Cyber Physical Systems Security:
Protecting Programmable Logic Controllers

Matthew Chan, Prof. Saman Zonouz

Work supported by the NSF under grant CNS-1703782

Introduction and Background

Programmable Logic Controllers (PLCs):

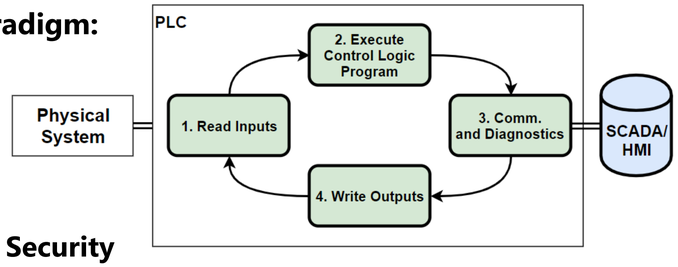
- Part of industrial control systems that manage critical infrastructure such as:
  - Power Systems
  - Water treatment
  - Automated Manufacturing/Refining
  - Elevators and Traffic Lights

Background:

- PLCs are flexible tools for industrial automation
- Required to have high availability and reliability
- Security policies and enforcement are lacking

Operation & Motivation

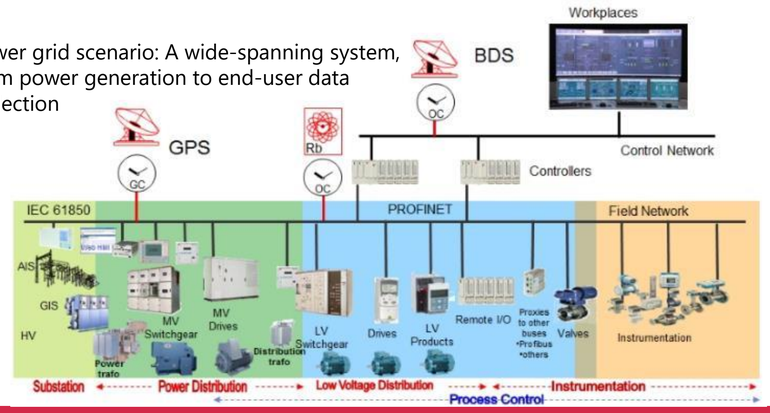
Scan Cycle Paradigm:



Motivation for Security

- PLCs are widely deployed in complex systems with large attack surfaces
- Legacy infrastructure and poor security practices leave systems vulnerable
- Growing trend of smart connected devices to enable greater functionality

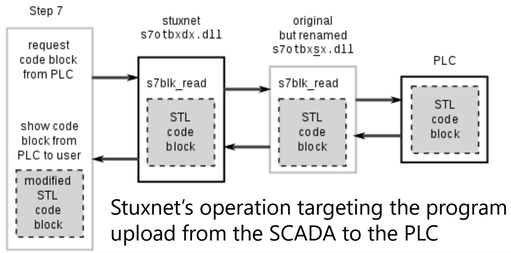
Power grid scenario: A wide-spanning system, from power generation to end-user data collection



Previous Work

Existing Exploits

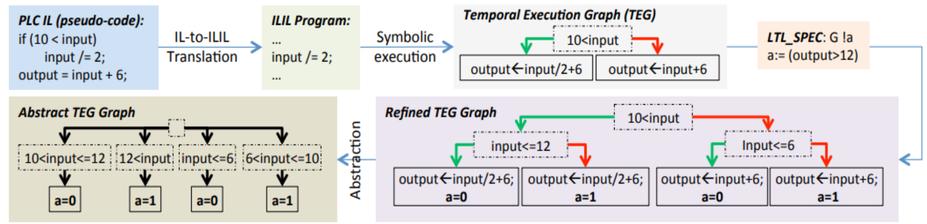
- The most (in)famous: **Stuxnet worm**
  - Targeted uranium refinery centrifuges
  - Infiltrated PLCs through SCADA control software
- **CrashOverride**
  - A sophisticated malware discovered in 2016 targeting a Ukrainian power substation
- **Harvey** [1]
  - A proof of concept PLC rootkit with physics awareness
- **Triton**
  - a malware targeting Middle East power stations



Stuxnet's operation targeting the program upload from the SCADA to the PLC

Existing External Defenses

- **WeaselBoard** [3]
  - External board for process monitoring
  - Uses traffic analysis to detect malicious activity
- **Trusted Safety Verifier (TSV)** [2]
  - Bump-in-the-wire solution for checking PLC programs as they are uploaded
  - Transforms assembly level PLC code (IL) into an Intermediate Language (ILIL)
  - Performs a symbolic analysis to check that safety conditions are maintained

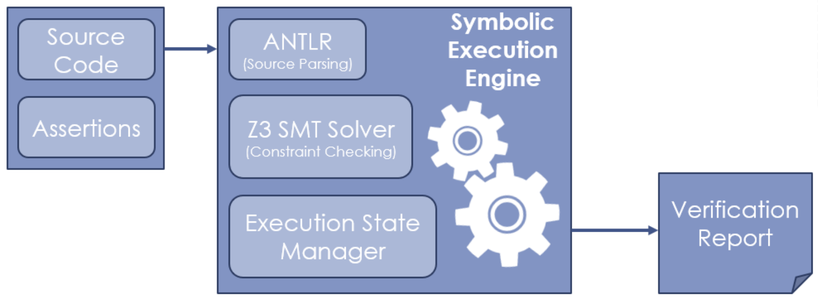


An example of TSV's program checking control flow

Current Work

TSV Extension: Symbolic Execution on Source

- Newer PLC security measures encrypt communication between the SCADA and PLC
- Modify symbolic execution to be applied on source code
- Developing language-specific grammars
- Can assist with debugging during program development



Testing

- OSCAT, a base library of PLC functions
- More complex controller code to analyze scalability
  - PLC controlled robotic arm example

An example of edge case behavior where symbolic execution excels

```
function min(a, b) { . . . }

// formal specification
ASSERT( min ≤ a && min ≤ b
&& (min = a || min = b) )
```

```
function min(a, b) {
    if (a < b) { return a; }
    else if (a = 100) { return -1; }
    else { return b; }
}
```

References:
[1] Garcia, Luis, et al. "Hey, my malware knows physics! Attacking PLCs with physical model aware rootkit." Proceedings of the Network & Distributed System Security Symposium, San Diego, CA, USA. 2017.
[2] McLaughlin, Stephen E., et al. "A Trusted Safety Verifier for Process Controller Code." NDSS. Vol. 14. 2014.
[3] https://www.sandia.gov/news/publications/labnews/articles/2016/04-03/weaselboard.html
[3] https://commons.wikimedia.org/wiki/File:IEC_62439-3_Clocks_in_Industrial_Automation_170208_HK.jpg
[4] https://commons.wikimedia.org/wiki/File:Siemens_Logo.jpg
[5] https://commons.wikimedia.org/wiki/File:Stuxnet_modifying_plc.svg
[6] https://dragos.com/blog/crashoverride/CrashOverride-01.pdf
[7] https://www.wired.com/story/triton-malware-targets-industrial-safety-systems-in-the-middle-east/