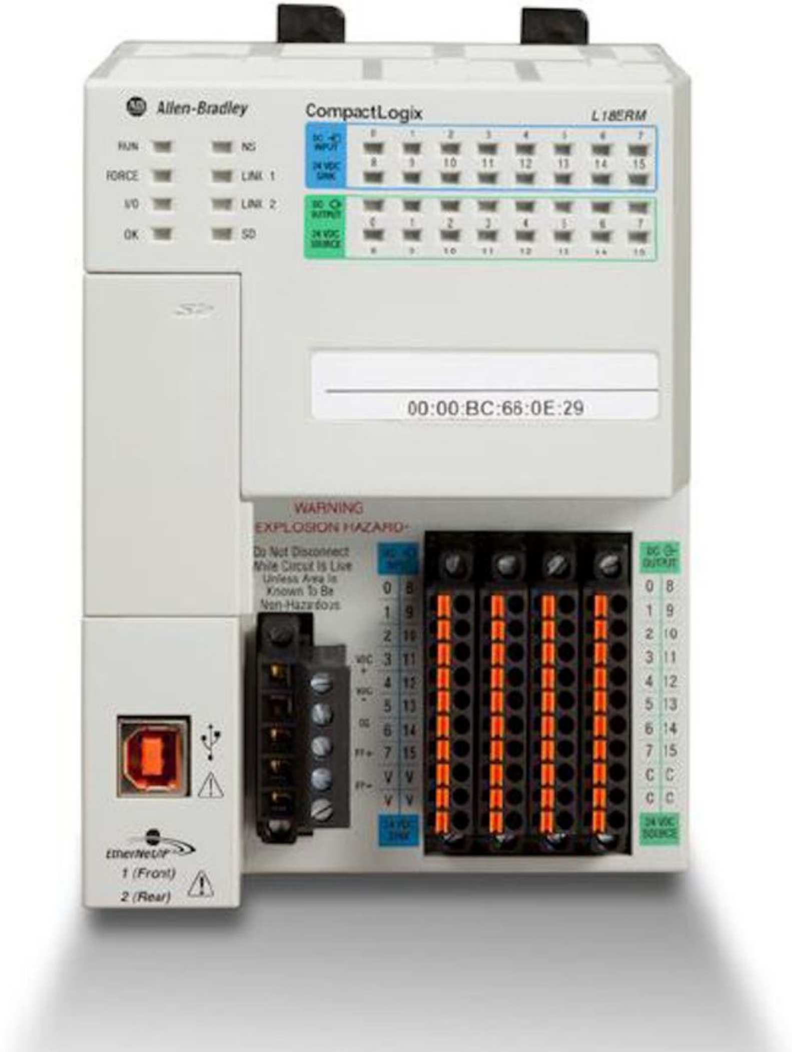
Physical System: Power Grid Network

Stuxnet attacked these two communication channels

Implementing Harvey: Device Selection and Specification

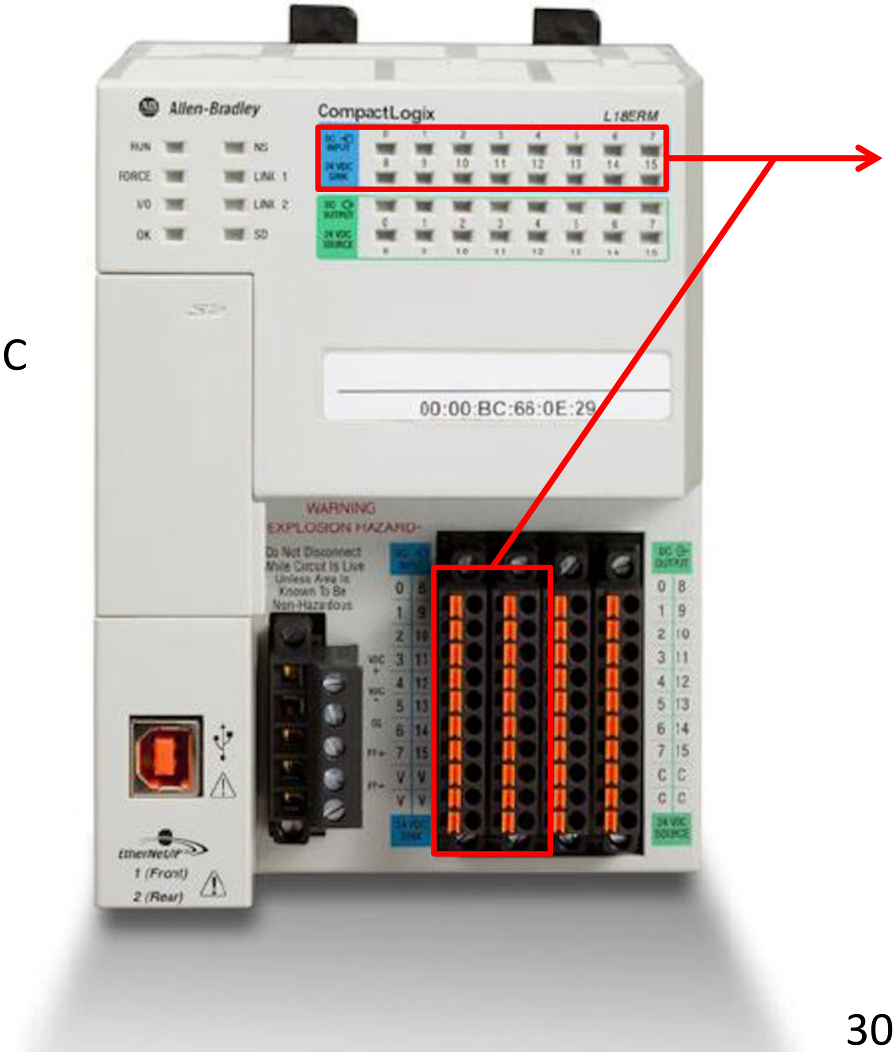
- Allen Bradley CompactLogix L1
- Based on Texas Instruments Stellaris LM3S2793 Microcontroller
  - Arm Cortex-M3 ISA

CompactLogix L1 PLC



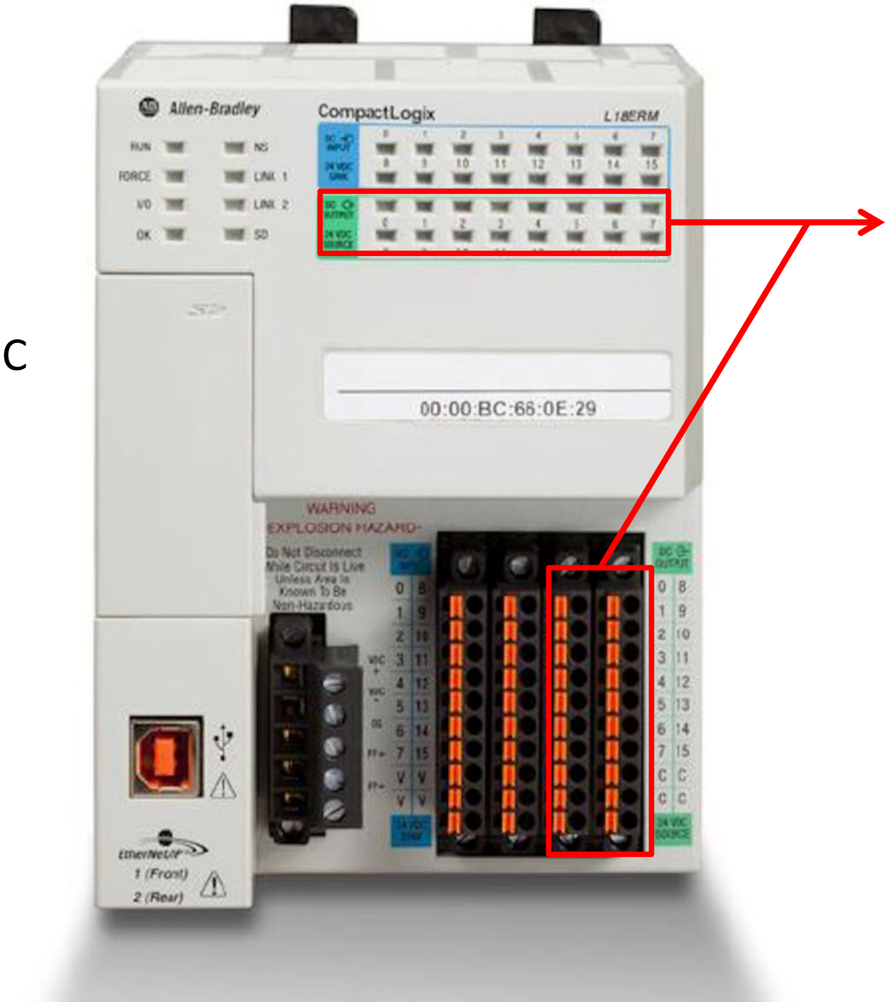
CompactLogix L1 PLC

- High Value (1) ~ 24 V DC
- Low Value (0) ~ 8 V DC



CompactLogix L1 PLC

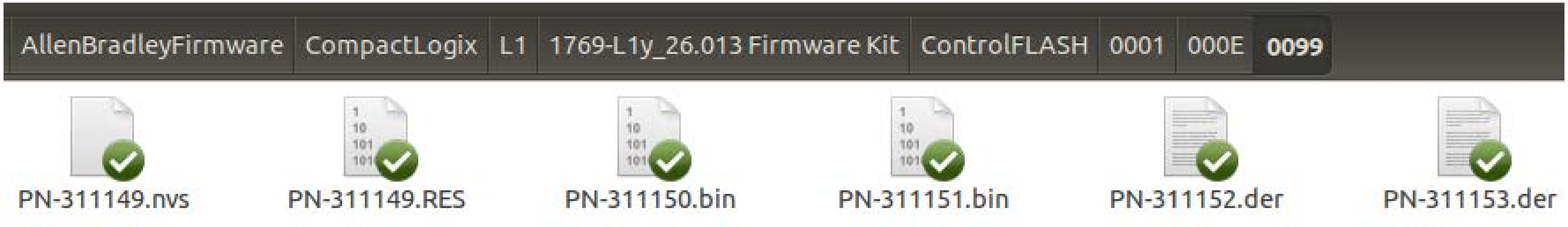
- High Value (1) ~ 24 V DC
- Low Value (0) ~ 8 V DC



16 Bit Digital Output

Analyzing the CompactLogix L1 Firmware Update Files

- There have been prior works that reverse engineer the firmware update procedure of different Allen Bradley PLCs
  - Although these works simply bricked the PLCs, they did provide a means of updating the firmware
- Although we spent a lot of time analyzing the firmware update files, we eventually found that analyzing the dumped memory was more efficient for our goals



PN-311149.nvs

PN-311149.RES

PN-311150.bin

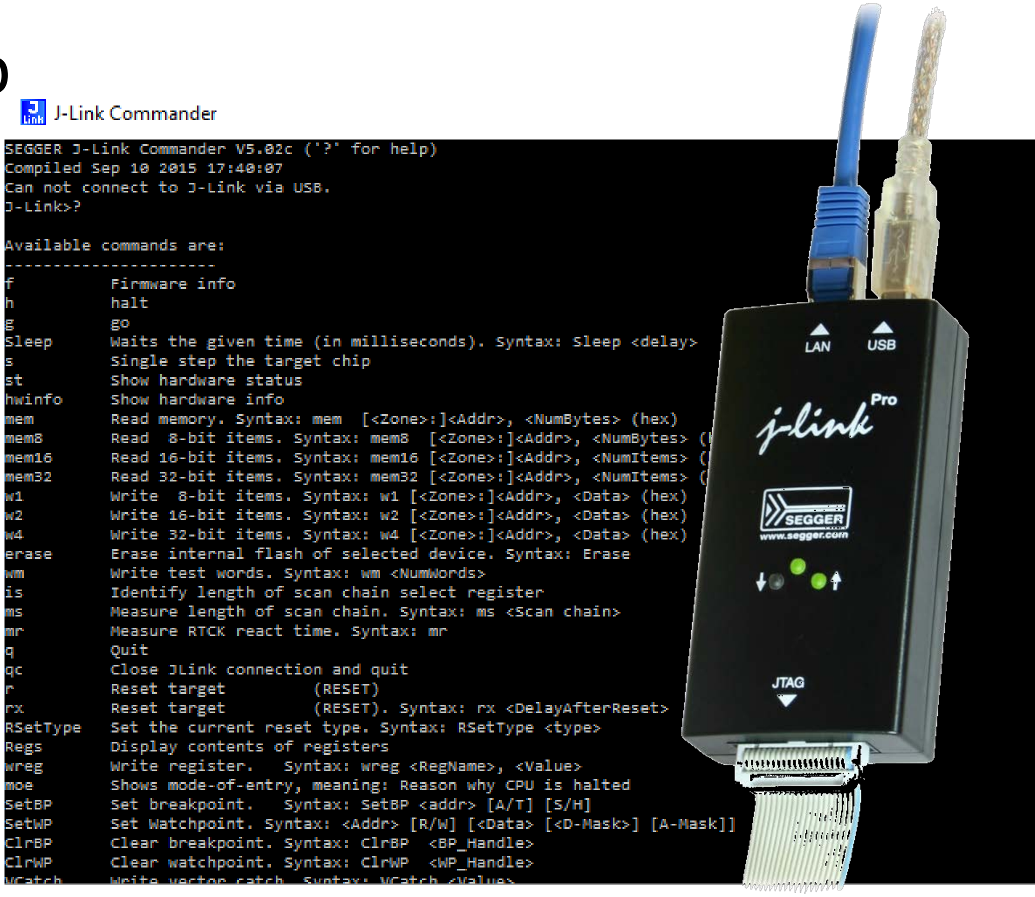
PN-311151.bin

PN-311152.der

PN-311153.der

JTAG Debugging

- Joint Test Action Group (JTAG) standard was designed to assist with device, board, and system testing, diagnosis and fault isolation
- Usually one of the first approaches used for reverse engineering efforts

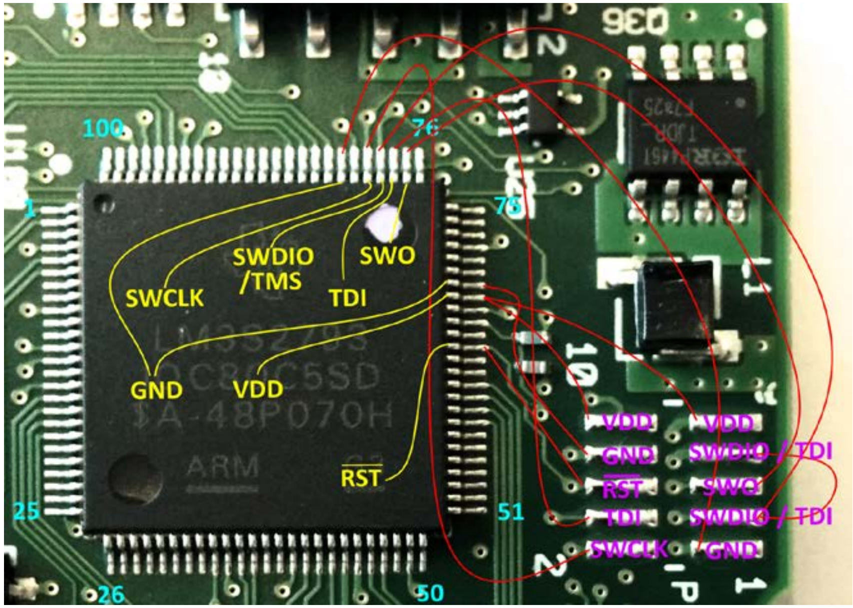


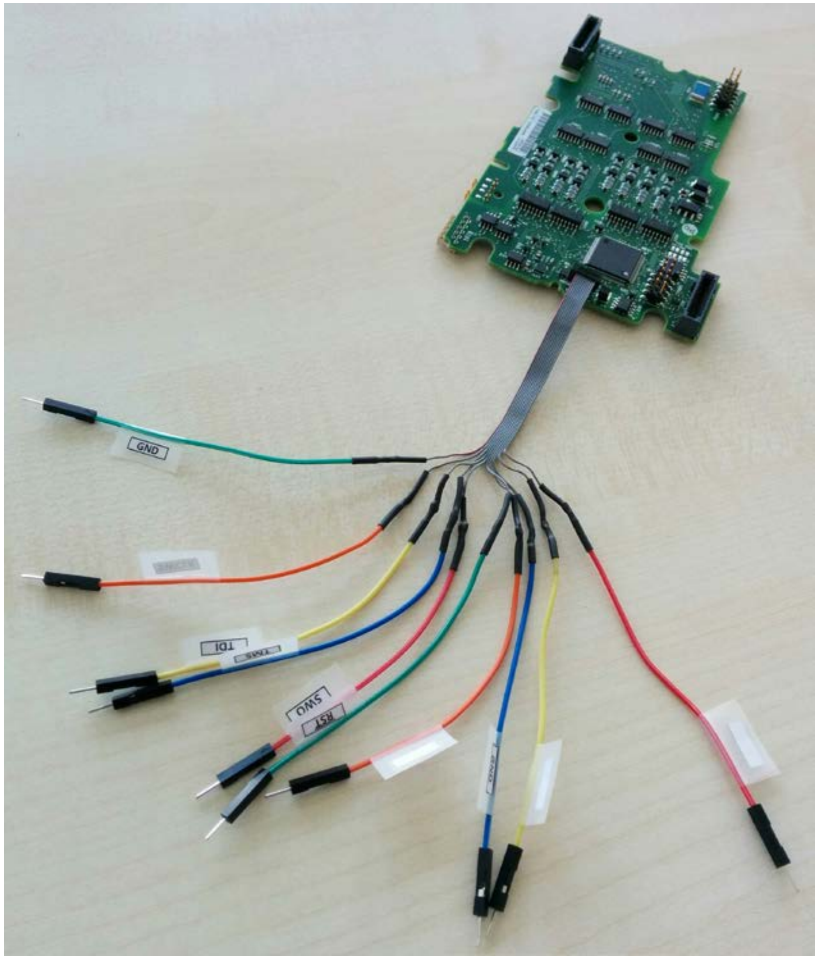
J-Link Commander

```
SEGGER J-Link Commander V5.02c ('?' for help)
Compiled Sep 10 2015 17:40:07
Can not connect to J-Link via USB.
J-Link>?

Available commands are:
----------------------
f          Firmware info
h          halt
g          go
Sleep      Waits the given time (in milliseconds). Syntax: Sleep <delay>
s          Single step the target chip
st         Show hardware status
hwinfo     Show hardware info
mem        Read memory. Syntax: mem  [<Zone>:]<Addr>, <NumBytes> (hex)
mem8       Read  8-bit items. Syntax: mem8  [<Zone>:]<Addr>, <NumBytes> (
mem16      Read 16-bit items. Syntax: mem16 [<Zone>:]<Addr>, <NumItems> (
mem32      Read 32-bit items. Syntax: mem32 [<Zone>:]<Addr>, <NumItems> (
w1         Write  8-bit items. Syntax: w1 [<Zone>:]<Addr>, <Data> (hex)
w2         Write 16-bit items. Syntax: w2 [<Zone>:]<Addr>, <Data> (hex)
w4         Write 32-bit items. Syntax: w4 [<Zone>:]<Addr>, <Data> (hex)
erase      Erase internal flash of selected device. Syntax: Erase
wm         Write test words. Syntax: wm <NumWords>
is         Identify length of scan chain select register
ms         Measure length of scan chain. Syntax: ms <Scan chain>
mr         Measure RTCK react time. Syntax: mr
q          Quit
qc         Close JLink connection and quit
r          Reset target         (RESET)
rx         Reset target         (RESET). Syntax: rx <DelayAfterReset>
RSetType   Set the current reset type. Syntax: RSetType <type>
Regs       Display contents of registers
wreg       Write register.   Syntax: wreg <RegName>, <Value>
moe        Shows mode-of-entry, meaning: Reason why CPU is halted
SetBP      Set breakpoint.   Syntax: SetBP <addr> [A/T] [S/H]
SetWP      Set Watchpoint. Syntax: <Addr> [R/W] [<Data> [<D-Mask>] [A-Mask]]
ClrBP      Clear breakpoint. Syntax: ClrBP  <BP_Handle>
ClrWP      Clear watchpoint. Syntax: ClrWP  <WP_Handle>
VCatch     Write vector catch. Syntax: VCatch <Value>
```

Memory Analysis with JTAG





Memory Analysis with JTAG

- Used JTAG interface to dump memory for code disassembly
- Used TI Stellaris LM3S2793 data sheet to find memory layout and built-in ROM functions

TEXAS INSTRUMENTS-PRODUCTION DATA

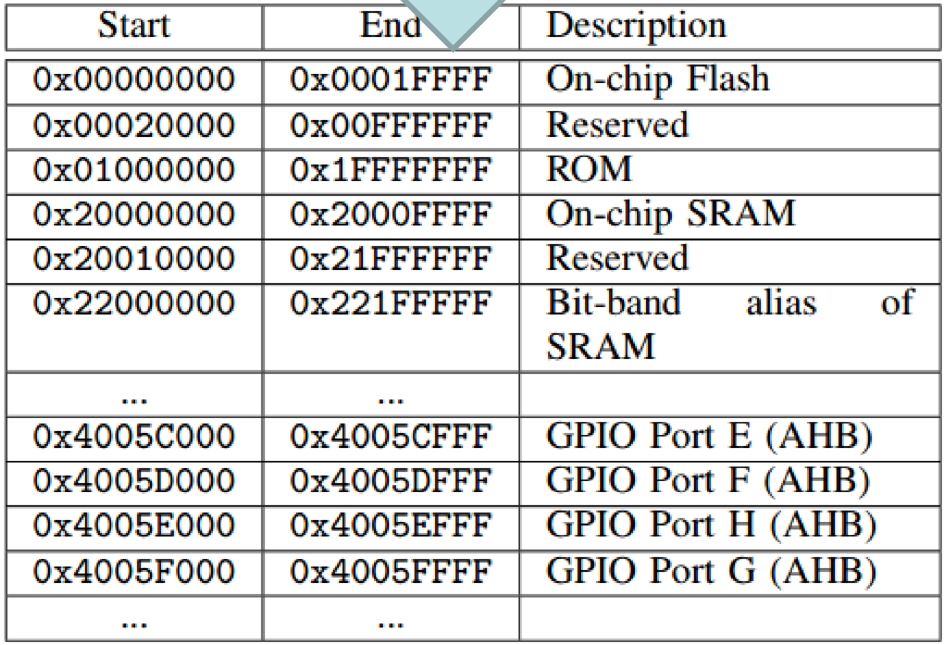
Texas Instruments

Stellaris® LM3S2793 Microcontroller

DATA SHEET

DS-LM3S2793-15852.2743
SPMS056O

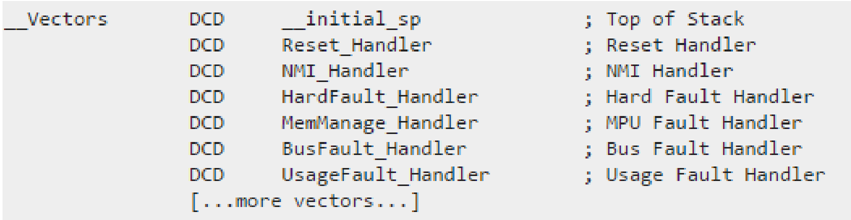
Copyright © 2007-2014
Texas Instruments Incorporated



Start	End	Description
0x00000000	0x0001FFFF	On-chip Flash
0x00020000	0x00FFFFFF	Reserved
0x01000000	0x1FFFFFFF	ROM
0x20000000	0x2000FFFF	On-chip SRAM
0x20010000	0x21FFFFFF	Reserved
0x22000000	0x221FFFFF	Bit-band alias of SRAM
...	...	
0x4005C000	0x4005CFFF	GPIO Port E (AHB)
0x4005D000	0x4005DFFF	GPIO Port F (AHB)
0x4005E000	0x4005EFFF	GPIO Port H (AHB)
0x4005F000	0x4005FFFF	GPIO Port G (AHB)
...	...	

Static Memory Analysis

- We followed the boot sequence to determine the control flow of the program
- We used the notion that for Cortex-M3 processors, the Reset Handler is located at address 0x0000004



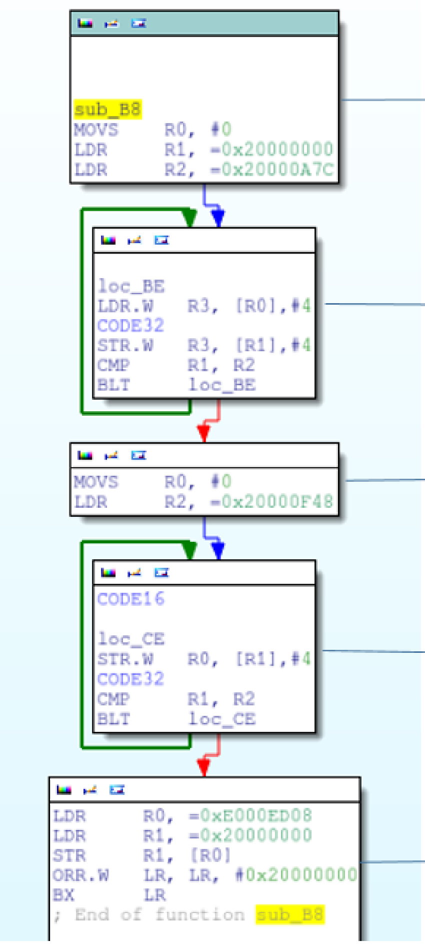
```
__Vectors       DCD     __initial_sp              ; Top of Stack
                DCD     Reset_Handler             ; Reset Handler
                DCD     NMI_Handler               ; NMI Handler
                DCD     HardFault_Handler         ; Hard Fault Handler
                DCD     MemManage_Handler         ; MPU Fault Handler
                DCD     BusFault_Handler          ; Bus Fault Handler
                DCD     UsageFault_Handler        ; Usage Fault Handler
                [...more vectors...]
```

Following the Boot Sequence with IDA Pro

Flash Memory

0x00000004 Reset Address = 0x000000E3

```
0x000000E3  BL      sub_B8
            BL      sub_51E
            LDR     R0, loc_120
            LDR     R1, =0xE000ED08
            STR     R0, [R1]
            LDR     R1, [R0]
            MOV     SP, R1
            LDR     R0, [R0,#(loc_4004 - 0x4000)]
            BX      R0 ; loc_E378
```



```
sub_B8
MOVS    R0, #0
LDR     R1, =0x20000000
LDR     R2, =0x20000A7C

loc_BE
LDR.W   R3, [R0],#4
CODE32
STR.W   R3, [R1],#4
CMP     R1, R2
BLT     loc_BE

MOVS    R0, #0
LDR     R2, =0x20000F48

CODE16

loc_CE
STR.W   R0, [R1],#4
CODE32
CMP     R1, R2
BLT     loc_CE

LDR     R0, =0xE000ED08
LDR     R1, =0x20000000
STR     R1, [R0]
ORR.W   LR, LR, #0x20000000
BX      LR
; End of function sub_B8
```

0x00000000 = Address of SP
0x20000000 = Start of SRAM
0x20000A7C = ???

Loc_BE:
R3 <= *(R0) + 4 = *(0)+4 = 20000B3C +4 = 20000B40
*(R1)+4 =20000B3C+4 =20000B40 <= 20000B40
R1 < R2 ? == 20000000 < 20000A7C --> Always false?

R0 = 0
R2 = 0x20000F48

*(R1)+4=20000B40 <= 0
R1 < R2? == 20000000 < 20000B40 -->Always false?

R0 = 0xE00ED08
R1 = 0x20000000
*(E00ED08) <= 0x20000000 (Cannot access E00ED08)
LR <= LR OR 0x20000000 == 200000E6
Branch to LR (200000E6)

- IDA Pro is a multi-processor disassembler and debugger
- We took the extracted firmware and disassembled it using IDA Pro

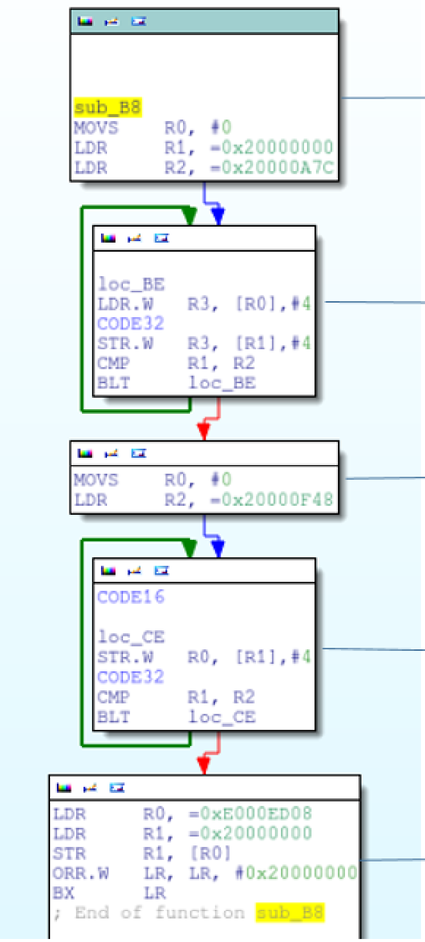
Following the Boot Sequence with IDA Pro

Flash Memory

0x00000004 Reset Address = 0x000000E3

```
0x000000E3 BL      sub_B8
           BL      sub_51E
           LDR     R0, loc_120
           LDR     R1, =0xE000ED08
           STR     R0, [R1]
           LDR     R1, [R0]
           MOV     SP, R1
           LDR     R0, [R0,#(loc_4004 - 0x4000)]
           BX      R0 ; loc_E378
```

- IDA Pro is a multi-processor disassembler and debugger
- We took the extracted firmware and disassembled it using IDA Pro



```
sub_B8
MOVS     R0, #0
LDR      R1, =0x20000000
LDR      R2, =0x20000A7C
```

0x00000000 = Address of SP
0x20000000 = Start of SRAM
0x20000A7C = ???

```
loc_BE
LDR.W    R3, [R0],#4
CODE32
STR.W    R3, [R1],#4
CMP      R1, R2
BLT      loc_BE
```

Loc_BE:
R3 <= *(R0) + 4 = *(0)+4 = 20000B3C +4 = 20000B40
*(R1)+4 =20000B3C+4 =20000B40 <= 20000B40
R1 < R2 ? == 20000000 < 20000A7C --> Always false?

```
MOVS     R0, #0
LDR      R2, =0x20000F48
```

R0 = 0
R2 = 0x20000F48

```
CODE16

loc_CE
STR.W    R0, [R1],#4
CODE32
CMP      R1, R2
BLT      loc_CE
```

*(R1)+4=20000B40 <= 0
R1 < R2? == 20000000 < 20000B40 -->Always false?

```
LDR      R0, =0xE000ED08
LDR      R1, =0x20000000
STR      R1, [R0]
ORR.W    LR, LR, #0x20000000
BX       LR
; End of function sub_B8
```

R0 = 0xE00ED08
R1 = 0x20000000
*(E00ED08) <= 0x20000000 (Cannot access E00ED08)
LR <= LR OR 0x20000000 == 200000E6
Branch to LR (200000E6)

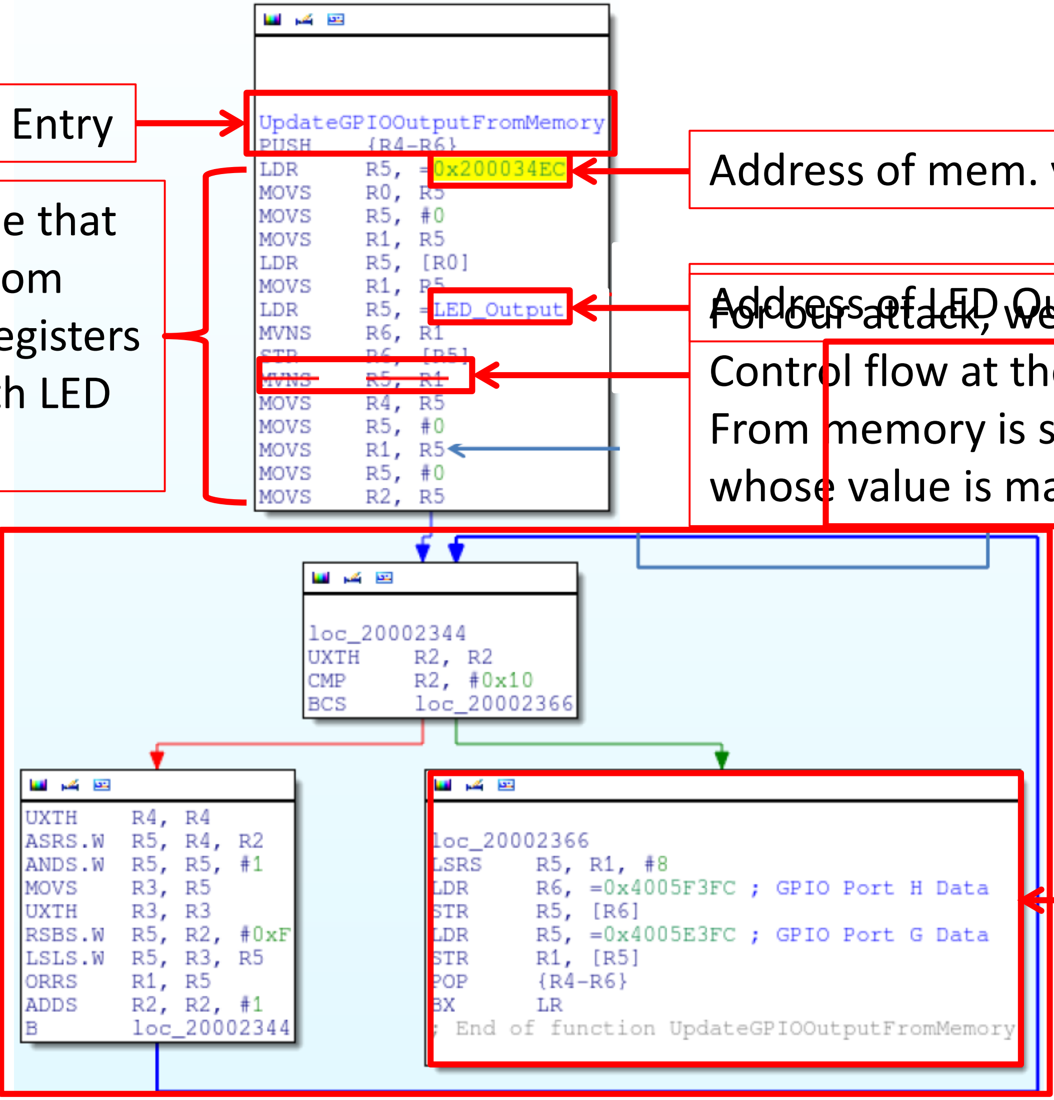
Static/Dynamic Analysis for I/O Interception

- Couldn’t analyze every possible path to determine I/O interception point
- Halted the CPU (via JTAG) during slow boot-up LED sequence and stepped through execution to see how LEDs values were being updated
  - Memory addresses of LED values led us to ISR’s responsible for forwarding GPIO values to and from PLCs

Static/Dynamic Analysis for I/O Interception

- Couldn’t analyze every possible path to determine I/O interception point
- Halted the CPU (via JTAG) during slow boot-up LED sequence and stepped through execution to see how LEDs values were being updated
  - Memory addresses of LED values led us to ISR’s responsible for forwarding GPIO values to and from PLCs

Modified GPIO-Output Update ISR



```
UpdateGPIOOutputFromMemory
PUSH    {R4-R6}
LDR     R5, =0x200034EC
MOVS    R0, R5
MOVS    R5, #0
MOVS    R1, R5
LDR     R5, [R0]
MOVS    R1, R5
LDR     R5, =LED_Output
MVNS    R6, R1
STR     R6, [R5]
~~MVNS    R5, R1~~
MOVS    R4, R5
MOVS    R5, #0
MOVS    R1, R5
MOVS    R5, #0
MOVS    R2, R5
```

```
loc_20002344
UXTH    R2, R2
CMP     R2, #0x10
BCS     loc_20002366
```

```
UXTH    R4, R4
ASRS.W  R5, R4, R2
ANDS.W  R5, R5, #1
MOVS    R3, R5
UXTH    R3, R3
RSBS.W  R5, R2, #0xF
LSLS.W  R5, R3, R5
ORRS    R1, R5
ADDS    R2, R2, #1
B       loc_20002344
```

```
loc_20002366
LSRS    R5, R1, #8
LDR     R6, =0x4005F3FC ; GPIO Port H Data
STR     R5, [R6]
LDR     R5, =0x4005E3FC ; GPIO Port G Data
STR     R1, [R5]
POP     {R4-R6}
BX      LR
; End of function UpdateGPIOOutputFromMemory
```

Function Entry

Section of code that stores value from app. layer in registers associated with LED Output

Address of mem. value

Address of LED Output

For our attack, we need to intercept the Control flow at the point where the value From memory is stored in the register whose value is manipulated in the loop, R5

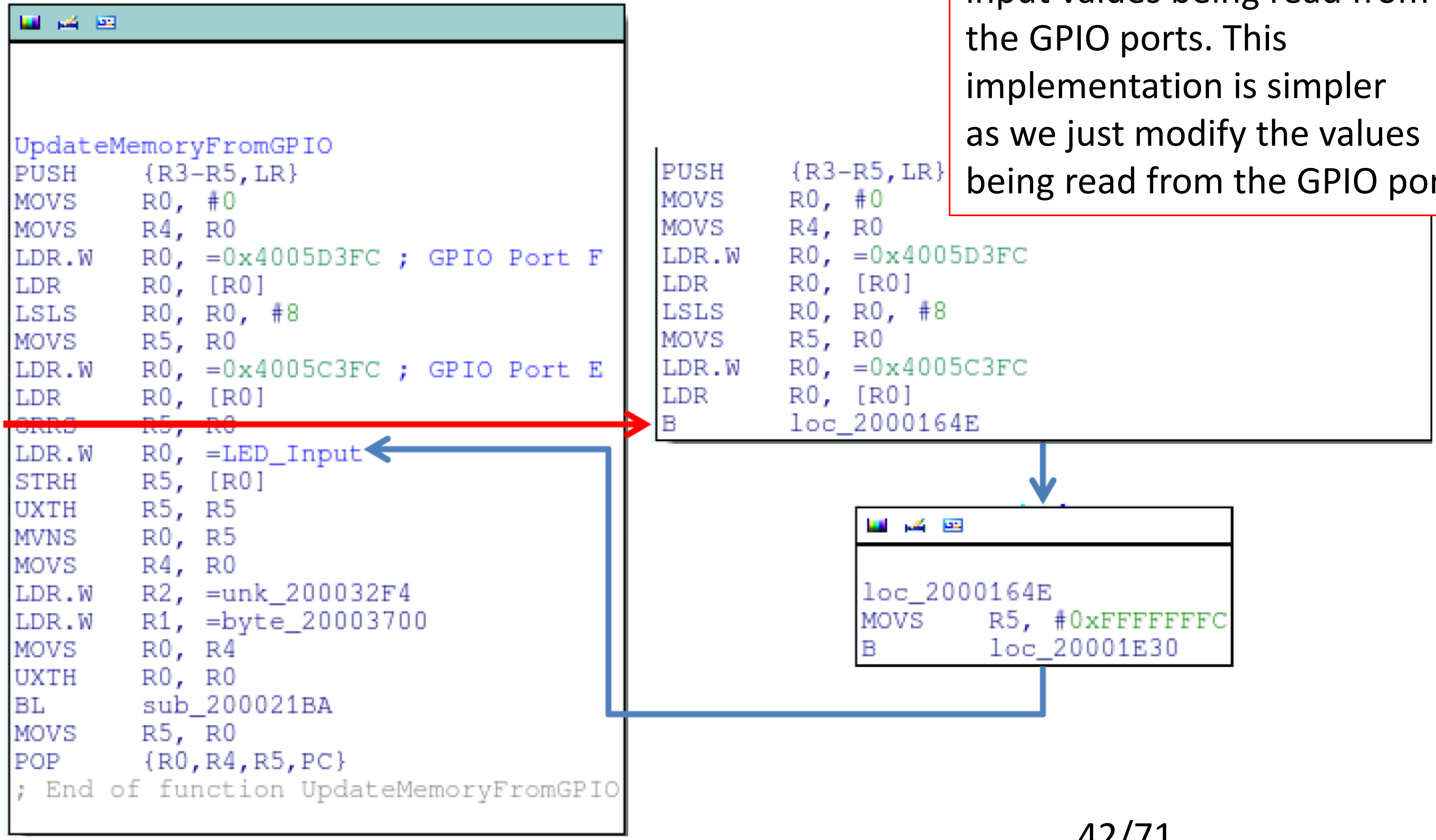
We branch to an arbitrary location of unused memory and run code that has has been injected. In this case, we store a a mask value to R5 to change the output values and branch back to the subsequent instructions.

Loop that changes value from memory to GPIO format

Once value is calculated, it is forwarded to GPIO output ports

Modified GPIO-Input Update ISR

We have a similar implementation for the input values being read from the GPIO ports. This implementation is simpler as we just modify the values being read from the GPIO ports



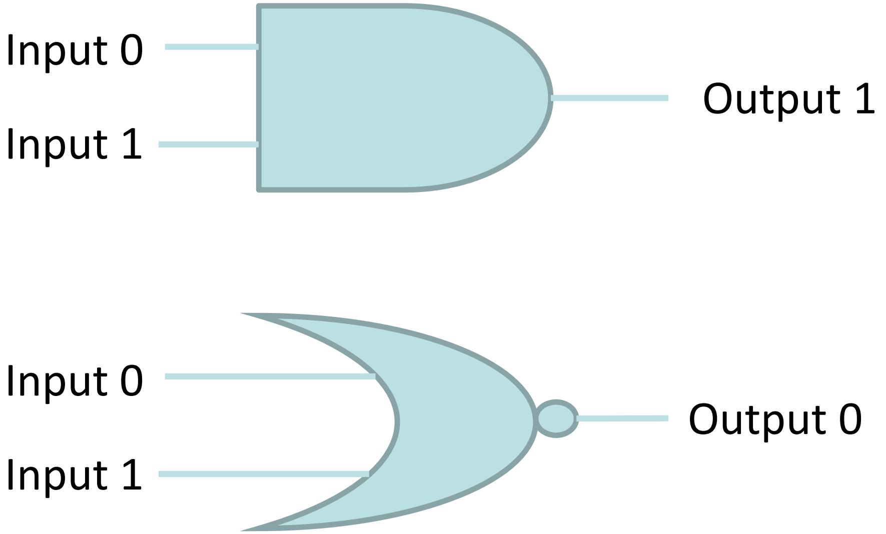
```
UpdateMemoryFromGPIO
PUSH    {R3-R5,LR}
MOVS    R0, #0
MOVS    R4, R0
LDR.W   R0, =0x4005D3FC ; GPIO Port F
LDR     R0, [R0]
LSLS    R0, R0, #8
MOVS    R5, R0
LDR.W   R0, =0x4005C3FC ; GPIO Port E
LDR     R0, [R0]
~~ORRS    R5, R0~~
LDR.W   R0, =LED_Input
STRH    R5, [R0]
UXTH    R5, R5
MVNS    R0, R5
MOVS    R4, R0
LDR.W   R2, =unk_200032F4
LDR.W   R1, =byte_20003700
MOVS    R0, R4
UXTH    R0, R0
BL      sub_200021BA
MOVS    R5, R0
POP     {R0,R4,R5,PC}
; End of function UpdateMemoryFromGPIO
```

```
PUSH    {R3-R5,LR}
MOVS    R0, #0
MOVS    R4, R0
LDR.W   R0, =0x4005D3FC
LDR     R0, [R0]
LSLS    R0, R0, #8
MOVS    R5, R0
LDR.W   R0, =0x4005C3FC
LDR     R0, [R0]
B       loc_2000164E
```

```
loc_2000164E
MOVS    R5, #0xFFFFFFFC
B       loc_20001E30
```

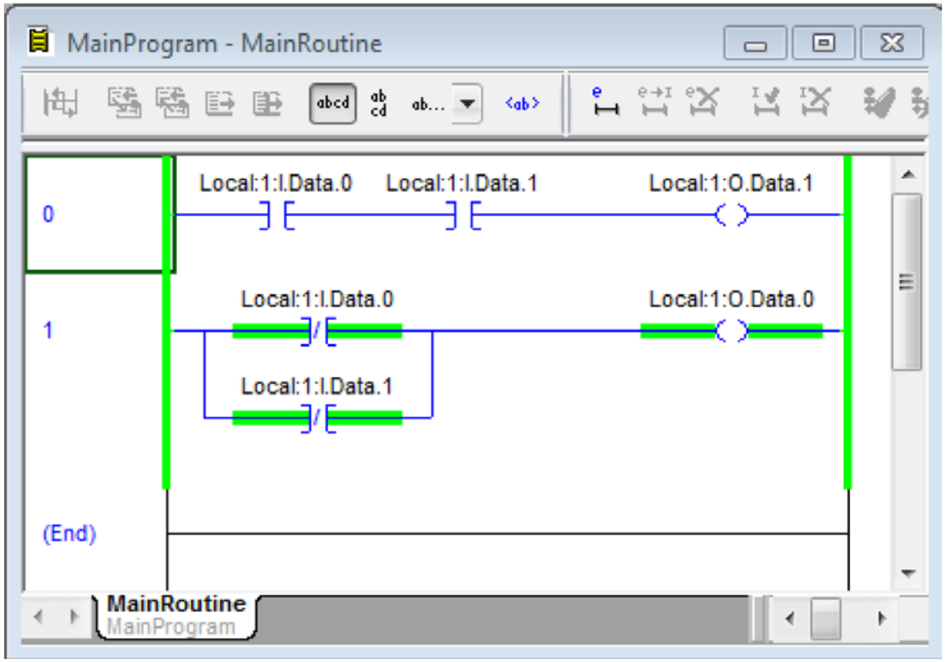
Example Attack Scenario

- Simple logic system:
  - If input ports 0 and 1 are high, then output port 1 is high (AND gate)
  - If input port 0 is low or input port 1 is low, then output port 0 is high (NOR gate)
- This system can represent a safety condition
  - We can only start a process (output port 1) if two safety conditions (input port 0 and input port 1) are met. Otherwise, we are in an idle position (output 0)

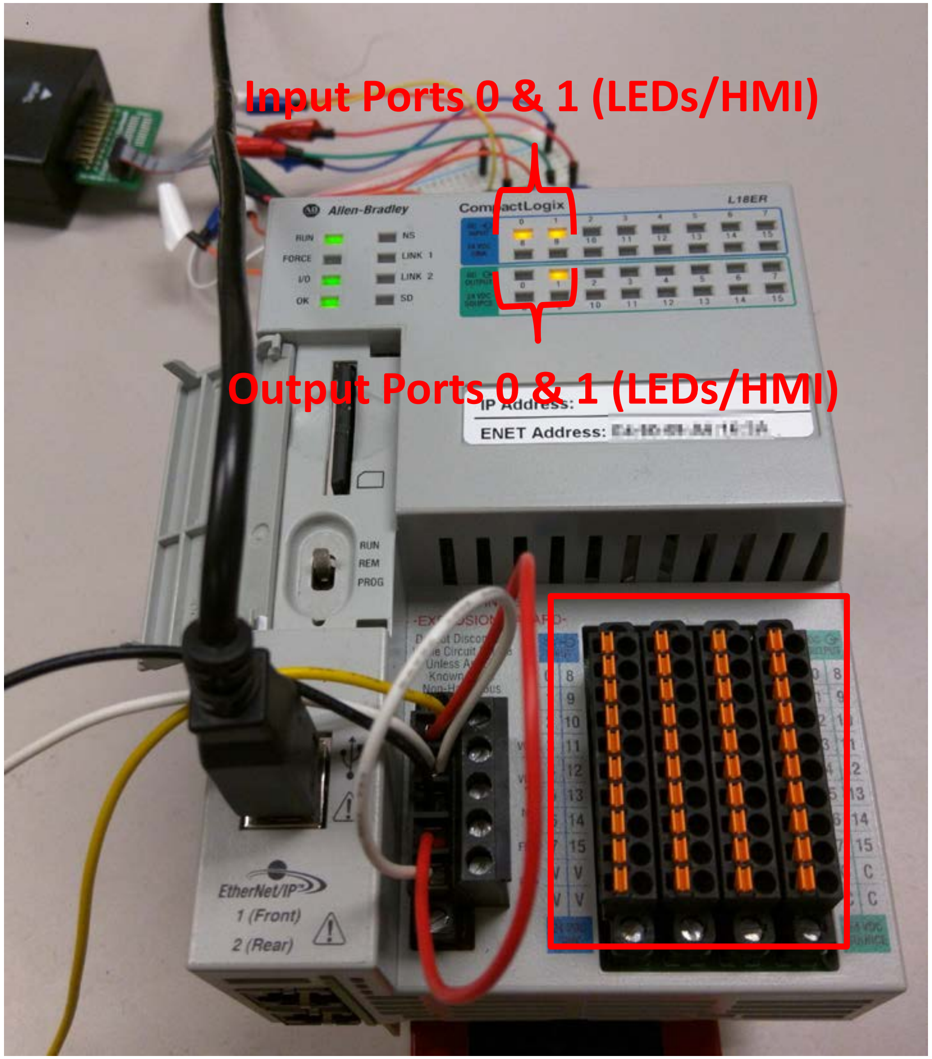


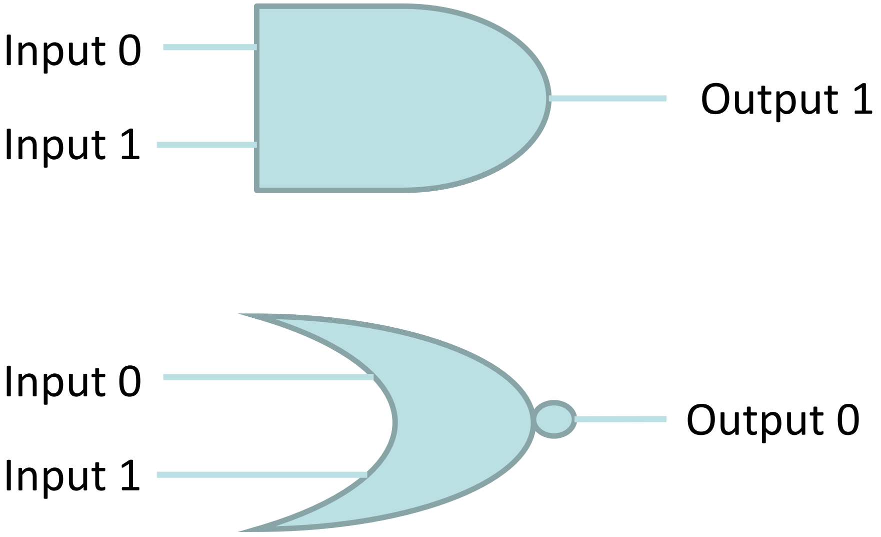
Simple Ladder Logic Program

- Ladder logic is a graphical programming language used to program simple circuit diagrams of relay logic hardware
- The system on the right represents the aforementioned AND and NOR gates
- The programming/ monitoring software, RSLogix 5000, is considered our HMI
  - LEDs and HMI read the updated values from the same addresses in memory



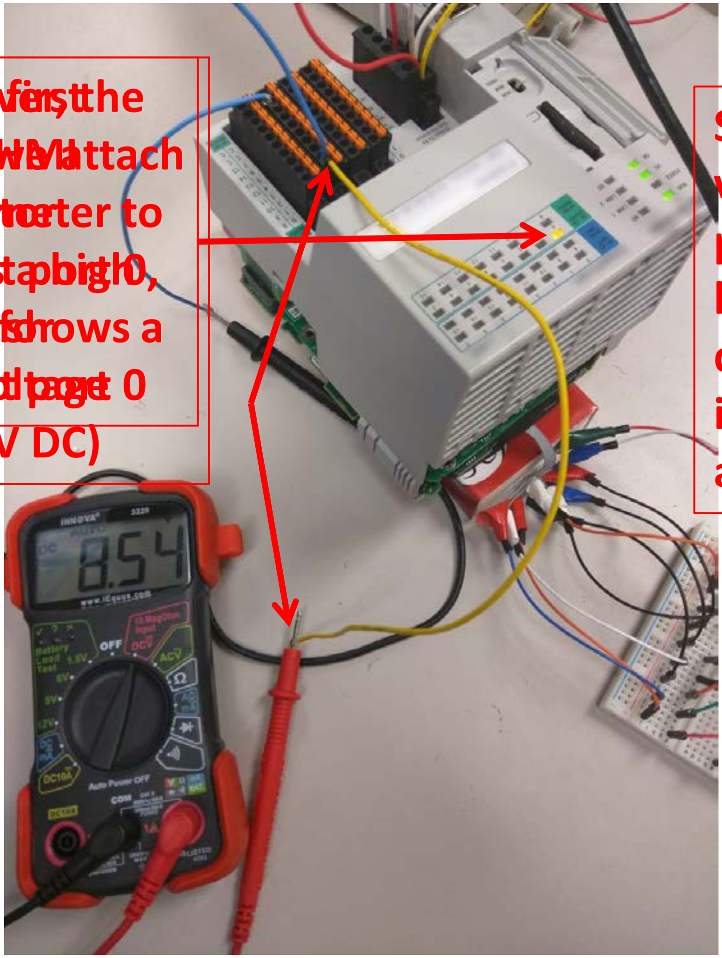
Spoofing Inputs





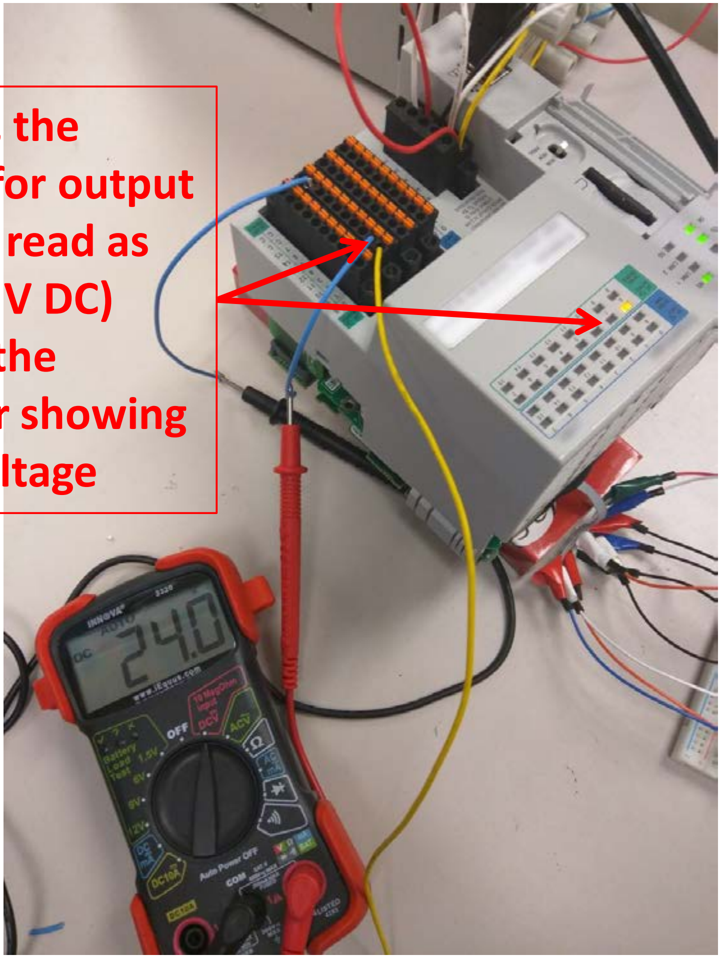
- The LEDs/HMI Indicators show that both input ports 0 and 1 are high, so output port 1 is high according to our ladder logic program
- There is no input connected! Output port 0 should be high and port 1 should be low!

Spoofing Outputs



In the first case, we attach a voltmeter to output port 0, and it shows a low voltage (8.54 V DC)

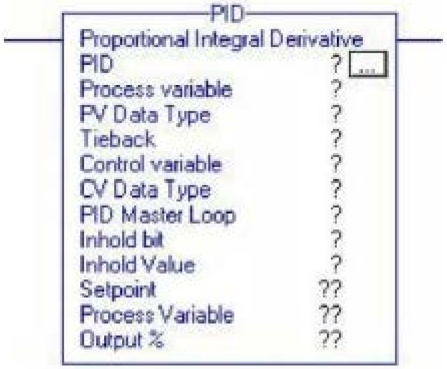
However, the LEDs/HMI indicator shows a high value for output port 0



Similary, the voltage for output port 1 is read as high (24 V DC) despite the indicator showing a low voltage

More Advanced Code Injection: PID Controller

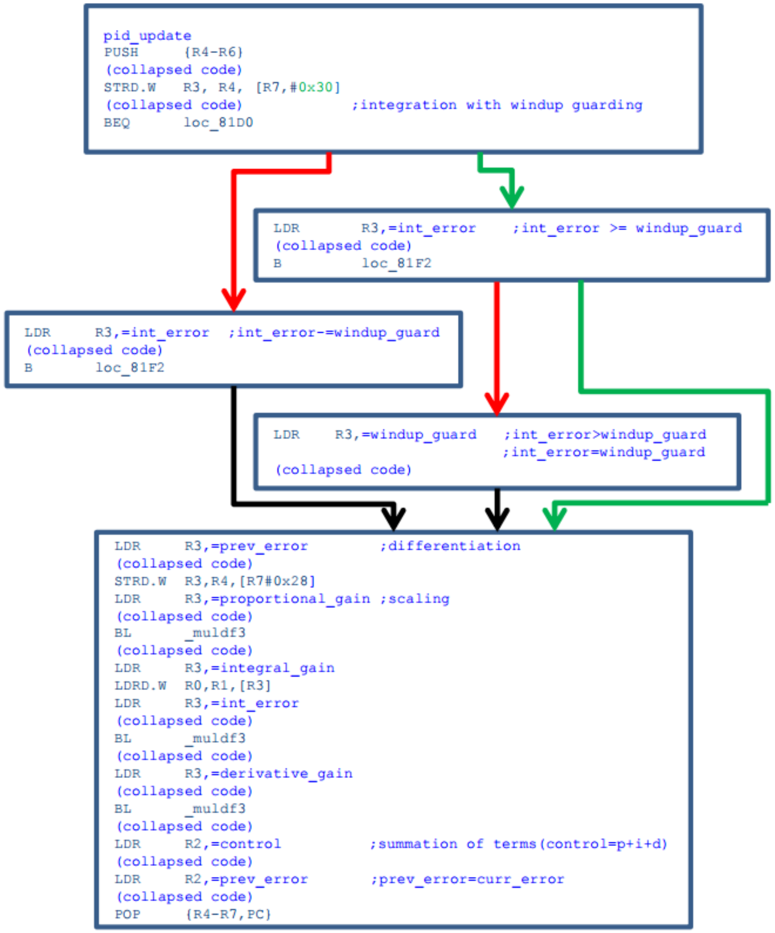
- Compiled an open-source PID controller code to determine space constraints
  - Did not have access to proprietary PID ladder logic instruction
  - Code was not optimized/stripped
  - PID implementation may only implement P or PI cases



```
PID
Proportional Integral Derivative
PID                  ?
Process variable     ?
PV Data Type         ?
Tieback              ?
Control variable     ?
CV Data Type         ?
PID Master Loop      ?
Inhold bit           ?
Inhold Value         ?
Setpoint            ??
Process Variable    ??
Output %            ??
```

Ladder Logic Instruction

Sample PID Code (collapsed)



```
pid_update
PUSH      {R4-R6}
(collapsed code)
STRD.W    R3, R4, [R7,#0x30]
(collapsed code)             ;integration with windup guarding
BEQ       loc_81D0
```

```
LDR       R3,=int_error    ;int_error >= windup_guard
(collapsed code)
B         loc_81F2
```

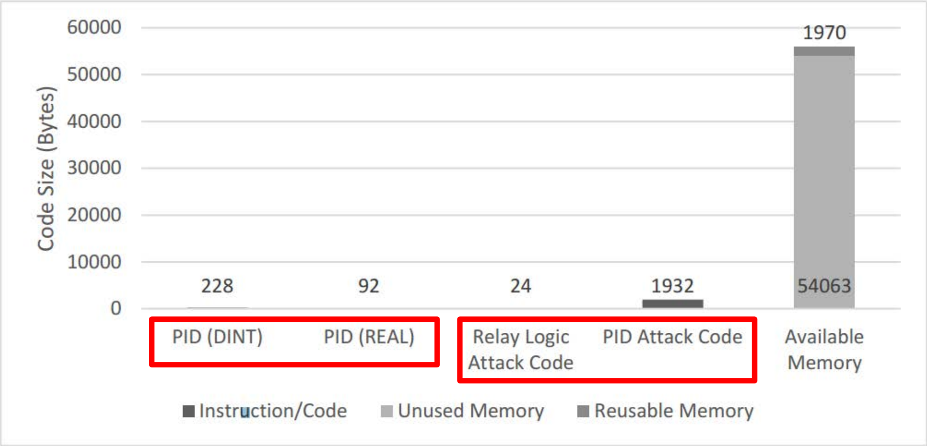
```
LDR     R3,=int_error  ;int_error-=windup_guard
(collapsed code)
B       loc_81F2
```

```
LDR     R3,=windup_guard    ;int_error>windup_guard
                            ;int_error=windup_guard
(collapsed code)
```

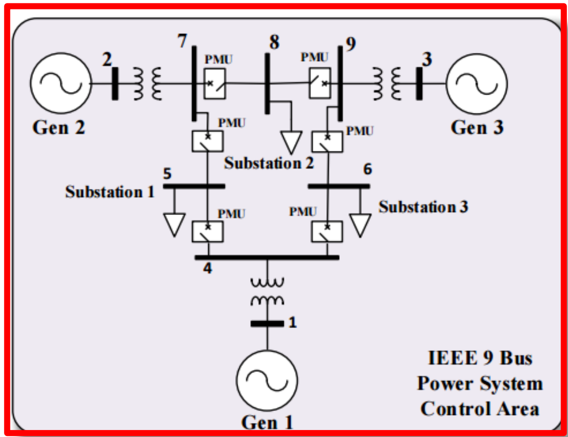
```
LDR     R3,=prev_error          ;differentiation
(collapsed code)
STRD.W  R3,R4,[R7#0x28]
LDR     R3,=proportional_gain ;scaling
(collapsed code)
BL      _muldf3
(collapsed code)
LDR     R3,=integral_gain
LDRD.W  R0,R1,[R3]
LDR     R3,=int_error
(collapsed code)
BL      _muldf3
(collapsed code)
LDR     R3,=derivative_gain
(collapsed code)
BL      _muldf3
(collapsed code)
LDR     R2,=control            ;summation of terms(control=p+i+d)
(collapsed code)
LDR     R2,=prev_error         ;prev_error=curr_error
(collapsed code)
POP     {R4-R7,PC}
```

Assessing Reusable Memory for Malware Injection

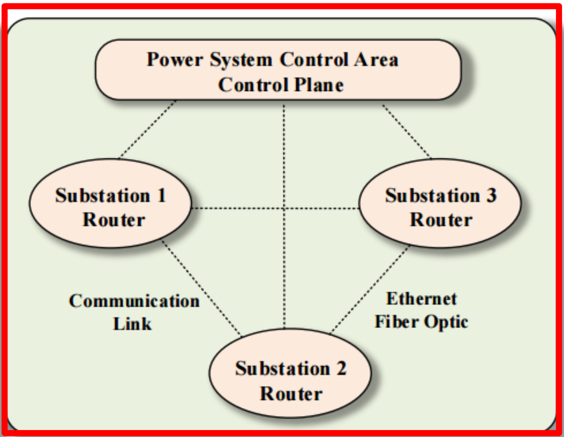
- Manually inspected code to determine "available" and "reusable" memory
  - "Reusable": code that is inaccessible due to the control flow of the code and can be overwritten
  - "Available": areas of memory that are not being used
- Available and reusable memory were sufficient enough to implement a PID attack code
  - PID attack code could be much leaner
  - Built-in PID instructions are significantly smaller than attack code



Evaluation on Smart Grid Test Bed



IEEE 9 Bus
Power System
Control Area



Power System Control Area
Control Plane

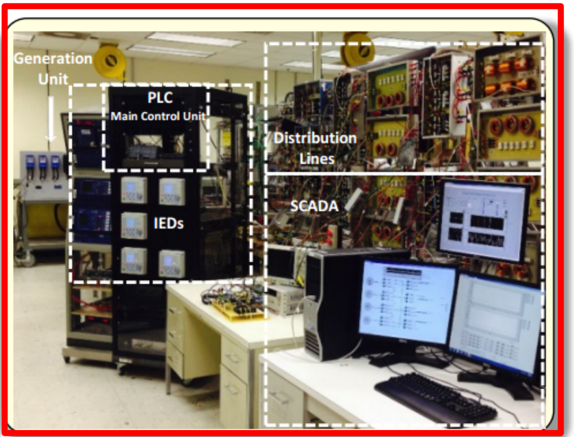
Substation 1
Router

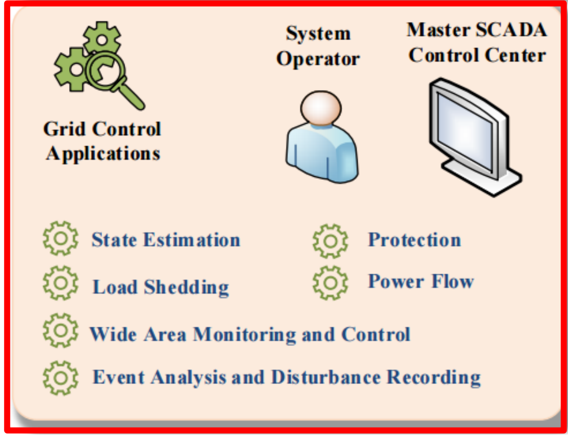
Substation 3
Router

Communication
Link

Ethernet
Fiber Optic

Substation 2
Router



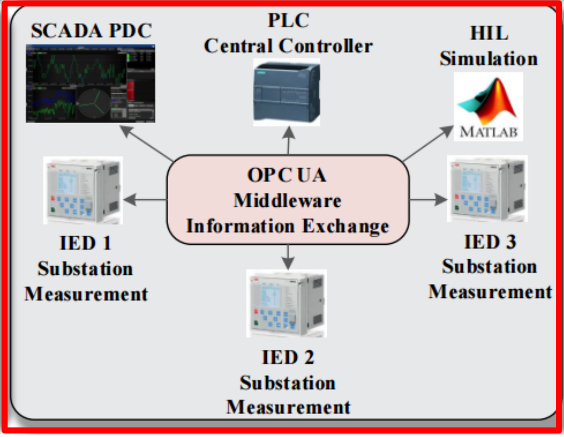


Grid Control
Applications

System
Operator

Master SCADA
Control Center

- State Estimation
- Protection
- Load Shedding
- Power Flow
- Wide Area Monitoring and Control
- Event Analysis and Disturbance Recording



SCADA PDC

PLC
Central Controller

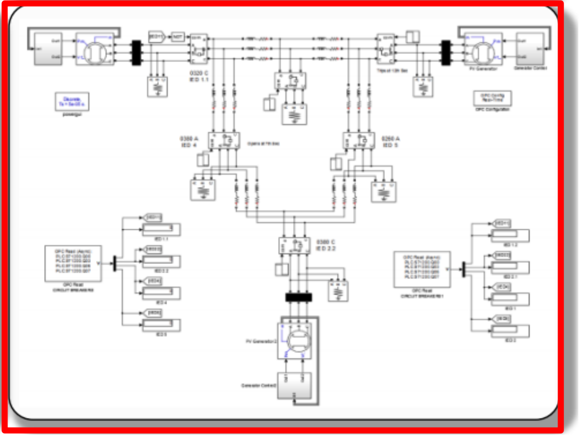
HIL
Simulation

OPC UA
Middleware
Information Exchange

IED 1
Substation
Measurement

IED 3
Substation
Measurement

IED 2
Substation
Measurement



Benign and Malicious Physical Models

Benign Optimal Power Flow (bOPF)

- Uses optimal power flow equations of power grid to minimize cost while ensuring safe operation, i.e.,

$$
\begin{aligned}
\min_{u} \quad & c(x,u) \\
\text{s.t.} \quad & P_i^g - P_i^l = \sum_k |V_i||V_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}) \\
& Q_i^g - Q_i^l = \sum_{k\in C} |V_i||V_k|(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}) \\
& P_l^g \leq P_l^{gmax} \\
& \forall i,j \in N,\ \forall l \in G,\ \forall k \in C
\end{aligned}
$$

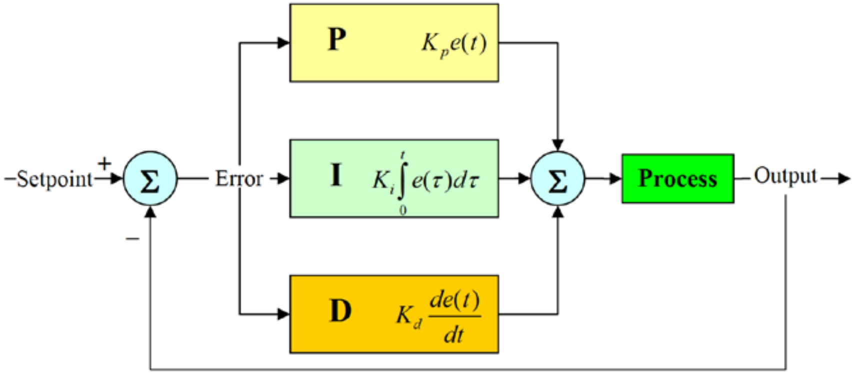
Malicious Optimal Power Flow (mOPF)

- Modified optimal power flow that maximizes cost while disregarding safety constraints, i.e.,

$$
\begin{aligned}
\max_{u} \quad & c(x,u) \\
\text{s.t.} \quad & P_i^g - P_i^l = \sum_k |V_i||V_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}) \\
& Q_i^g - Q_i^l = \sum_{k\in C} |V_i||V_k|(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}) \\
& \forall i,j \in N,\ \forall l \in G,\ \forall k \in C
\end{aligned}
$$

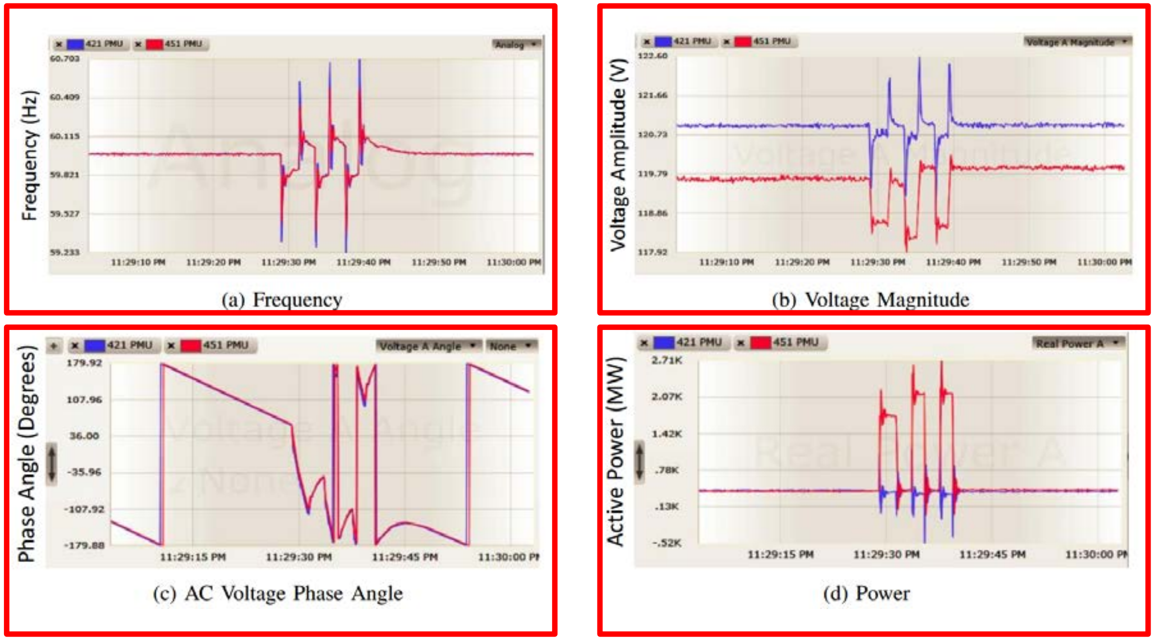
PID Controllers for Inner Loops of OPF Models

- Calculated commands of OPF models are used as set-points to be maintained by inner-loop proportional-integral-derivative (PID) controllers
- Harvey maintains an benign PID controller and associated set of variables along with a malicious PID controller



Steady-State System Malicious Attack: Actual Power System Measurements

- Repeated heavy load circuit breaker open/close triggering without loss of power system stability
  - Transmission line is opened/closed several times via a circuit breaker
- Although attack resulted in the system exceeding permissible limits, stability was maintained



(a) Frequency

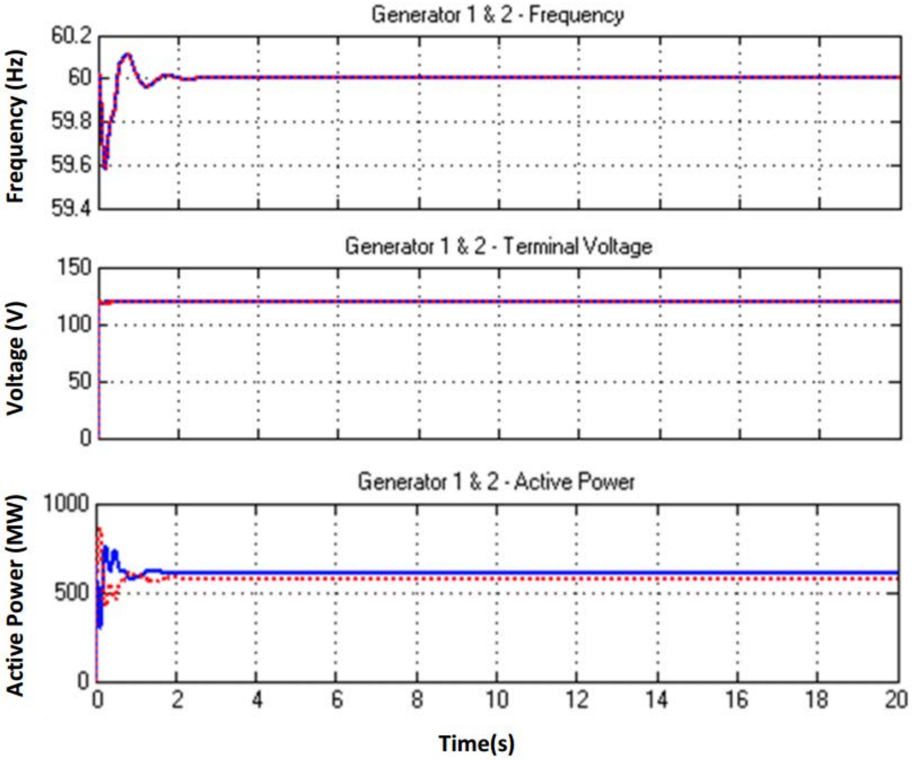
(b) Voltage Magnitude

(c) AC Voltage Phase Angle

(d) Power

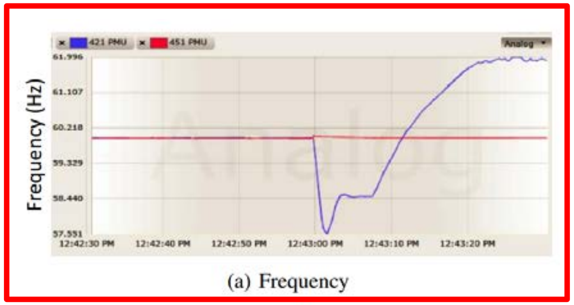
Steady-State System Malicious Attack: Faked Measurements

- Harvey ran parallel benign model to generate fake legitimate-looking sensor measurements to operators
- Such an attack caused minor perturbations due to equipment operational noise
  - They are shown as minor perturbations within safety limits
  - Such minor perturbations are normally observed

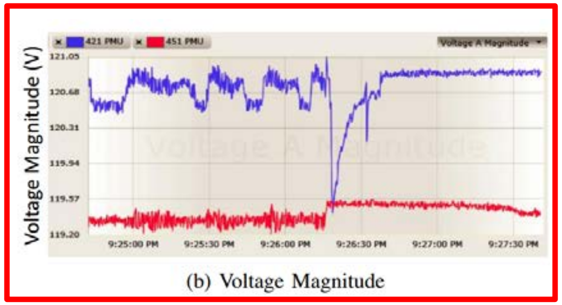


Adversary-Optimal Control Attack: Actual Power System Measurements

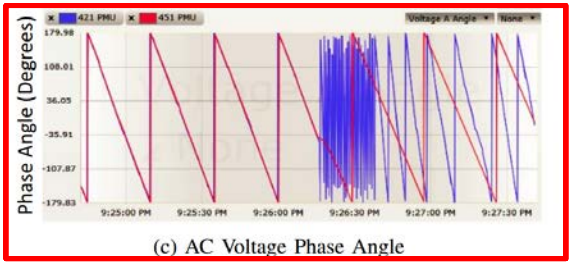
- Optimal malicious attack using real-world control algorithms, mOPF
  - Remove safety margin conditions
  - Replace cost minimization with maximization
  - Predefined stealthy conditions, e.g., “no power generator disconnect from the rest of the power grid”
  - Set nominal frequency reference to 62 Hz



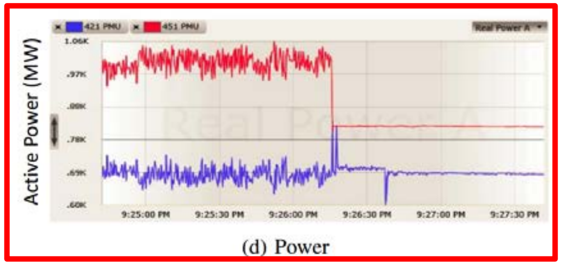
(a) Frequency



(b) Voltage Magnitude



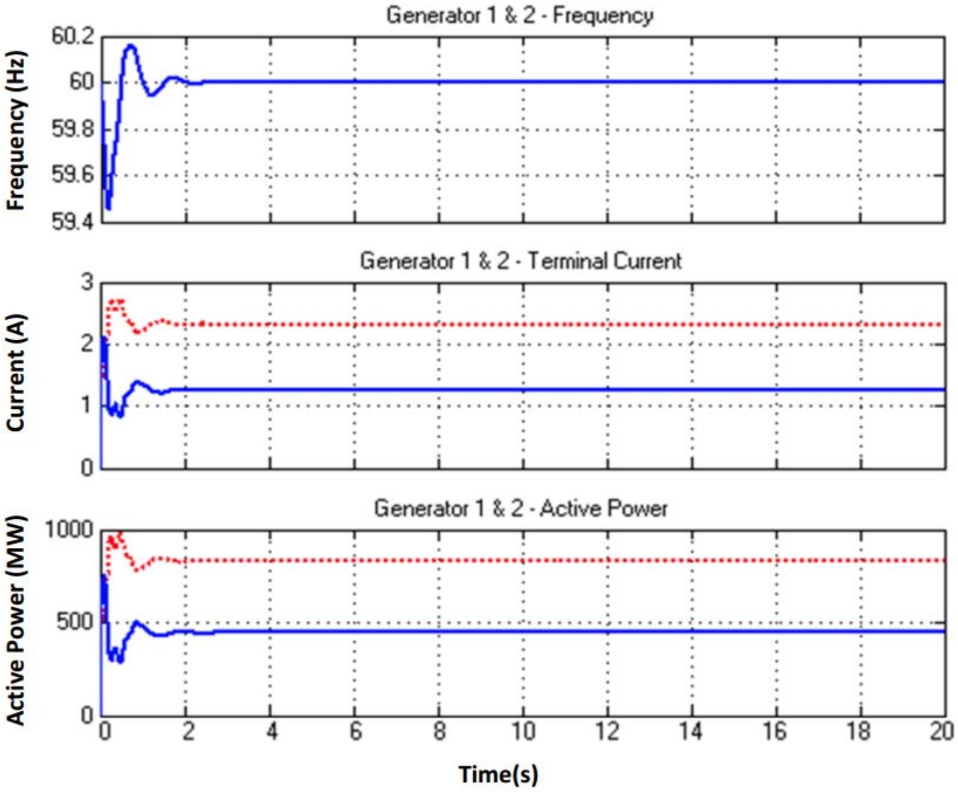
(c) AC Voltage Phase Angle



(d) Power

Adversary-Optimal Control Attack: Faked Measurements

- Harvey ran benign OPF in parallel and sent fabricated measurements back to HMI
- Similar perturbations were observed



Limitations

- Current implementation relies on JTAG implantation
- Accuracy of the physical models are limited to the amount of memory required by the implementations
- For a distributed attack, PLCs cannot rely on network communication
    - Communication relies on sensing and actuating, e.g., side-channel attack

Possible Mitigation Solutions for Harvey

- Remote-attestation
  - Verifier to check the software integrity of the system
- Secure boot
  - Trusted platform module to verify by the device itself
- External bump-in-the-wire between PLC and physical plant
  - Monitor sensor-to-PLC and PLC-to-actuator data streams

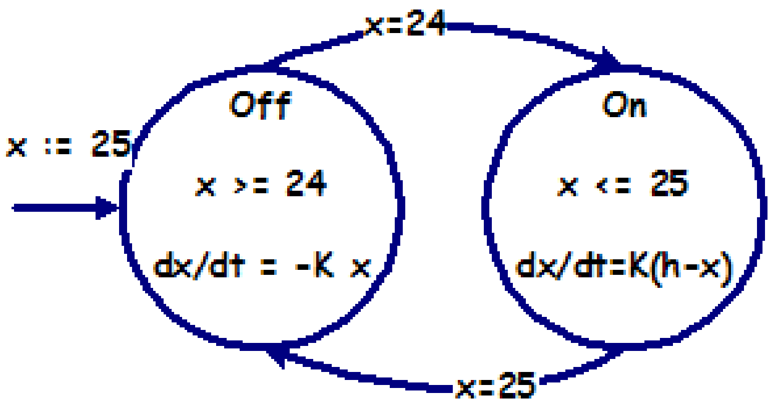
Responsible Disclosure

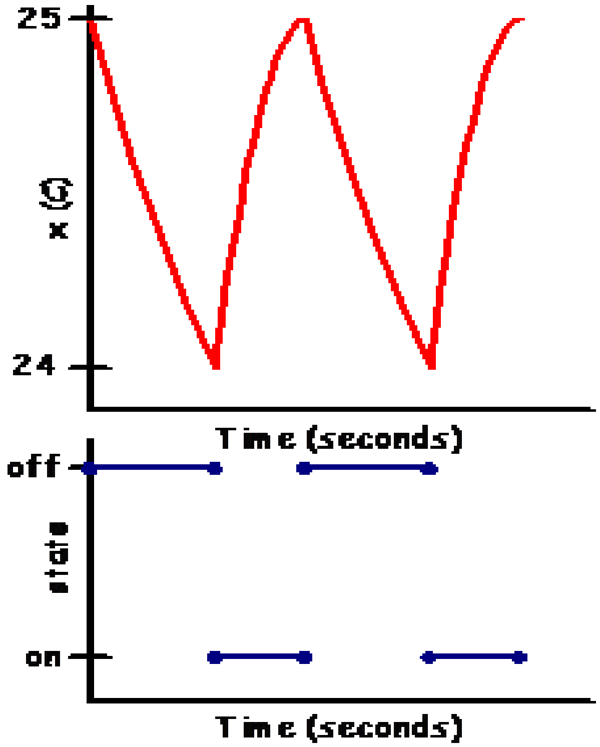
- We notified Allen Bradley of the possible repercussions of previously demonstrated firmware vulnerabilities
- The company allowed us to publish the details of our work in the Network and Distributed System Symposium (NDSS) 2017 conference

VERIFICATION OF CYBER-PHYSICAL MODELS

Hybrid Systems

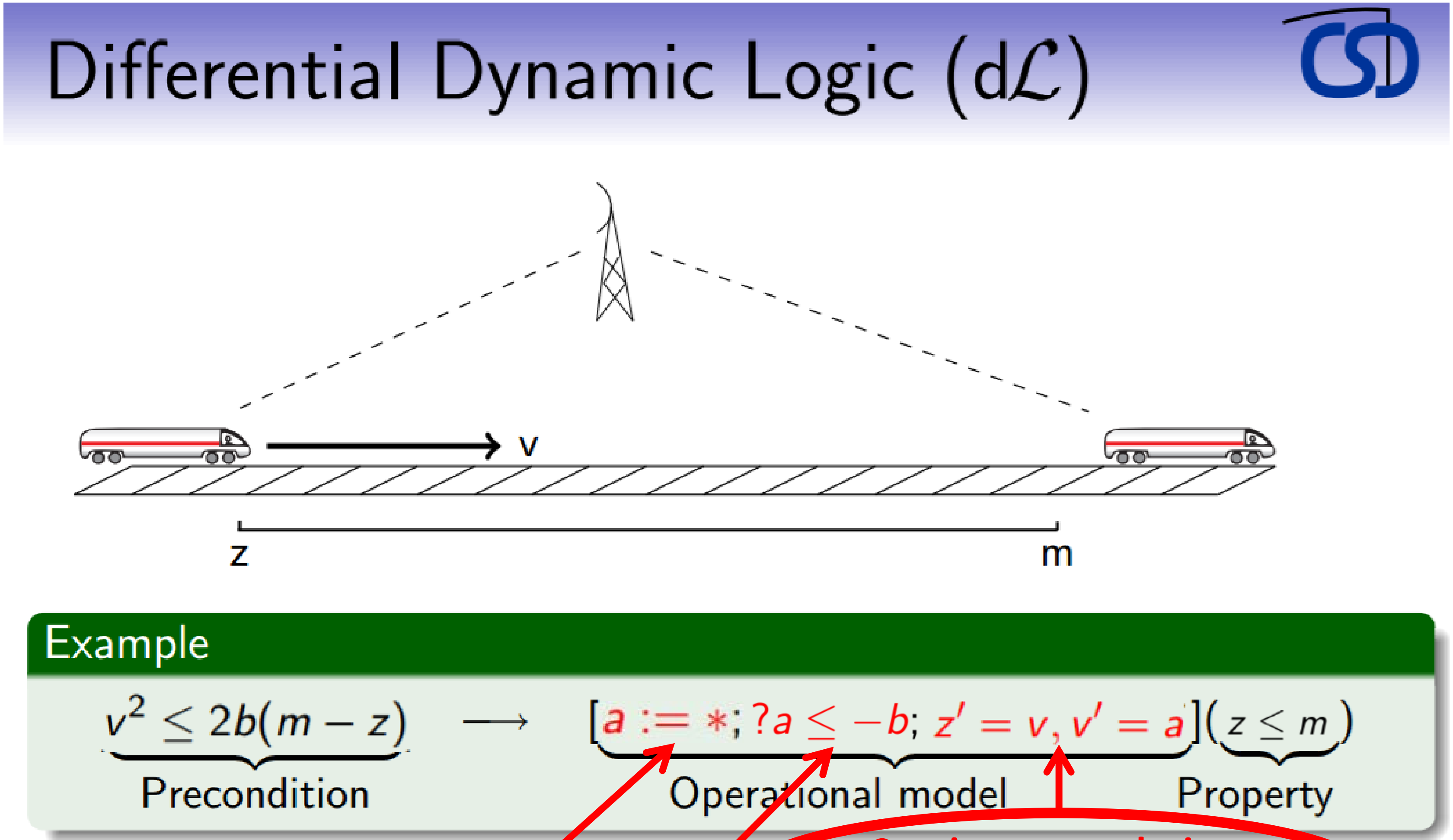
Hybrid automata: Thermostat example





- Control temperature by turning a heater *On* and *Off*
- Hybrid state: $(x,s) \in [24, 25] \times \{On, Off\}$

Hybrid Verification of Cyber-Physical Systems



Differential Dynamic Logic (d$\mathcal{L}$)

Example

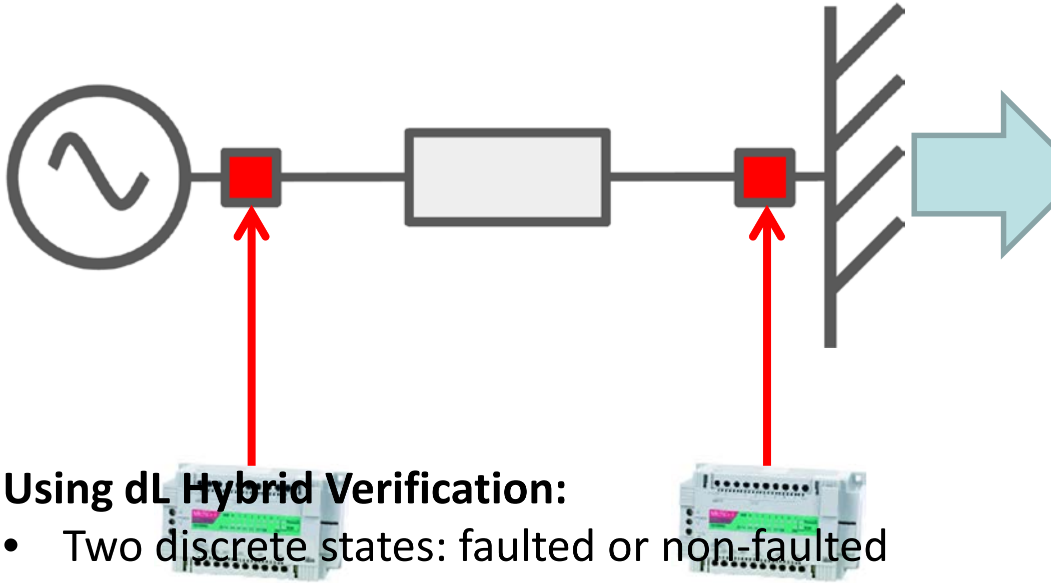
$$\underbrace{v^2 \le 2b(m-z)}_{\text{Precondition}} \longrightarrow \underbrace{[a := *; ?a \le -b; z' = v, v' = a]}_{\text{Operational model}} \underbrace{(z \le m)}_{\text{Property}}$$

Random assignment

Test

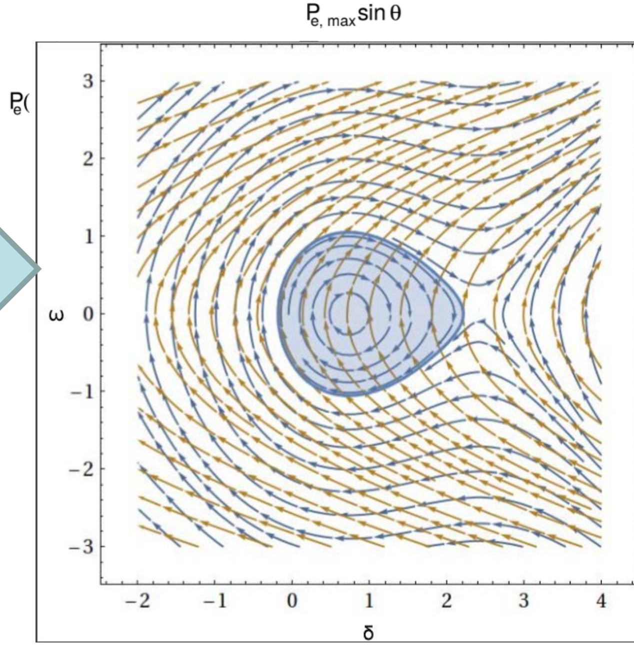
Continuous evolution: differential equation

Verifying the Transient Stability of Single-Machine Infinite-Bus (SMIB) System



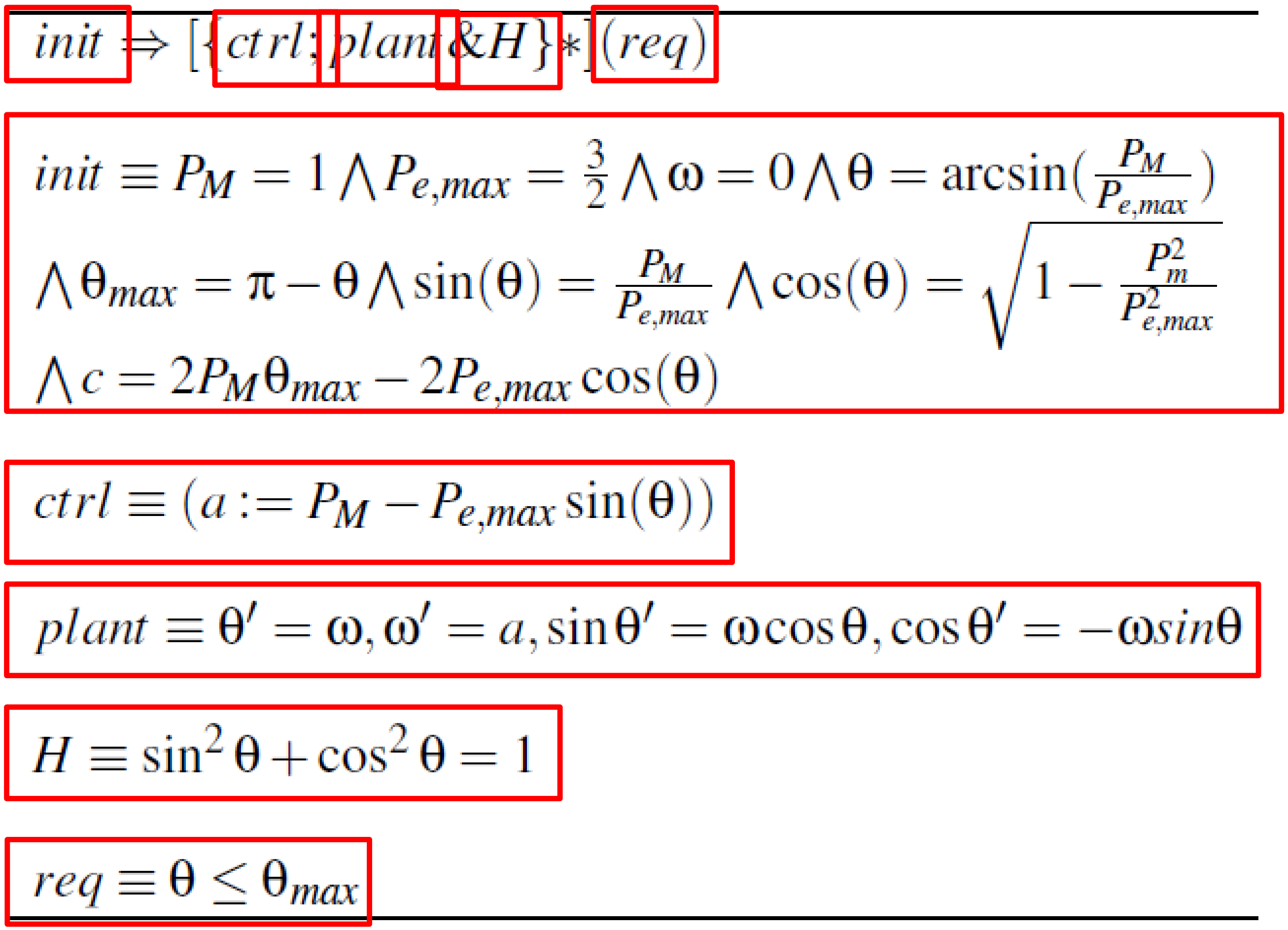
Using dL Hybrid Verification:

- Two discrete states: faulted or non-faulted
- Several simplifications made for verification



Hybrid Invariant Region

Final SMIB Hybrid Program



$$init \Rightarrow [\{ctrl; plant \& H\}*](req)$$

$$init \equiv P_M = 1 \wedge P_{e,max} = \frac{3}{2} \wedge \omega = 0 \wedge \theta = \arcsin(\frac{P_M}{P_{e,max}})$$
$$\wedge \theta_{max} = \pi - \theta \wedge \sin(\theta) = \frac{P_M}{P_{e,max}} \wedge \cos(\theta) = \sqrt{1 - \frac{P_m^2}{P_{e,max}^2}}$$
$$\wedge c = 2P_M\theta_{max} - 2P_{e,max}\cos(\theta)$$

$$ctrl \equiv (a := P_M - P_{e,max}\sin(\theta))$$

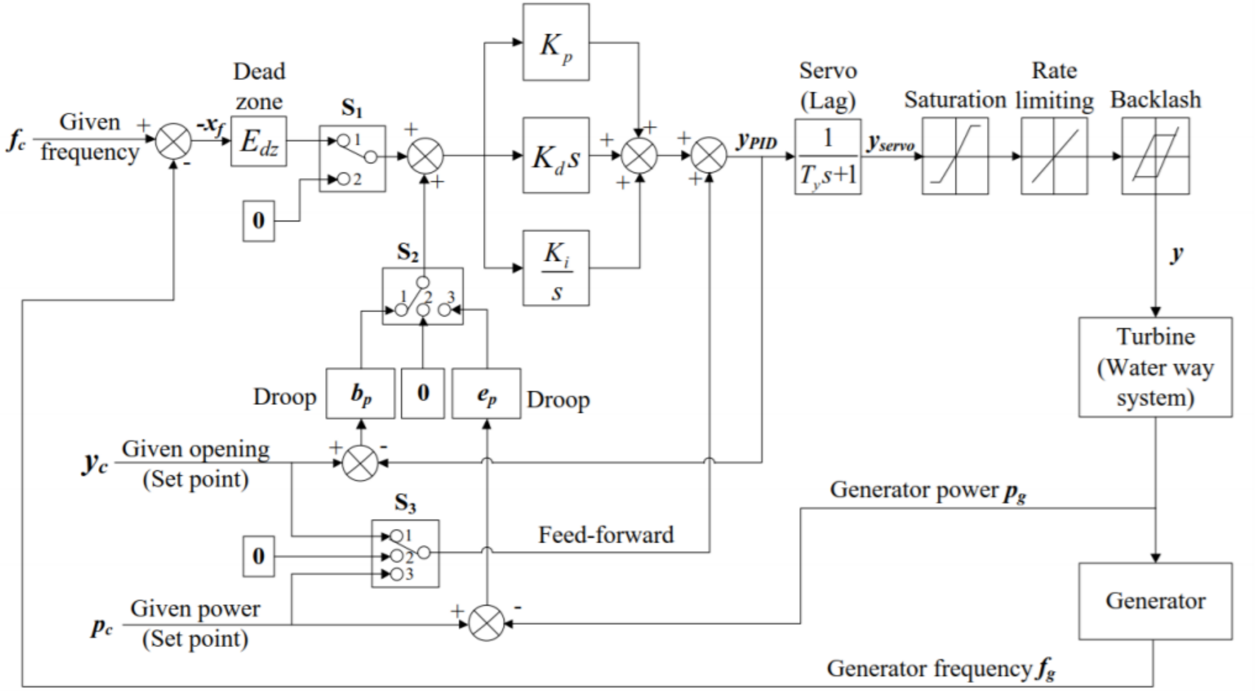
$$plant \equiv \theta' = \omega, \omega' = a, \sin\theta' = \omega\cos\theta, \cos\theta' = -\omega sin\theta$$

$$H \equiv \sin^2\theta + \cos^2\theta = 1$$

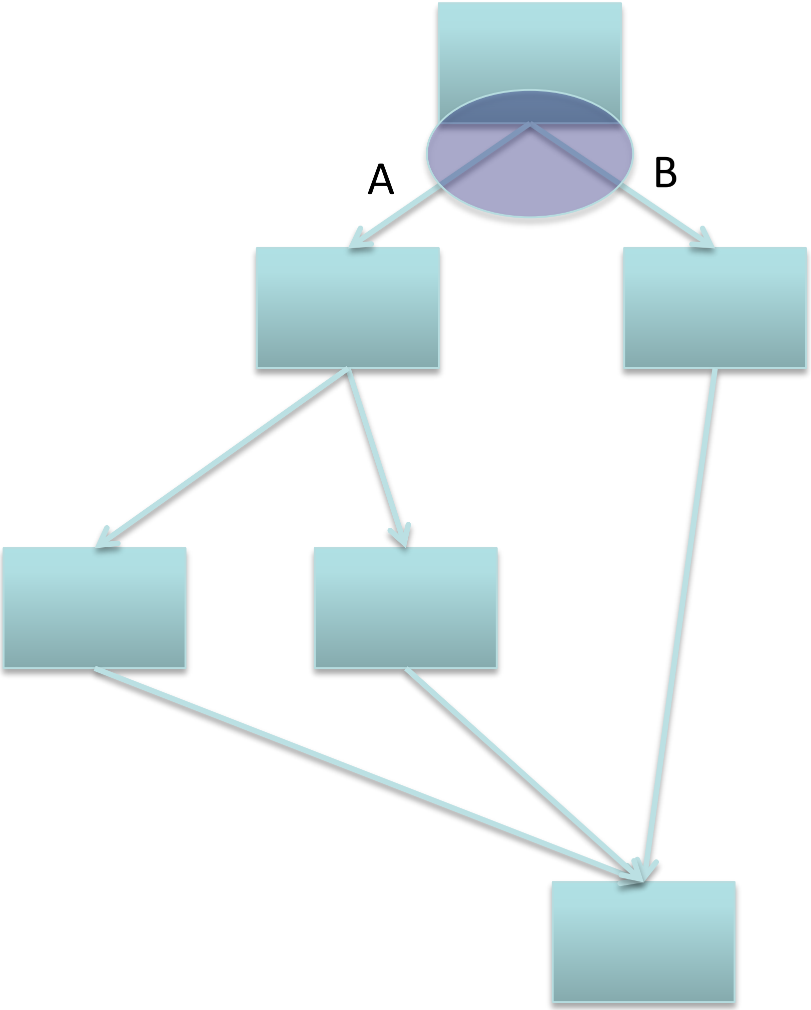
$$req \equiv \theta \leq \theta_{max}$$

Current and Future Work: Extending SMIB Model

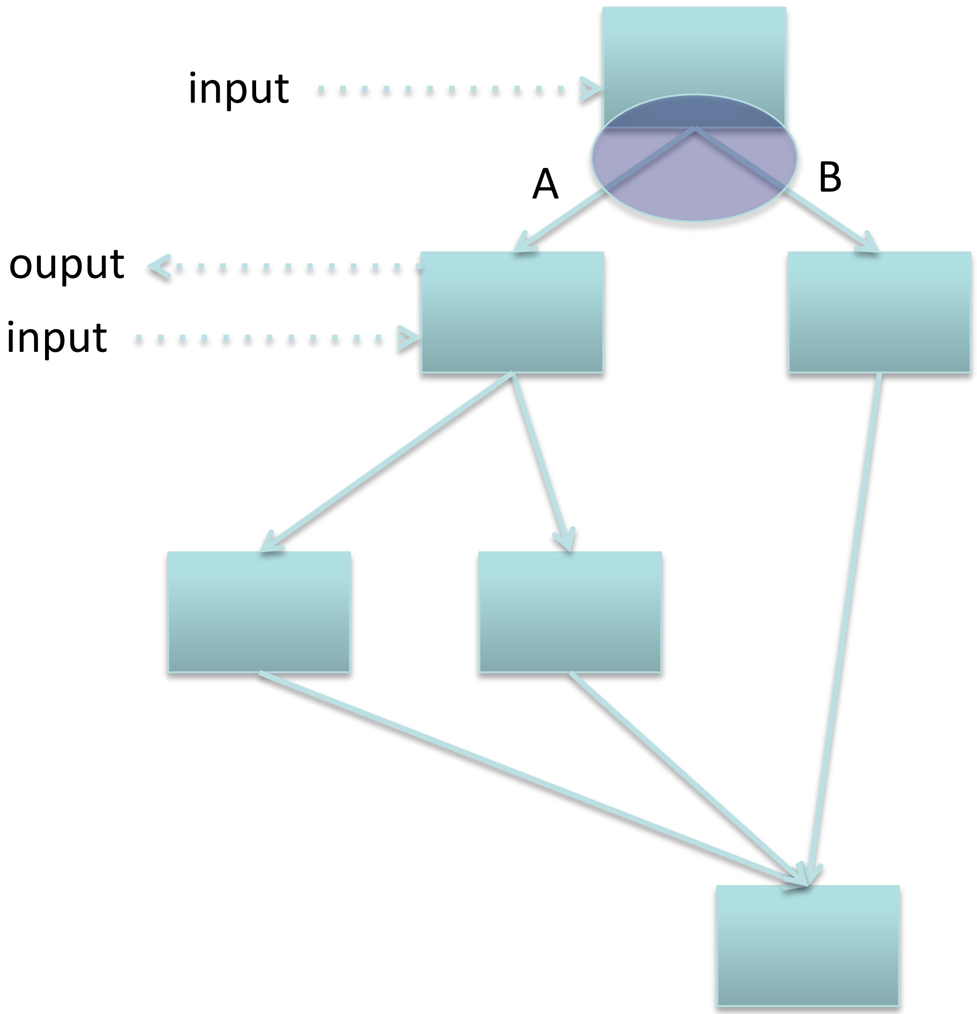
- Extending SMIB model to include model for governor of hydro power unit



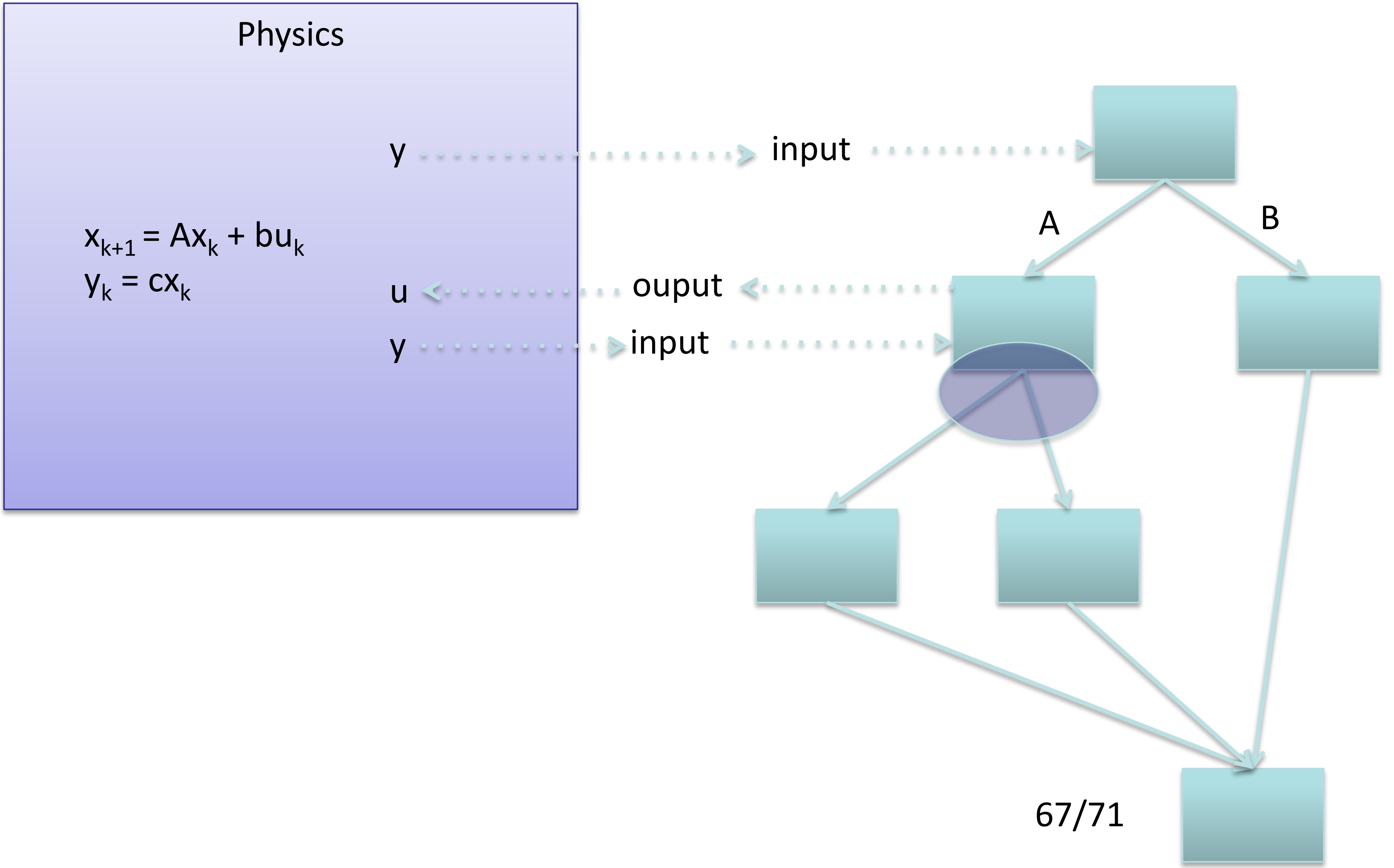
Current and Future Work: Cyber-Physical Control Flow Integrity



Current and Future Work:
Cyber-Physical Control Flow Integrity



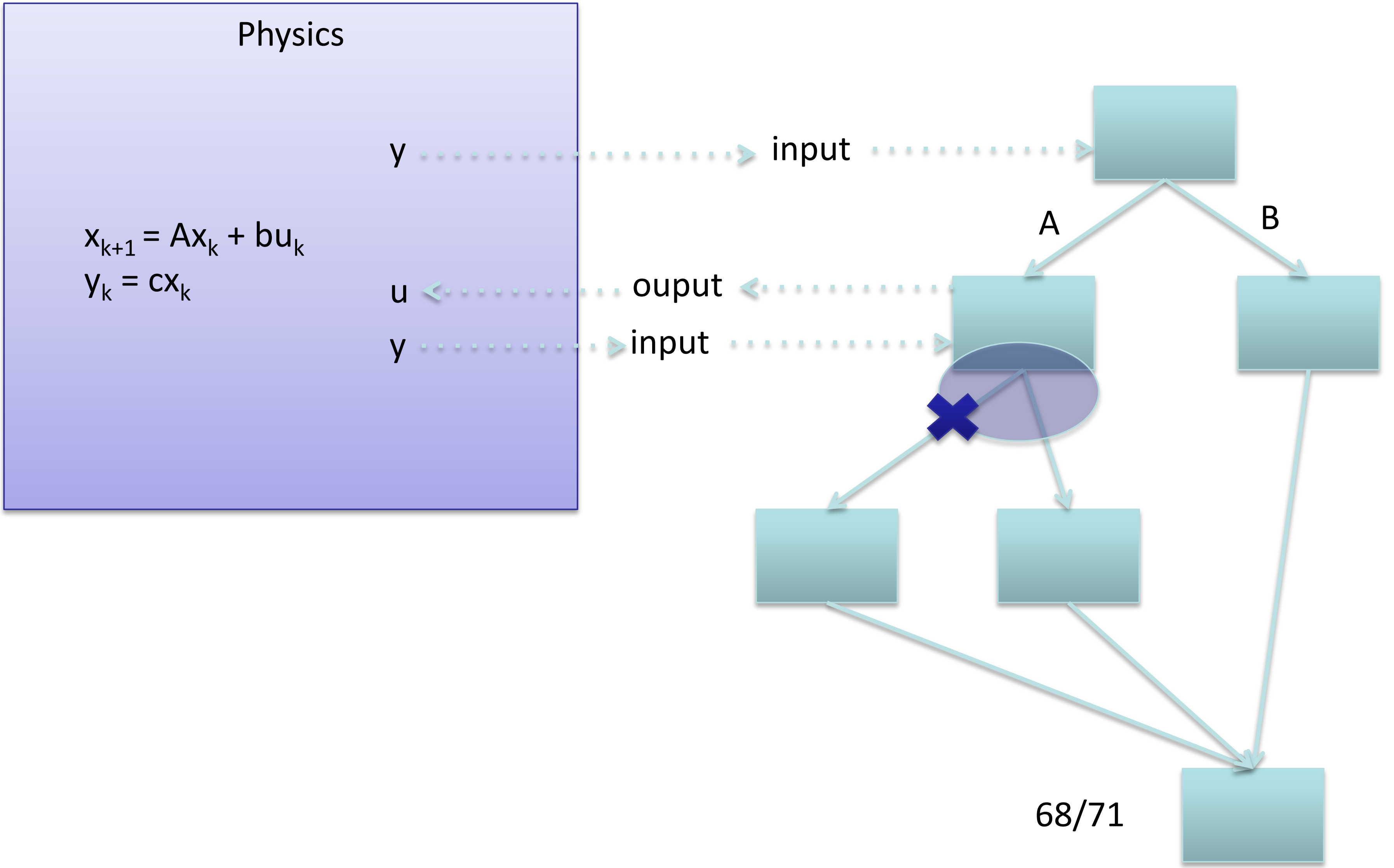
Current and Future Work:
Cyber-Physical Control Flow Integrity



Physics

$$x_{k+1} = Ax_k + bu_k$$
$$y_k = cx_k$$

Current and Future Work: Cyber-Physical Control Flow Integrity



Physics

$x_{k+1} = Ax_k + bu_k$

$y_k = cx_k$

Conclusion

- We presented Harvey, a PLC rootkit that implements a physics-aware man-in-the-middle attack against cyber-physical control systems
- Harvey damages the underlying physical system while providing the operators with the exact view of the system that they would expect to see following their commands
- We presented device-oriented verification of cyber-physical systems with a focus on the electric power grid using differential dynamic logic

Thank You!

Luis Garcia
E-mail: l.garcia2@rutgers.edu

List of Publications

- Journal Articles:
  - Katherine R. Davis, Charles M. Davis, Saman A. Zonouz, Rakesh B. Bobba, Robin Berthier, Luis Garcia, Peter W. Sauer**, A Cyber-Physical Modeling and Assessment Framework for Power Grid Infrastructures**, IEEE Transactions on Smart Grid, 2015
- Conference/Workshop Articles:
  - Luis Garcia, Henry Senyondo, Stephen McLaughlin, Saman Zonouz, **Covert Channel Communication Through Physical Interdependencies in Cyber-Physical Infrastructures**, IEEE SmartGridComm, 2014
  - Saman Zonouz, Luis Garcia, **TMQ: Threat Model Quantification in Smart Grid Critical Infrastructures**, IEEE SmartGridComm, 2014
  - Gabriel Salles-Loustau, Luis Garcia, Kaustubh Joshi, Saman Zonouz, **Swirls: Context-Aware Information-Flow-Based Micro-Security Perimeters for Mobile Devices**, IEEE/FIP International Conference on Dependable Systems and Networks (DSN), 2016
  - Luis Garcia, Dong Wei, Leandro Pfleger de Aguiar, Saman Zonouz, **Detecting PLC Control Corruption via On-Device Runtime Verification**, IEEE Resilience Week (RWS), 2016
  - Luis Garcia, Ferdinand Brasser, Mehmet Hazar, Osama Mohammed, Ahmad-Reza Sadeghi, Saman Zonouz, **Hey, My Malware Knows Physics!**
    Attacking PLCs with Physical Model Aware Rootkit, Network and Distributed System Security Symposium (NDSS), 2017
  - Luis Garcia, Khalil Ghorbal, Saman Zonouz, **Transient Stability of Power Systems: A Case Study in Formal Verification**, ACM International Conference on Hybrid Systems: Computation and Control (HSCC), 2017