A Trusted Safety Verifier for Processor Controller Code

Stephen McLaughlin, Saman Zonouz, Devin Pohly, Patrick McDaniel

Penn State University, University of Miami

NDSS 2014

Industrial Control Systems 101

- Control systems are ***everywhere***
  - Power grid, automation, manufacturing, transportation, etc.
  - Big implications if hacked
- Control systems are attractive for attackers
  - Stuxnet
  - Port scanning PLCs
  - Remote network attacks on PLCs

Trusted Computing Base

- “Set of all subsystems that are required for security of a system”
  - Hardware
  - Firmware
  - Network
  - Software
  - You name it!
- **Problem: ICS TCB is TOO BIG!**
- **Solution: Build a system to reduce the size**

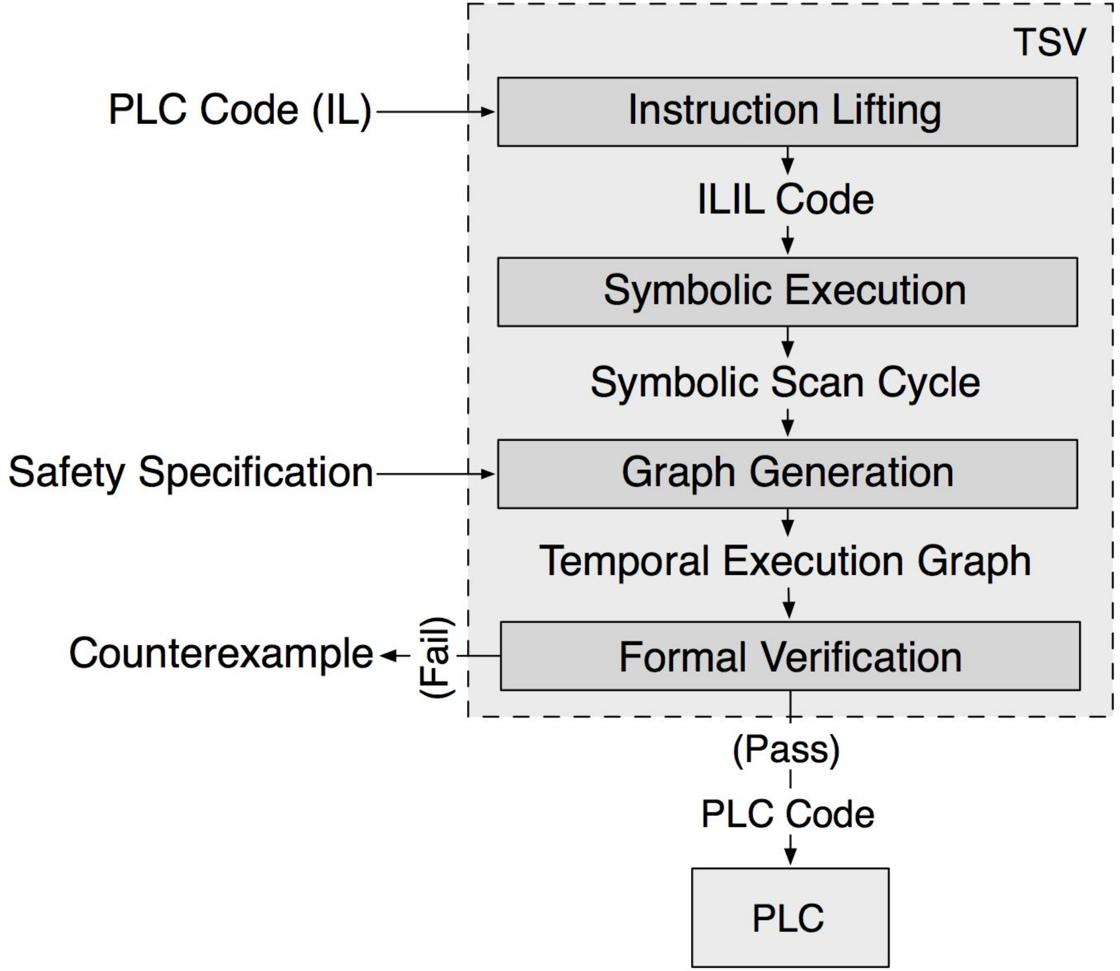
Programmable Logic Controller (PLC)

- “Digital, multi-input multi-output computer used for automation of physical machinery”
- Reads sensor measurements, takes action on those measurements
- Has a fixed program that runs continuously as long as the PLC is on
  - Each execution called a *scan cycle*
  - Input memory, program execution, output memory
- **TSV Goal:** Efficiently check code coming over the interface for safety properties

Threat Model

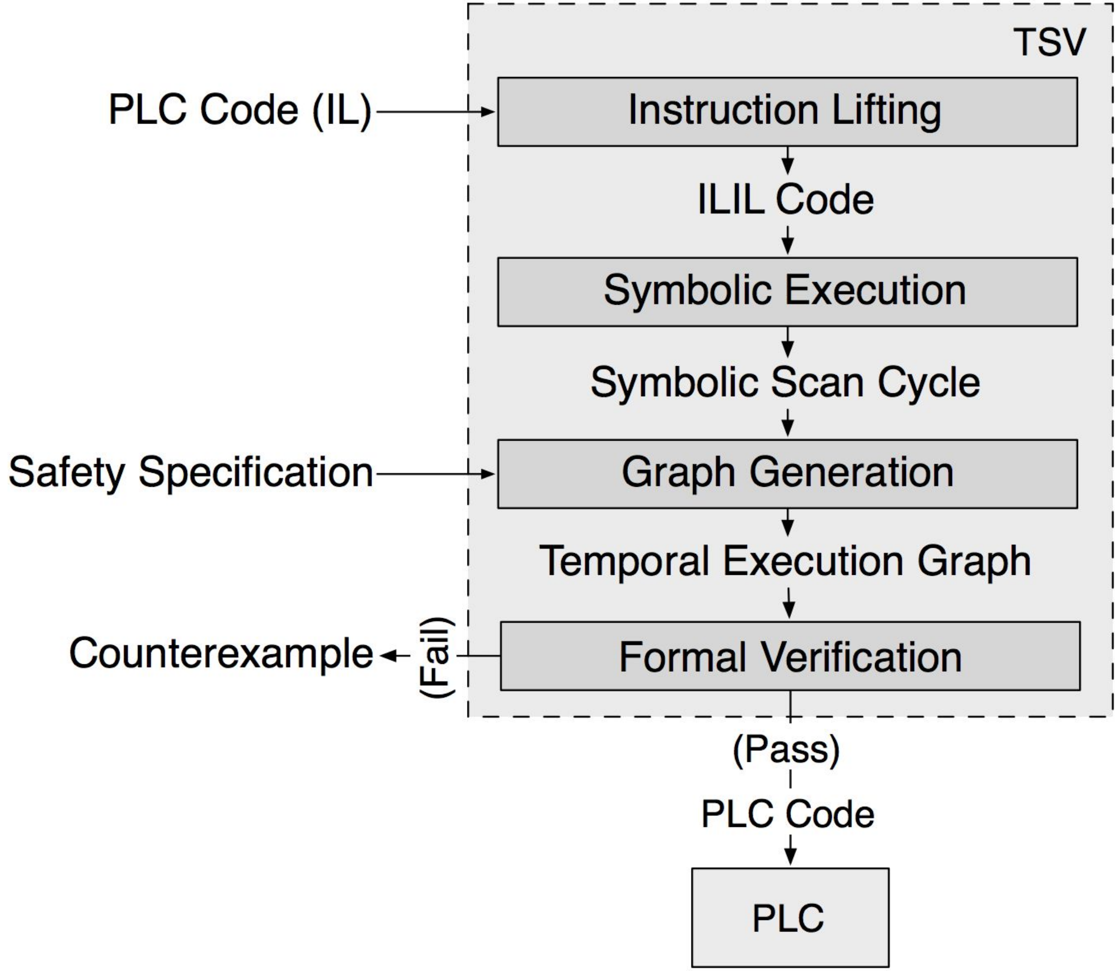
Limitations

- Underlying assumption is that the *firmware on PLC is trusted*
    - Firmware attacks can bypass PLC
- Bad sensor data is out of scope
- Inside attacker is out of scope


TSV
PLC Code (IL)
Instruction Lifting
ILIL Code
Symbolic Execution
Symbolic Scan Cycle
Safety Specification
Graph Generation
Temporal Execution Graph
Counterexample
(Fail)
Formal Verification
(Pass)
PLC Code
PLC

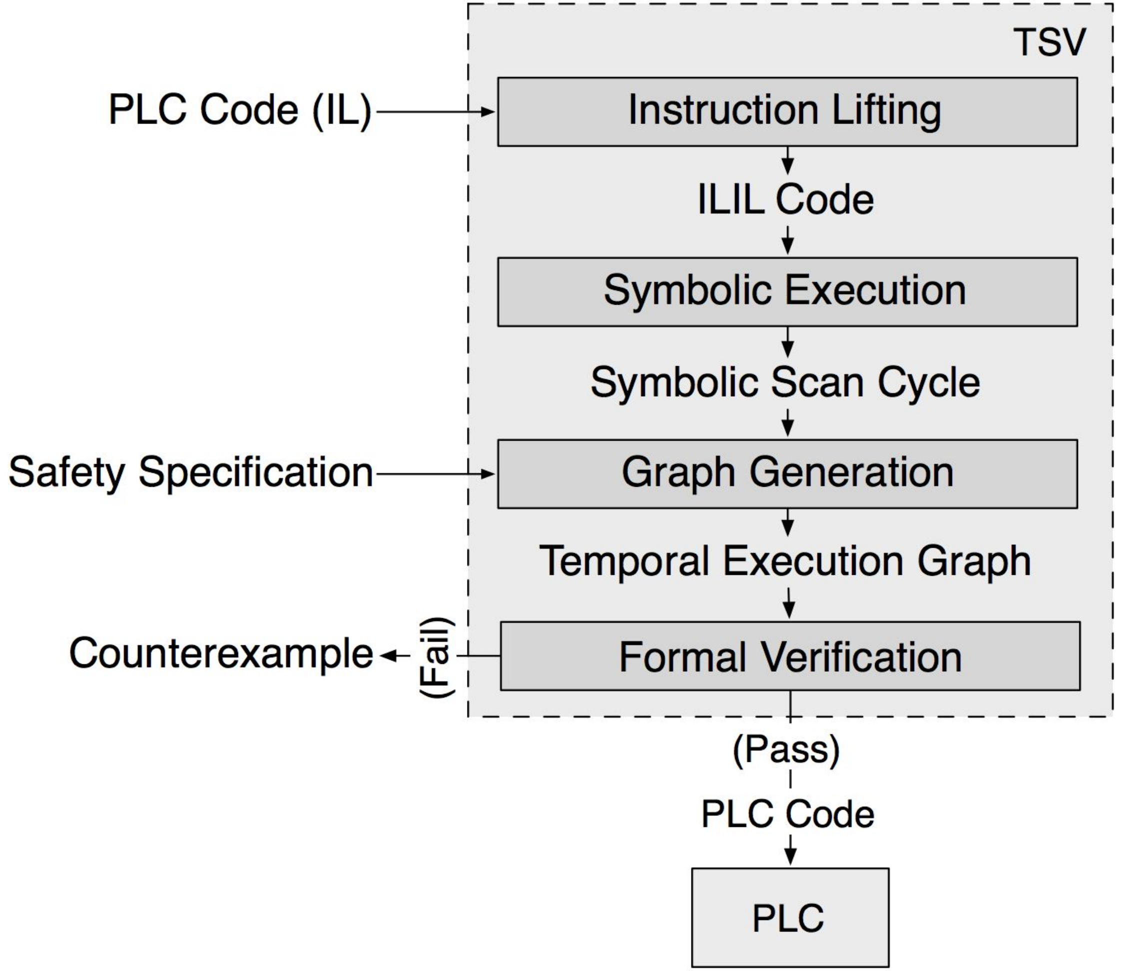
IL(IL)

- PLC programs are written in “Instruction List”, which is a form of ladder logic
  - Looks like a graphical diagram based on logic hardware
  - Set of instructions to operate physical machinery
- IL programs are *vendor specific*, so they are very hard to analyze directly
- Authors built ILIL, which is a set of “top-level instructions followed by function definitions”
  - Elevated IL to a program with *no side effects*, so safety properties can be measured


TSV
PLC Code (IL)
Instruction Lifting
ILIL Code
Symbolic Execution
Symbolic Scan Cycle
Safety Specification
Graph Generation
Temporal Execution Graph
Counterexample
(Fail)
Formal Verification
(Pass)
PLC Code
PLC

Symbolic Execution

- Analyze a program to see what inputs produce what outputs
  - Naively think of trying all possible inputs and outputs to every branch in a program
- TSV creates *symbolic scan cycle*
  - Set of logic formulas that should map all inputs and outputs
  - SMT solver used to prune infeasible branches
  - Hard deadline termination reduces the symbol equation space
  - Lump all inputs that produce similar outputs to reduce solving space
- Why is this useful?


TSV
PLC Code (IL)
Instruction Lifting
ILIL Code
Symbolic Execution
Symbolic Scan Cycle
Safety Specification
Graph Generation
Temporal Execution Graph
Counterexample
(Fail)
Formal Verification
(Pass)
PLC Code
PLC

Checking Temporal Properties

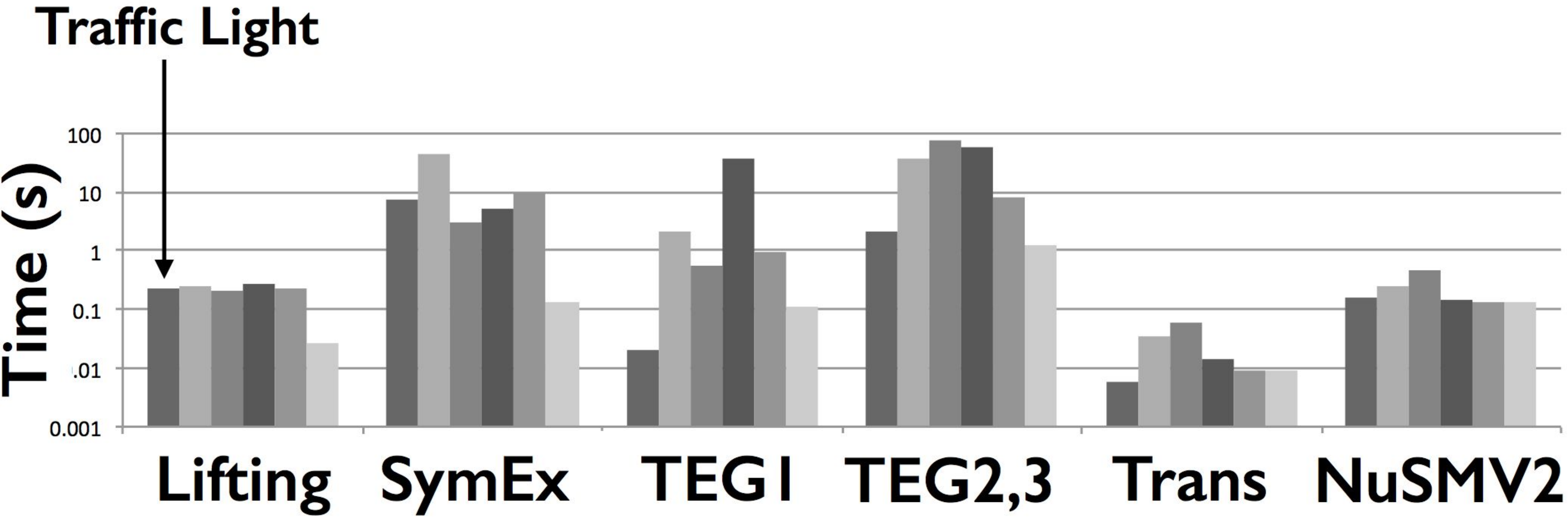
- PLCs are always running, so multiple runs of the same program touch *pre-modified memory values*
  - Symbolic execution is not enough, we need *temporal model checking*
  - Use Linear Temporal Logic (LTL) to define safety rules

$$\varphi ::= \mathit{true} \mid b \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \, \mathbf{U} \, \varphi \mid \mathbf{X} \, \varphi,$$

Temporal Execution Graph

- High level: State machine that models the symbols from step 1 and the safety properties from step 2
- Inputs: *symbolic scan cycle, LTL predicates, termination deadline*
- Steps
  - Initialize the TEG with 0 values for all PLC variables/predicates/symbols
  - Load scan cycle and predicates
  - For each predicate, evaluate if it is satisfiable from previous state
  - If so, create a new state, and determine whether or not it already exists in the graph
    - If it exists, don’t add it
    - If it doesn’t, create a state transition
- Malicious code discovery: Show to the operator an example of why the system *failed the check*

Performance


Traffic Light
Time (s)
100
10
1
0.1
.01
0.001
Lifting
SymEx
TEG1
TEG2,3
Trans
NuSMV2

Discussion