جامعة نيويورك أبوظبي
NYU | ABU DHABI

MoMA LAB

ICSPatch: Automated Vulnerability Localization and Non-Intrusive Hotpatching in Industrial Control Systems using Data Dependence Graphs

Prashant Rajput **Constantine Doumanidis** Michail Maniatakos

USENIX 2023 August 15, 2023

What are Industrial Control Systems?

- Industrial Control Systems (ICS)
  - Ruggedized systems
  - Interface with the real world
  - Examples: PLCs, SCADA systems, etc.

What are Industrial Control Systems?

- Industrial Control Systems (ICS)
  - Ruggedized systems
  - Interface with the real world
  - Examples: PLCs, SCADA systems, etc.

- Part of critical infrastructure
  - Power grid
  - Nuclear plants
  - Desalination facilities

What are Industrial Control Systems?

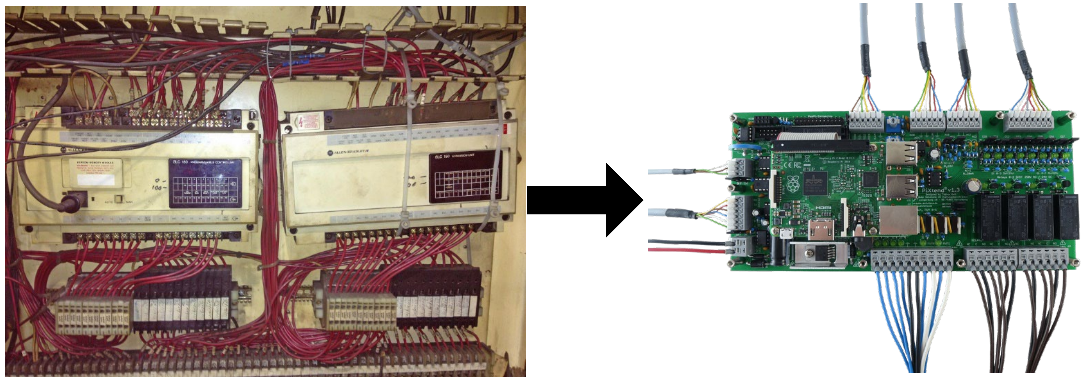
- Industrial Control Systems (ICS)
  - Ruggedized systems
  - Interface with the real world
  - Examples: PLCs, SCADA systems, etc.

- Part of critical infrastructure
  - Power grid
  - Nuclear plants
  - Desalination facilities

- ICS robustness is paramount for safety

What are Modern ICS?

- Industry 4.0 / Industrial IoT
- ICS evolve into typical computers
  - Generic third-party SoCs
  - General-purpose OS
  - Remote connections



Siemens S7-1500 PLC line, Siemens

Vendor	Devices	Variants	Notes	Firmware Groups	OS base
Schneider Electric	M241	15	Performance intensive hardware	M241, M251	VxWorks
	M251	2	Compact device for distributed environments		
	M258	6	Bulk I/O and communications	M258	VxWorks
Siemens	S7-1200	19	Entry level PLC	S7-1200	Utilizes OpenBSD components
	S7-1500	38	Feature rich, future proof	S7-1500, ET 200SP, ET 200pro	Debian Linux
	ET 200SP	3	I/O oriented control cabinet device		
	ET 200pro	9	Modular and rugged I/O device		
WAGO	PFC100	6	Compact and modular PLC	PFC100, PFC200	Linux
	PFC200	21	Powerful and feature rich PFC100		
	PFC200 (2nd generation)	18	Evolution of the PFC200	PFC200 (2nd gen.)	Linux
ABB	AC500 V3	10	Mainstream AC500 V3	AC500 V3 /eCo/S/XC	Yocto Linux
	AC500-eCo V3	12	Cost effective version		
	AC500-S V3	3	Safety automation oriented version		
	AC500-XC V3	6	Extreme operating conditions version		
Total		168			

[1] Doumanidis, C., Xie, Y., Rajput, P. H., Pickren, R., Sahin, B., Zonouz, S., & Maniatakos, M. (2023). Dissecting the Industrial Control Systems Software Supply Chain. IEEE Security & Privacy.

PLC Execution Model

- Programmable Logic Controllers (PLCs)
  - An industrial computer continuously monitoring the state of input, makes decision based on a custom program to control state of output devices

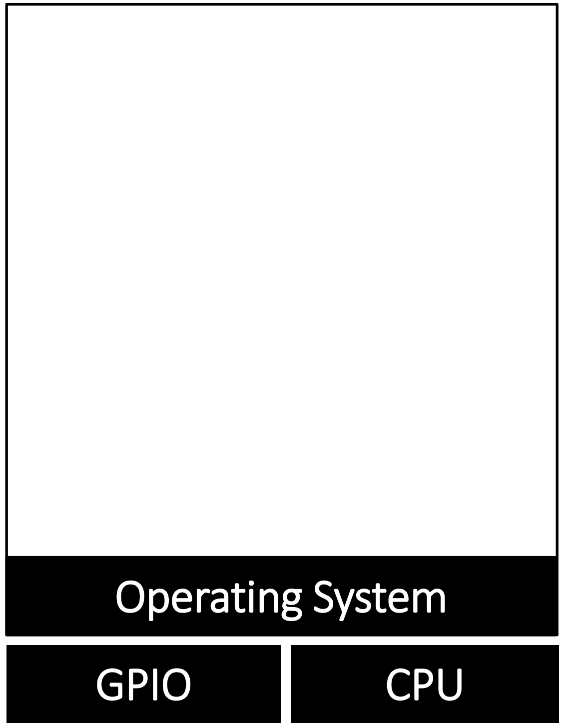


Fig. 1: Execution model for Codesys runtime.

PLC Execution Model

- Programmable Logic Controllers (PLCs)
  - An industrial computer continuously monitoring the state of input, makes decision based on a custom program to control state of output devices
- Runtime
  - Collection of components necessary for proper execution of the application binary

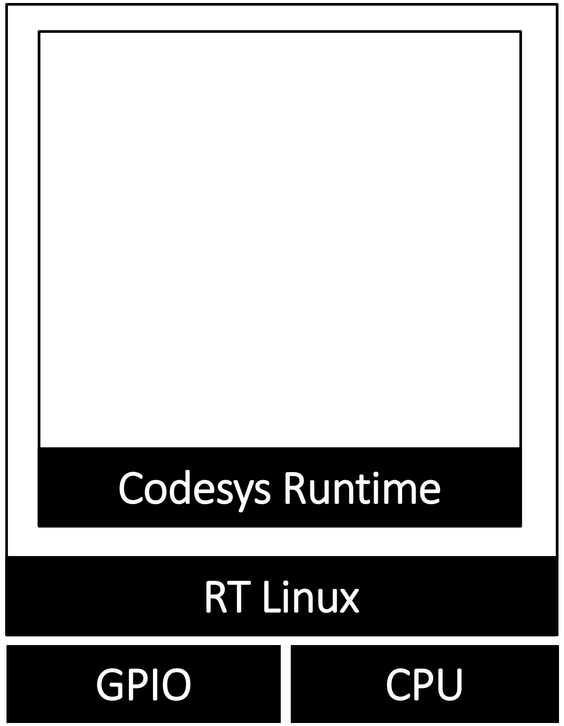


Fig. 1: Execution model for Codesys runtime.

PLC Execution Model

- Programmable Logic Controllers (PLCs)
  - An industrial computer continuously monitoring the state of input, makes decision based on a custom program to control state of output devices
- Runtime
  - Collection of components necessary for proper execution of the application binary
- Scan Cycle
  - Continuously scan program, input scan, execute program, output scan

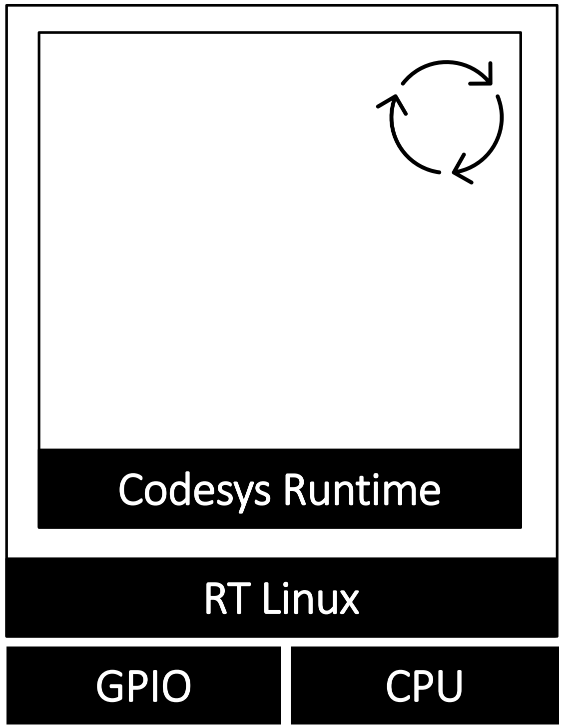


Fig. 1: Execution model for Codesys runtime.

PLC Execution Model

- Programmable Logic Controllers (PLCs)
  - An industrial computer continuously monitoring the state of input, makes decision based on a custom program to control state of output devices
- Runtime
  - Collection of components necessary for proper execution of the application binary
- Scan Cycle
  - Continuously scan program, input scan, execute program, output scan
- Control Application
  - IEC 61131-3 compliant code regulating a physical industrial process

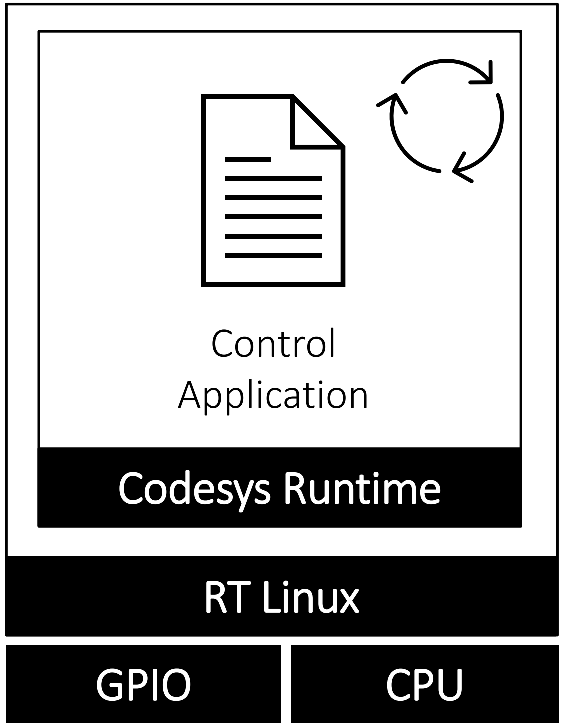
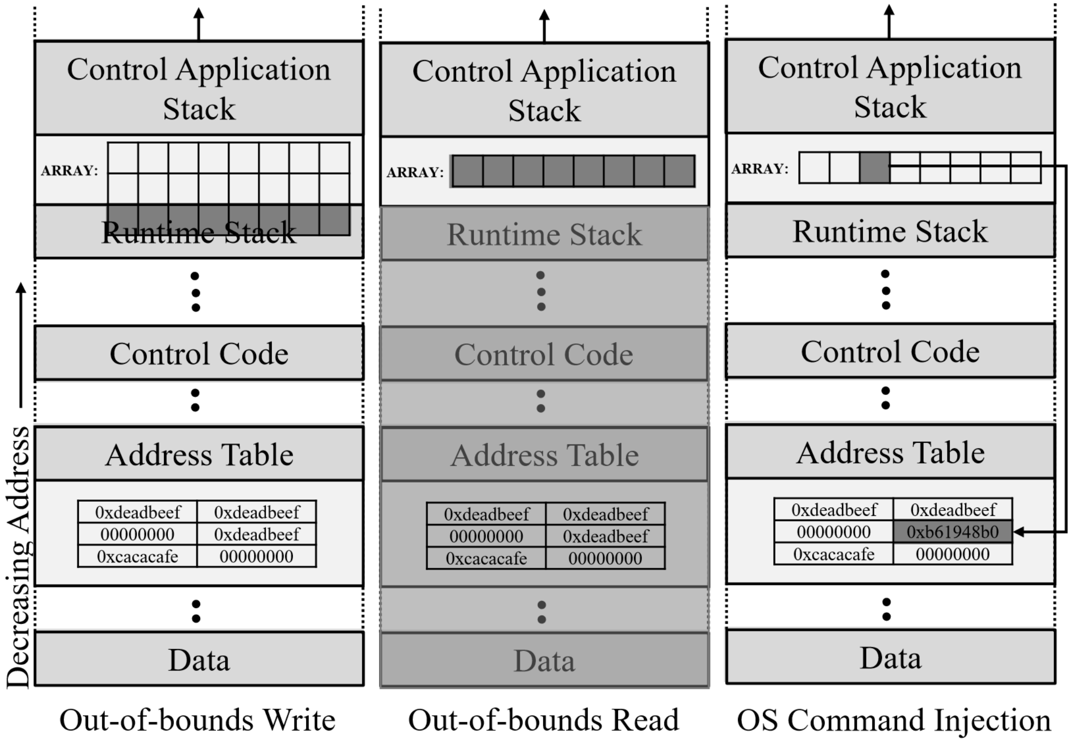


Fig. 1: Execution model for Codesys runtime.

PLC Binary Crashes!

- Crashes are signals of potentially exploitable vulnerabilities
- Vulnerabilities need to be patched
- Patching requires:
  - Vendor to produce a patch
  - PLC to be restarted
- However:
  - Vendors may not be able to produce a patch quickly (or ever)
  - PLC cannot be taken offline before next scheduled downtime



ICSPatch

- Hotpatching
  - Dynamically updating application without downloading a new version or even restarting it

ICSPatch

- Hotpatching
  - Dynamically updating application without downloading a new version or even restarting it
- Why?
  - Hotpatching for real-time applications remains unexplored, except HERA [1] and RapidPatch [2]
  - Here, application binary executes in the context of a runtime
  - Proprietary format
  - Unknown vulnerabilities
  - No upstream patch source

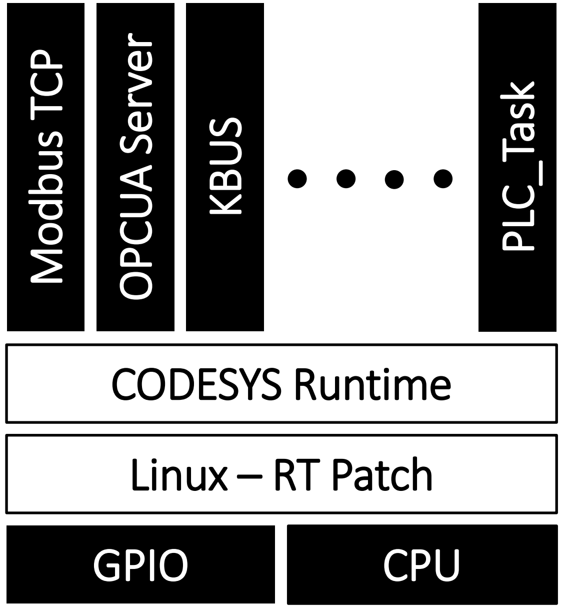


Fig. 1: Codesys-based PLC software stack.

[1] Niesler, C., Surminski, S., & Davi, L. (2021, February). HERA: Hotpatching of Embedded Real-time Applications. In NDSS.
[2] He, Y., Zou, Z., Sun, K., Liu, Z., Xu, K., Wang, Q., ... & Li, Q. (2022). RapidPatch: Firmware Hotpatching for Real-Time Embedded Devices. In 31th USENIX Security Symposium (USENIX Security 22).

ICSPatch

- Creating a diverse dataset
  - 5 sectors
  - 4 type of vulnerabilities
    - OOB write
    - OOB read
    - OS command injection
    - Improper input validation
  - 4/5 most dangerous software weaknesses for 2021 [1]

Table 1: A diverse synthetic control application dataset.

Shared Library	Imported Functions	Aircraft Flight Control				Anaerobic Dig estion Reactor				Chemical Plant				Desalination Plant				Smart Grid			
		CWE-787	CWE-125	CWE-78	CWE-20	CWE-787	CWE-125	CWE-78	CWE-20	CWE-787	CWE-125	CWE-78	CWE-20	CWE-787	CWE-125	CWE-78	CWE-20	CWE-787	CWE-125	CWE-78	CWE-20
SysMem23	SysMemSet	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	SysMemMove	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SysMem	SysMemSet	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	●
	SysMemMove	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	●	○	○	○	○
	SysMemCpy	○	○	○	○	○	○	○	○	○	○	○	●	○	●	○	○	○	○	○	○
MemUtils	MemSet	○	○	○	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	MemCpy	○	●	○	○	●	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○
	BitCpy	●	○	○	○	○	●	○	●	○	○	○	○	○	○	○	○	○	●	○	○
IEC 61131-3 code																					
Out-of-bounds array index		○	○	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○	○	●	○

CWE-787/CWE-125: Out-of-Bounds Write/Read CWE-78: OS Command Injection
CWE-20: Improper Input Validation

[1] 2021 CWE Top 25 Most Dangerous Software Weaknesses, MITRE, https://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html.

Methodology

System design overview

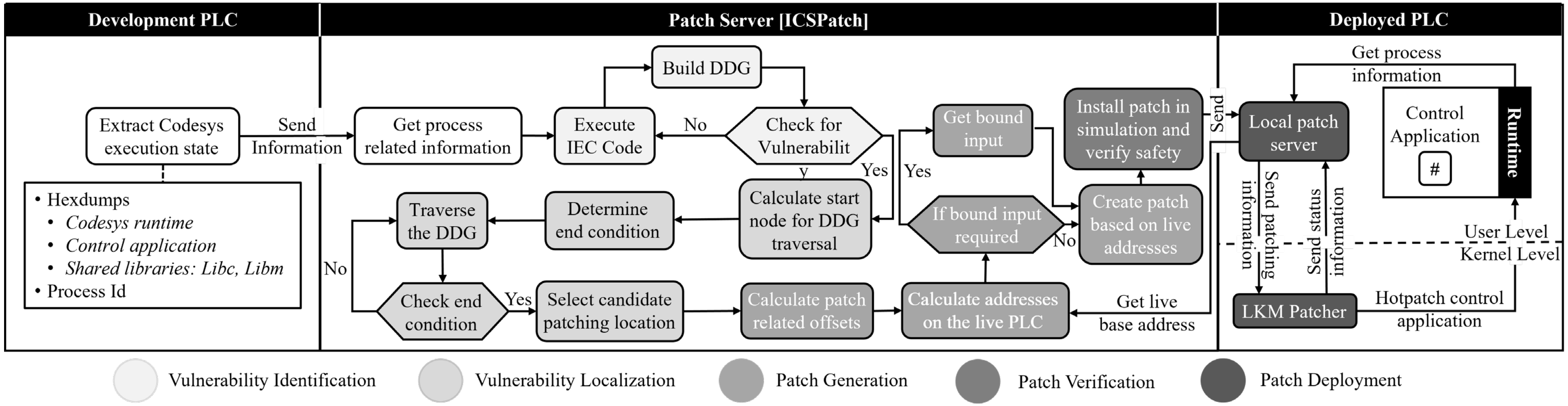


Fig. 1: ICSPatch System Design.

- **Threat Model**
  - Remote adversary with MiTM capabilities
  - Adversary limited to data injection/modification attacks
  - ICSPatch does not assume upstream patch source
  - ICSPatch assumes at least one exploit input

Methodology

System design overview

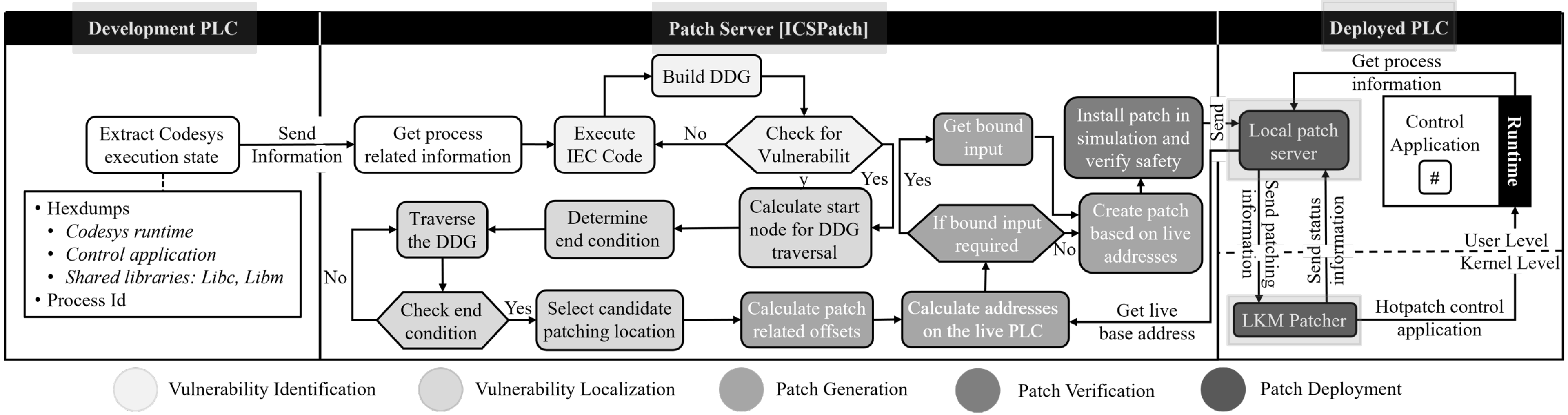


Fig. 1: ICSPatch System Design.

- **Threat Model**
  - Remote adversary with MiTM capabilities
  - Adversary limited to data injection/modification attacks
  - ICSPatch does not assume upstream patch source
  - ICSPatch assumes at least one exploit input

Methodology

Step 1: Vulnerability Identification & Localization

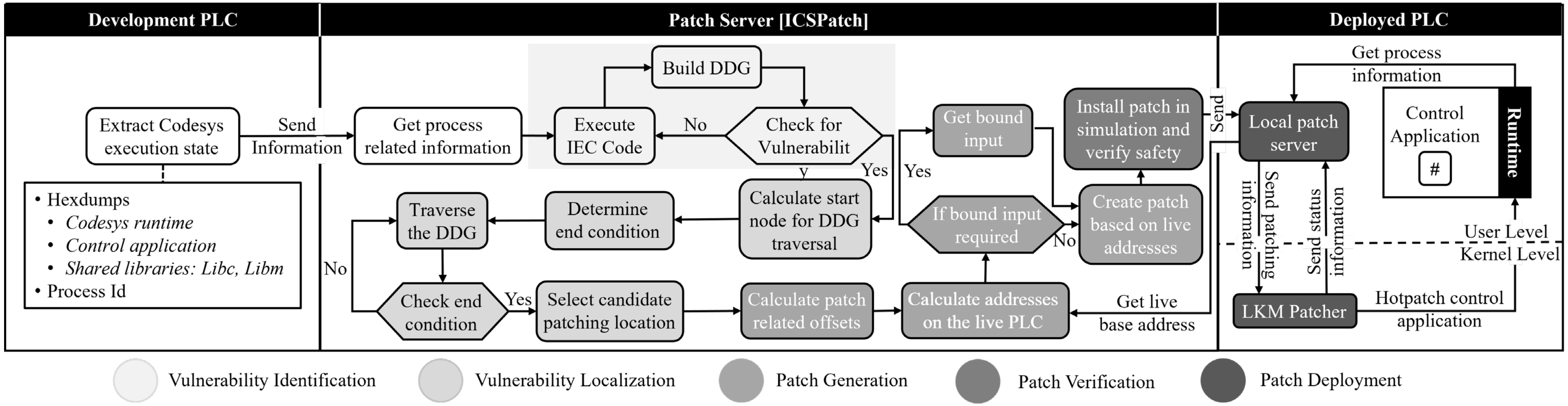


Fig. 1: ICSPatch System Design.

- **Vulnerability Identification**

OOB_WRITE_RULE (ALERT): OOB_WRITE [WRITE_ADDRESS > RUNTIME_STACK and WRITE_ADDRESS < RUNTIME_TEXT] "Vulnerability Detected."

Name	Action	Patch Identifier	Definition	Message
OOB_WRITE_RULE	(ALERT):	OOB_WRITE	[WRITE_ADDRESS > RUNTIME_STACK and WRITE_ADDRESS < RUNTIME_TEXT]	"Vulnerability Detected."
		OOB_READ	READ_ADDRESS > RUNTIME_STACK AND READ_ADDRESS < RUNTIME_DATA	
		OS_CMD_INJ	WRITE_ADDRESS > RUNTIME_STACK AND WRITE_ADDRESS > RUNTIME_ADDRESS_TABLE	

Fig. 2: ICSPatch Rule Example.

Methodology

Step 1: Vulnerability Identification & Localization

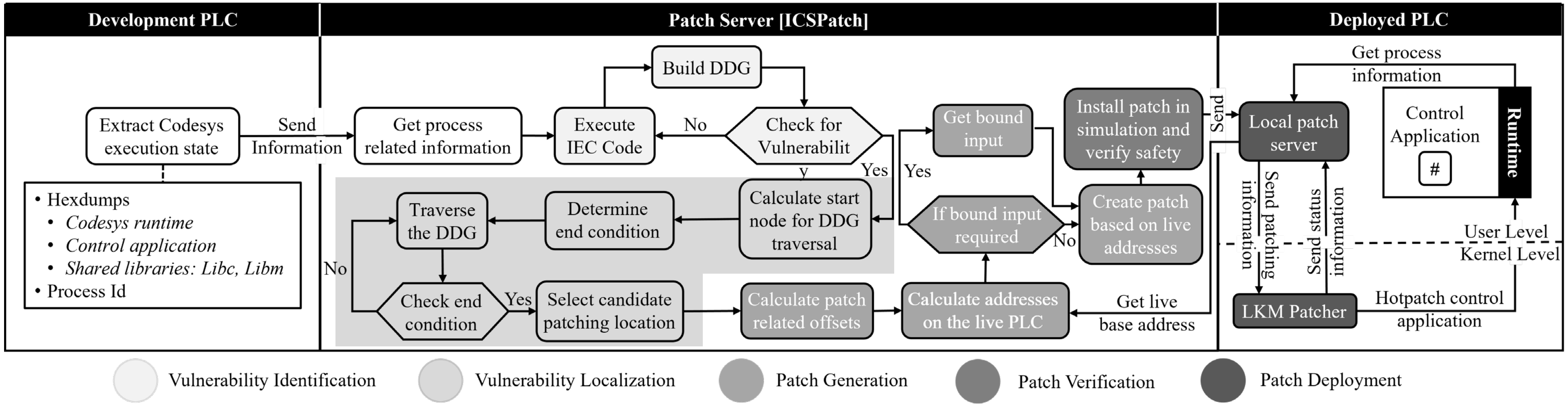


Fig. 1: ICSPatch System Design.

- **Vulnerability Localization**
  - Traverse back on the DDG
  - Locate the closest node to the boundary between control application and the runtime.

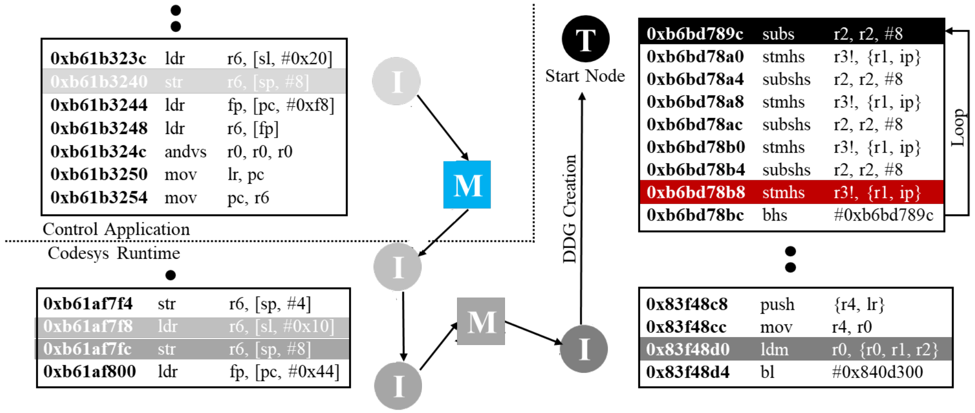


Fig. 3:Vulnerability localization in ICSPatch using Data Dependence Graph.

Methodology

Step 1: Vulnerability Identification & Localization

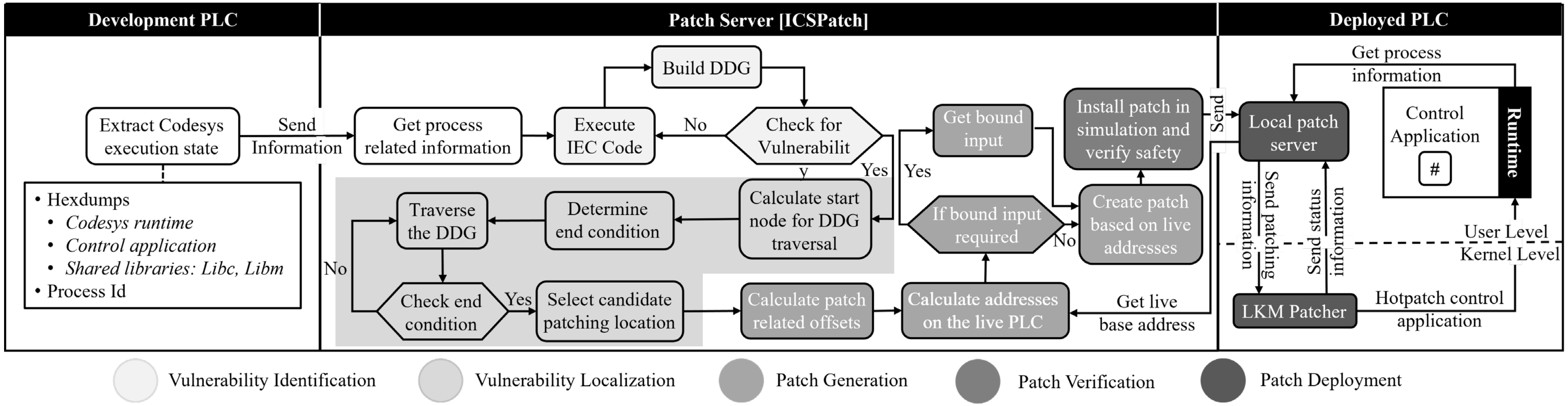


Fig. 1: ICSPatch System Design.

- **Vulnerability Localization**
  - Traverse back on the DDG
  - Locate the closest node to the boundary between control application and the runtime

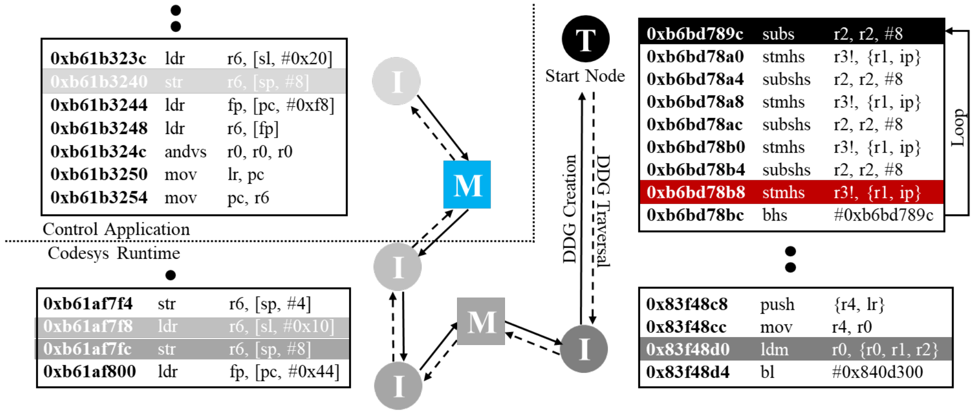


Fig. 3:Vulnerability localization in ICSPatch using Data Dependence Graph.

Methodology

Step 1: Vulnerability Identification & Localization

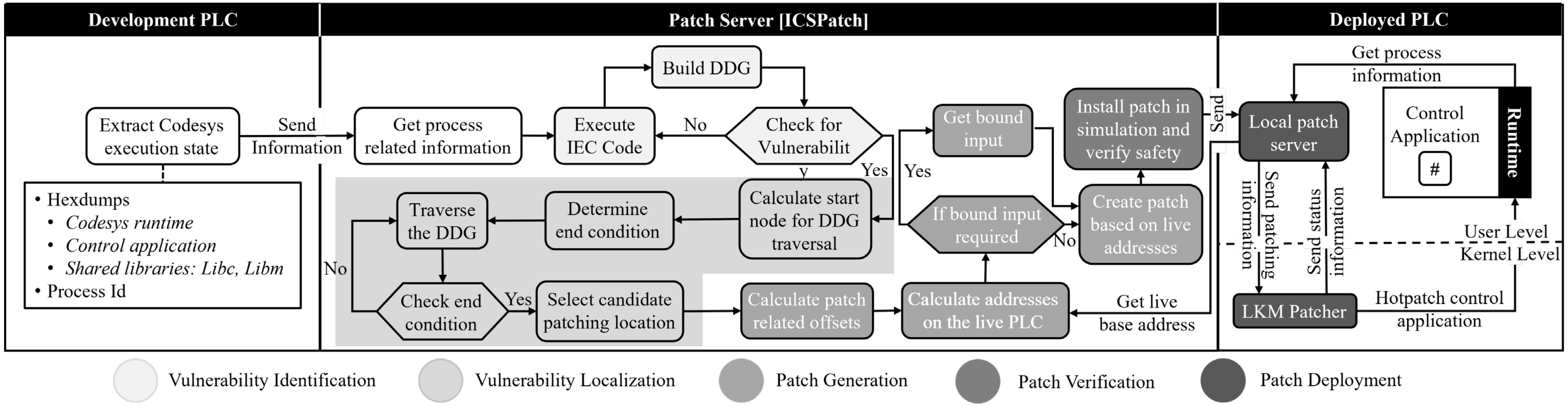


Fig. 1: ICSPatch System Design.

- **Vulnerability Localization**
  - Traverse back on the DDG
  - Locate the closest node to the boundary between control application and the runtime.

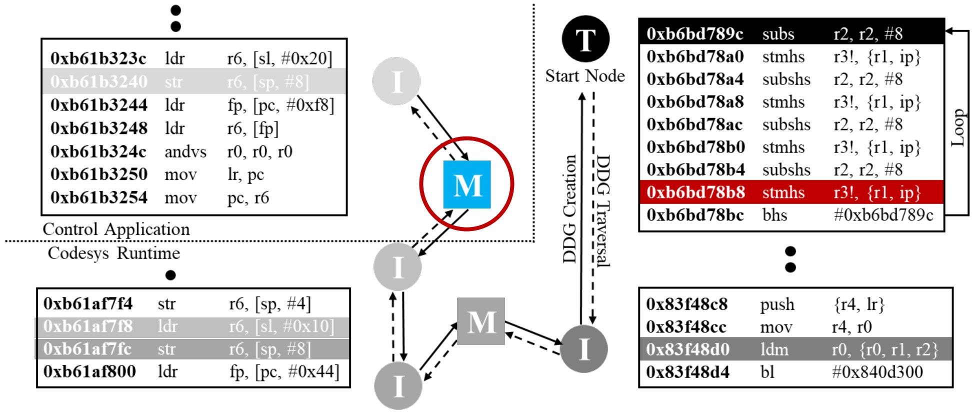


Fig. 3:Vulnerability localization in ICSPatch using Data Dependence Graph.

Methodology

Step 2: Patch Generation & Deployment

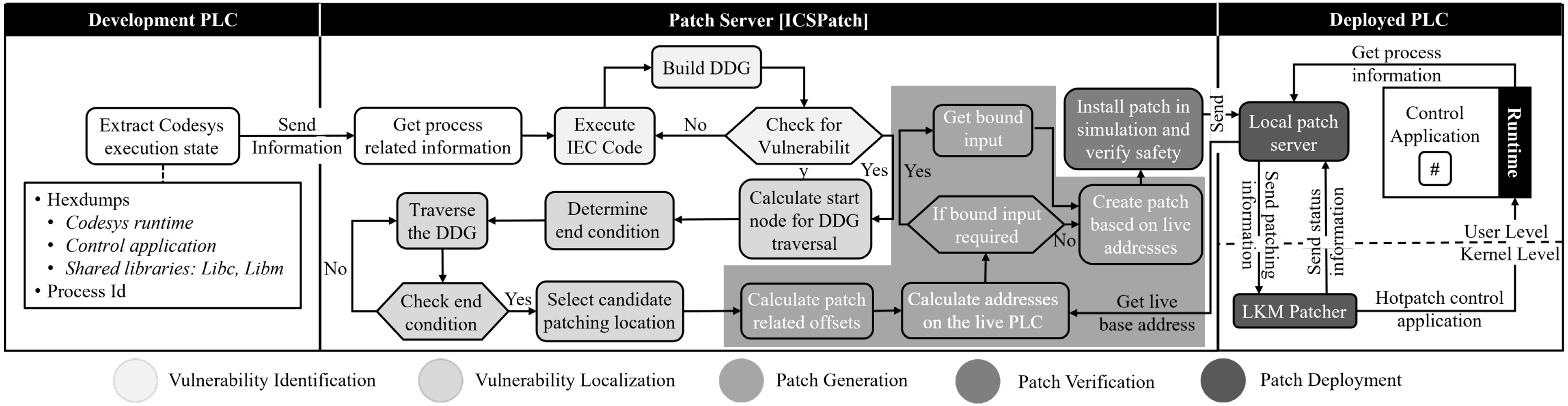


Fig. 1: ICSPatch System Design.

- **Patch Generation**
  - No upstream patch sources for control application
  - Memory related vulnerabilities require bound checking patches
  - Populate skeleton patches with:
    - Vulnerable bound memory location M
    - User defined bound
    - Next function offset into the address table

Methodology

Step 2: Patch Generation & Deployment

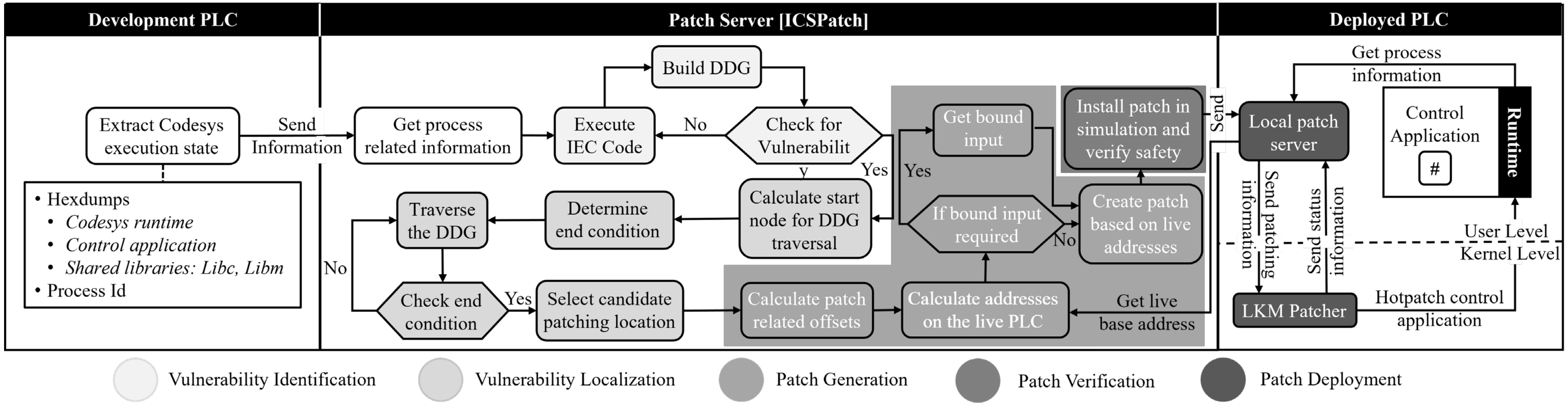


Fig. 1: ICSPatch System Design.

- **Patch Generation**
  - No upstream patch sources for control application
  - Memory related vulnerabilities require bound checking patches
  - Populate skeleton patches with:
    - Vulnerable bound memory location M
    - User defined bound
    - Next function offset into the address table
- **Patch Verification**
  - Load the patch in angr simulation instance
  - Execute and check vulnerability rulesets

Methodology

Step 2: Patch Generation & Deployment

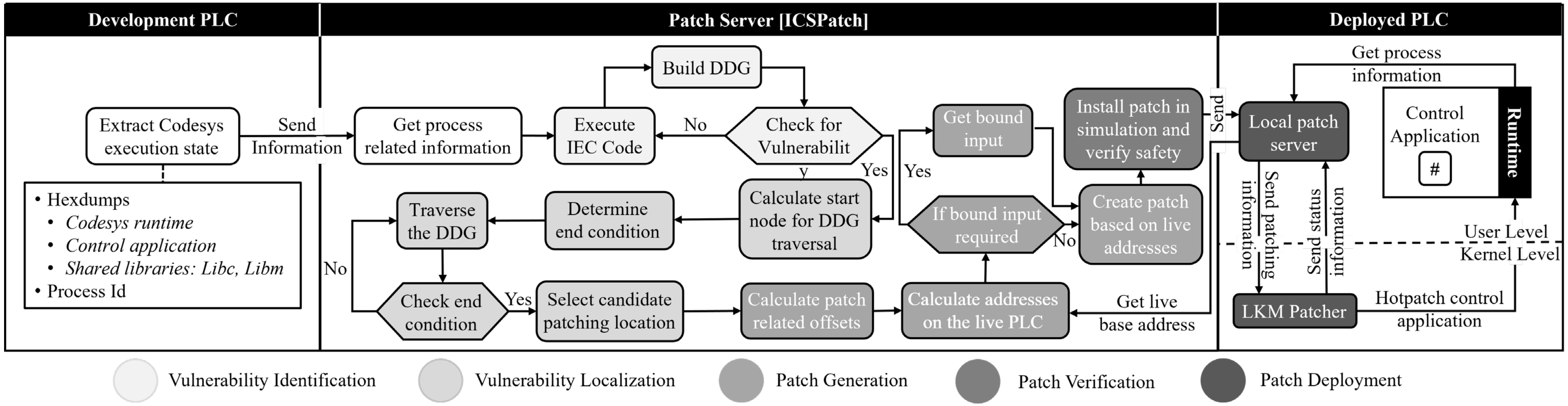


Fig. 1: ICSPatch System Design.

- **Branching in Control Applications**
  1. Load base address of address table

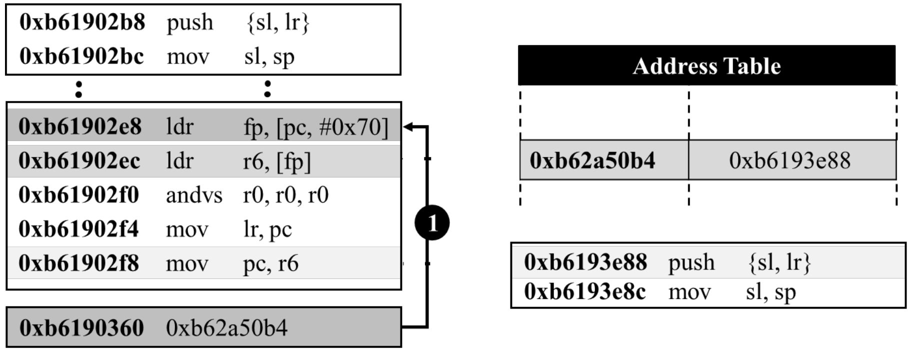


Fig. 4: Branching in Codesys compiled control applications.

Methodology

Step 2: Patch Generation & Deployment

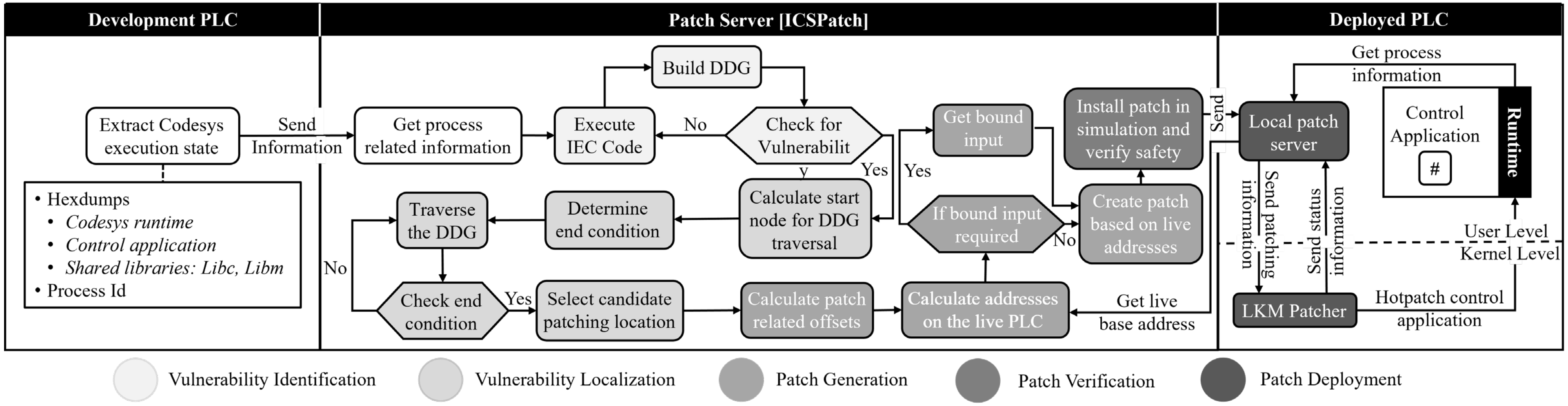


Fig. 1: ICSPatch System Design.

- **Branching in Control Applications**
  1. Load base address of address table
  2. Load the address of the next function

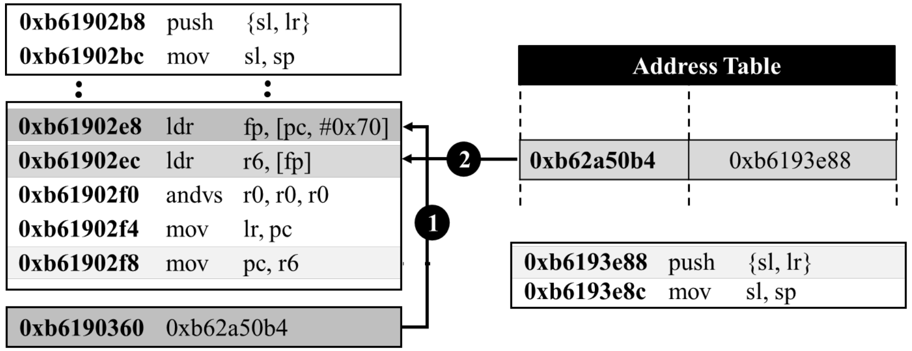


Fig. 4: Branching in Codesys compiled control applications.

Methodology

Step 2: Patch Generation & Deployment

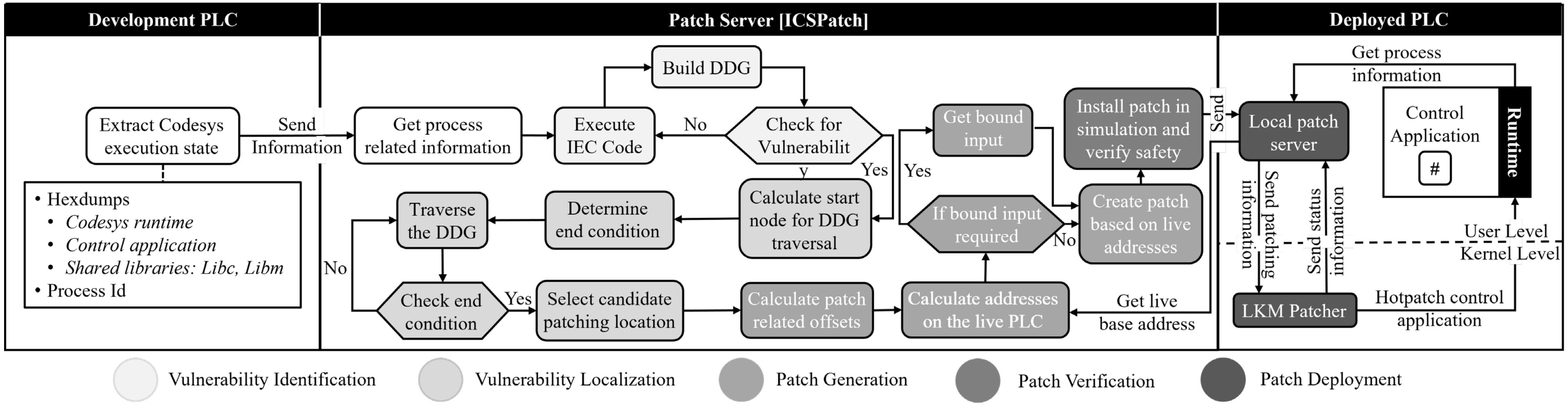


Fig. 1: ICSPatch System Design.

- **Branching in Control Applications**
  1. Load base address of address table
  2. Load the address of the next function
  3. Modify the value of the PC

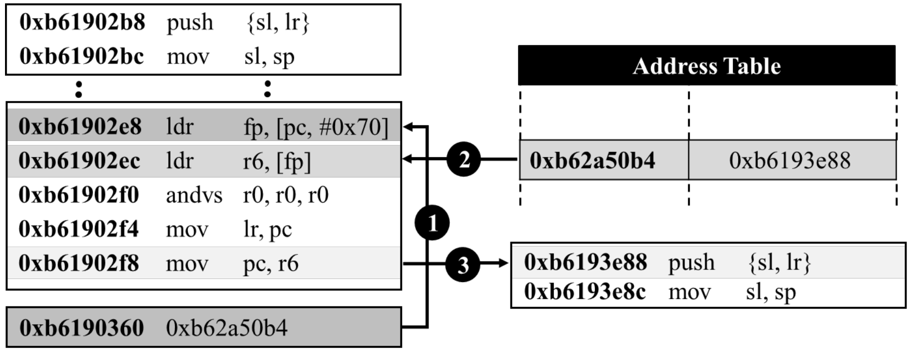


Fig. 4: Branching in Codesys compiled control applications.

Methodology

Step 2: Patch Generation & Deployment

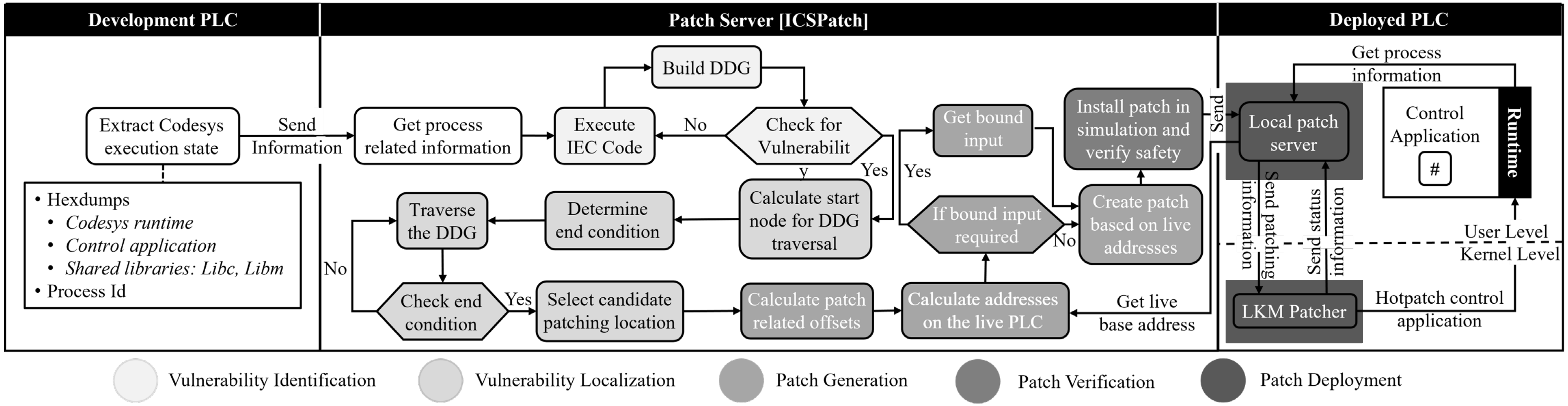


Fig. 1: ICSPatch System Design.

- **Patch Deployment**
  1. Write patch at empty memory location
  2. Write patch address into an empty address table entry
  3. Modify the offset to the base address table to load patch address (critical)

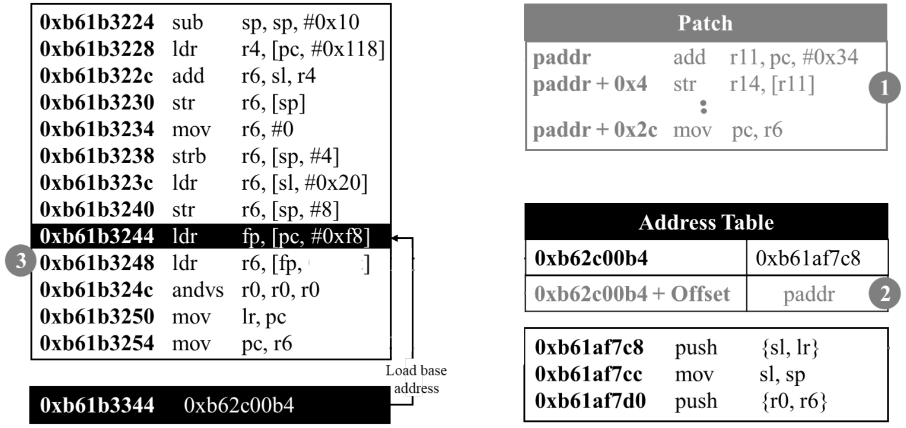


Fig. 5: Steps to modify control flow in control applications.

Methodology

Step 2: Patch Generation & Deployment

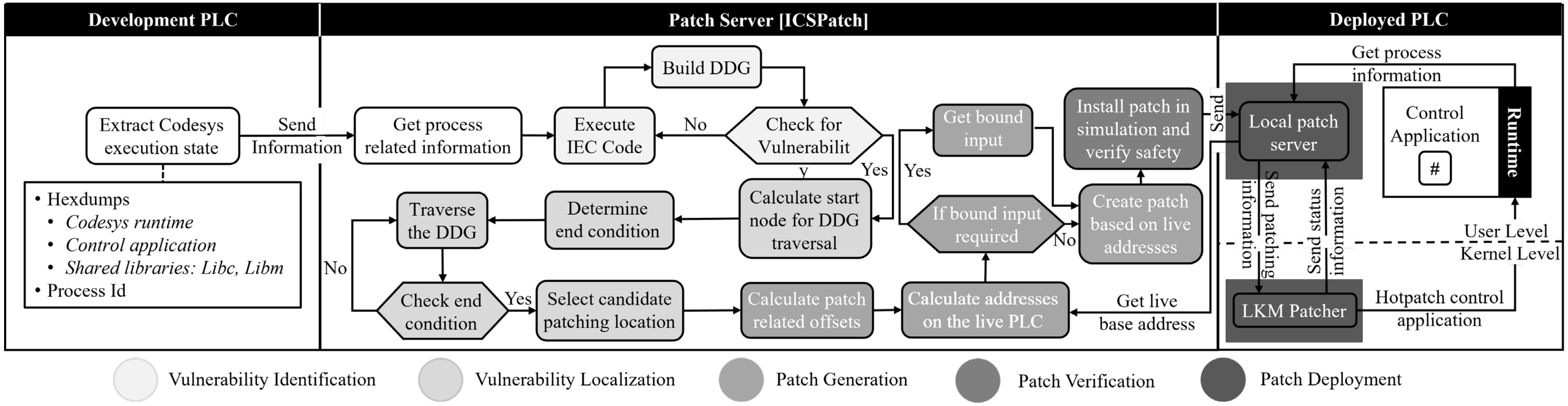


Fig. 1: ICSPatch System Design.

- **Patch Deployment**
  1. Write patch at empty memory location
  2. Write patch address into an empty address table entry
  3. Modify the offset to the base address table to load patch address (critical)

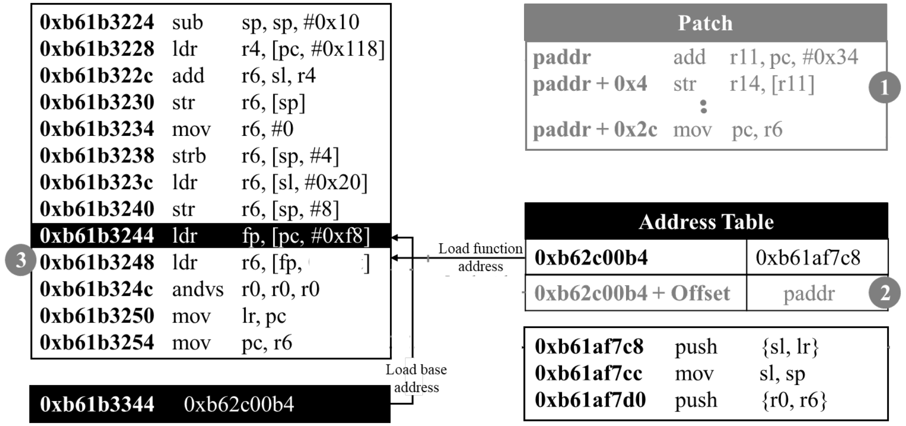


Fig. 5: Steps to modify control flow in control applications.

Methodology

Step 2: Patch Generation & Deployment

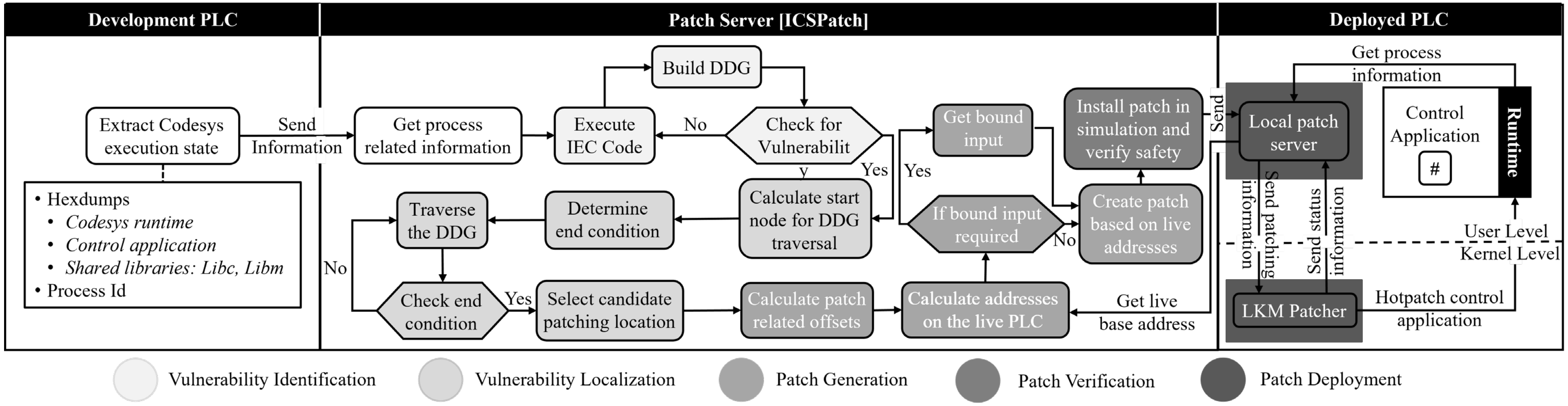


Fig. 1: ICSPatch System Design.

- **Patch Deployment**
  1. Write patch at empty memory location
  2. Write patch address into an empty address table entry
  3. Modify the offset to the base address table to load patch address (critical)

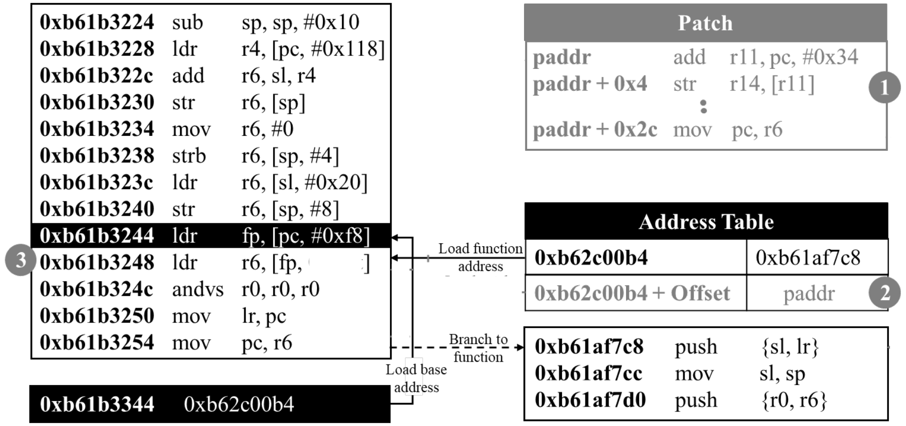


Fig. 5: Steps to modify control flow in control applications.

Methodology

Step 2: Patch Generation & Deployment

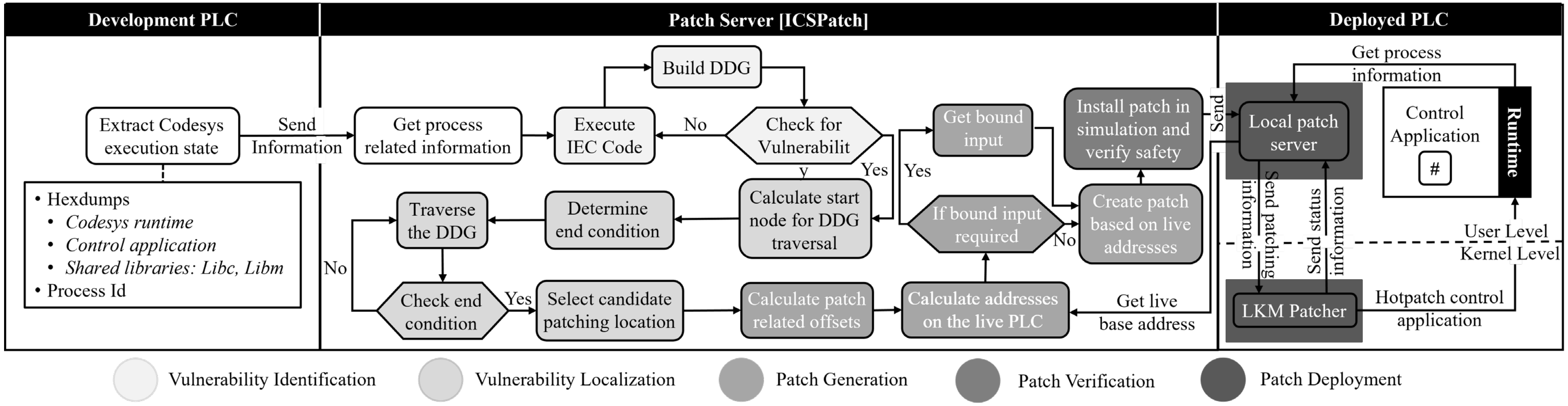


Fig. 1: ICSPatch System Design.

- **Patch Deployment**
  1. Write patch at empty memory location
  2. Write patch address into an empty address table entry
  3. Modify the offset to the base address table to load patch address (critical)

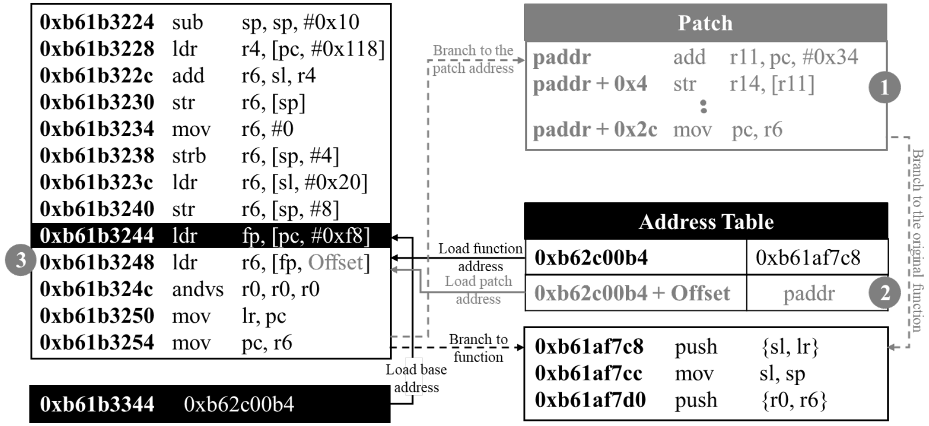


Fig. 5: Steps to modify control flow in control applications.

Experimental Results

- **Timing Overhead**
  1. **Normal:** 13 instructions (32 bits) + patch address + hook
     Exception: Does not load base address and removes ldr instruction
  2. Increased latency due to program structure (loop)
  3. Critical operation modifying execution flow by overwriting ldr offset (hook)
  4. Minimum scan cycle impact

Table 1: ICSPatch execution timings and overheads for the 24 vulnerable binaries.

Critical Infrastructure	Vulnera-bility	Time (ms) Vulnerability Localization	Patch Generation	Patch Verification (s)	Patch Deployment Total	Patch Deployment Critical (µs) ③	Mean Execution Time (µs) Pre-patch	Post-patch	Difference ④	Achieved Scan Cycle (µs)	Memory (Bytes)
Aircraft Flight Control	CWE-20	3.06	178.57	7.71	252.13	0.22	20.09	21.06	0.97	69.59	64
	CWE-787	6.72	166.39	67.18	332.73	0.3	18.05	20.04	1.99	77.2	64
	CWE-787	4.42	143.68	33.43	232.2	0.26	25.4	27.19	1.8	74.94	64
	CWE-125	4.85	178.59	28.93	252.24	0.46	21.6	21.78	0.17	74.54	64 ①
	CWE-78	1.54	203.03	9	230.17	0.3	16.61	20.5	3.89	72.87	56
Anaerobic Reactor	CWE-20	5.05	134.98	10	234.42	0.31	20.04	21.04	0.99	134.46	64
	CWE-787	3.78	126.87	2.86	232.07	0.35	17.02	17.07	0.05	71.46	64
	CWE-787	4.82	130.18	7.17	246.06	0.24	19.99	20.24	0.25	147.74	64
	CWE-787	4.44	124.95	2.16	223.7	0.22	21.32	21.37	0.05	74.49	64
	CWE-125	5.91	125.24	4.62	234.49	0.23	16.89	18.94	2.04	77.3	64
	CWE-125	4.96	221.38	169.1	236.26	0.28	23.98	28.08	4.09	152.96	64
	CWE-78	1.44	171.11	6.16	298.04	0.3	15.10	19.3	4.19	71.7	56
Chemical Plant	CWE-20	5	126	1.85	254.16	0.3	14.82	17.45	2.63 ②	67.75	64
	CWE-787	3.76	183.18	424.23	236.42	0.31	36.62	54.03	17.4	148.54	64
	CWE-125	6.61	170.95	17.83	253.94	0.23	21.37	25.8	4.42	71.41	64
	CWE-78	1.64	127.89	20.26	252.72	0.26	24.09	26.64	2.54	72.38	56
Desalination Plant	CWE-20	5.44	134.94	7.4	244	0.24	18.6348	19.88	1.25	75.82	64
	CWE-787	4.81	127.11	3.32	238.35	0.23	15.717	18.07	2.35	75.79	64
	CWE-125	4.94	139.36	11.25	241.02	0.25	19.3252	19.99	0.66	73.08	64
	CWE-78	1.52	133.81	5.5	230.13	0.26	17.483	20.52	3.04	80.5	56
Smart Grid	CWE-20	3.95	133.9	3	264.83	0.27	15.1286	17.15	2.02	65.71	64
	CWE-787	3.64	134.14	9.2	247.91	0.22	26.0833	27.02	0.94	83.69	64
	CWE-125	5.73	126.27	4.6	227.34	0.23	20.0931	22.98	2.89	97.48	64
	CWE-78	1.46	222.22	6.6	232.86	0.34	25.2406	27.31	2.07	77.12	56

CWE-20: Improper Input Validation CWE-787: Out-of-Bounds Write CWE-125: Out-of-Bounds Read CWE-78: OS Command Injection

Table 2: Detailed breakdown of ICSPatch used on Aircraft Flight Control CWE-20 vulnerable binary.

Phases	Preparation	Vulnerability Localization			Patch Generation				Patch Deployment					
Steps	Hexdump Extraction	Load Hexdumps	Control App Execution	DDG Traversal	Locate Live Addresses	Hook Creation	Patch Creation	Patch Verification	MV (Address Table)	MW (Address Table)	MV (Patch)	MW (Patch)	MV (Hook)	MW (Hook)
Device	Development PLC	ICSPatch (angr)			Deployed PLC	ICSPatch			Deployed PLC					
Time (s)	733.11	52.02	4.73	0.003	0.06	0.09	0.03	7.71	0.016	0.033	0.054	0.055	0.05	0.043

MV: Memory Verification MW: Memory Write

Experimental Results

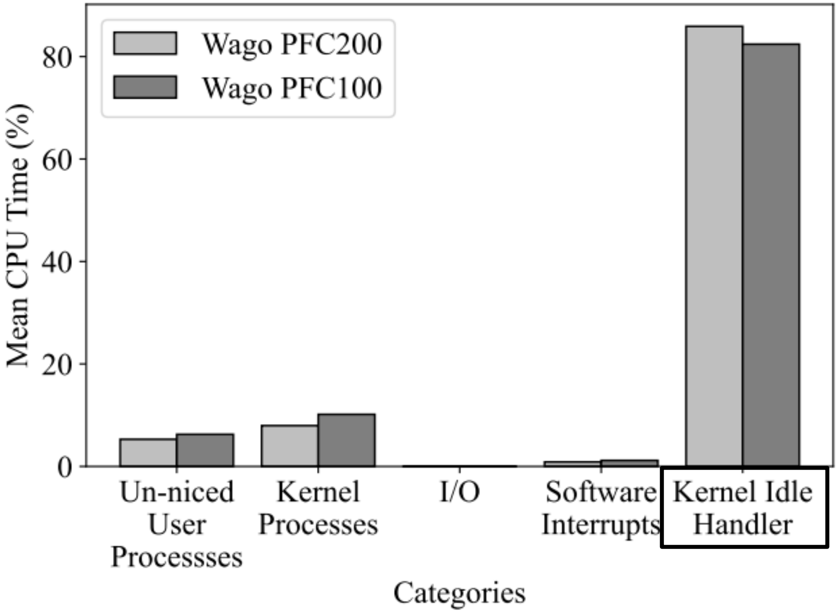


Fig. 1: CPU utilization by different operations on a PLC.

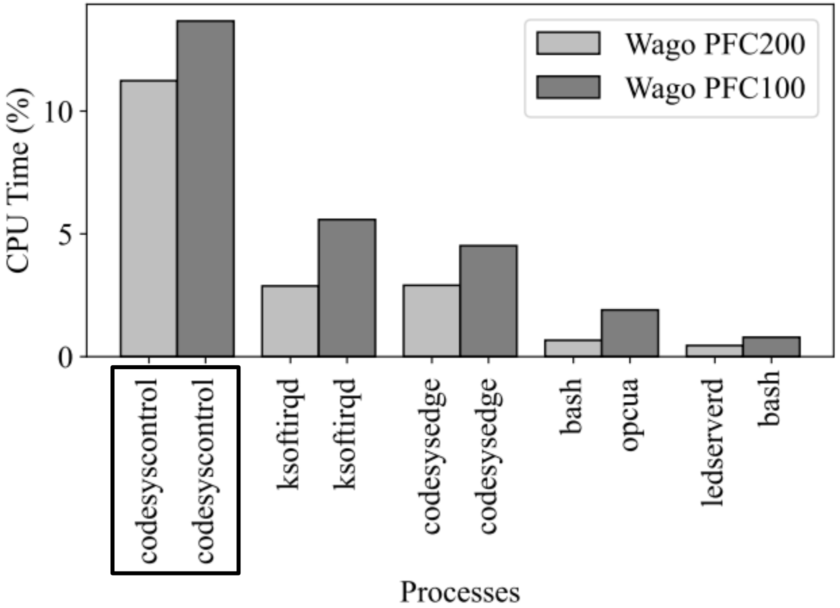


Fig. 2: CPU utilization of top 5 processes.

- Codesys runtime utilizes 14% and 11% CPU for WAGO PFC 100 and 200, respectively
- Before the critical operation
  - Change runtime's nice value to 19 (lowest)
  - preempt_disable() and local_irq_disable()

Case Study

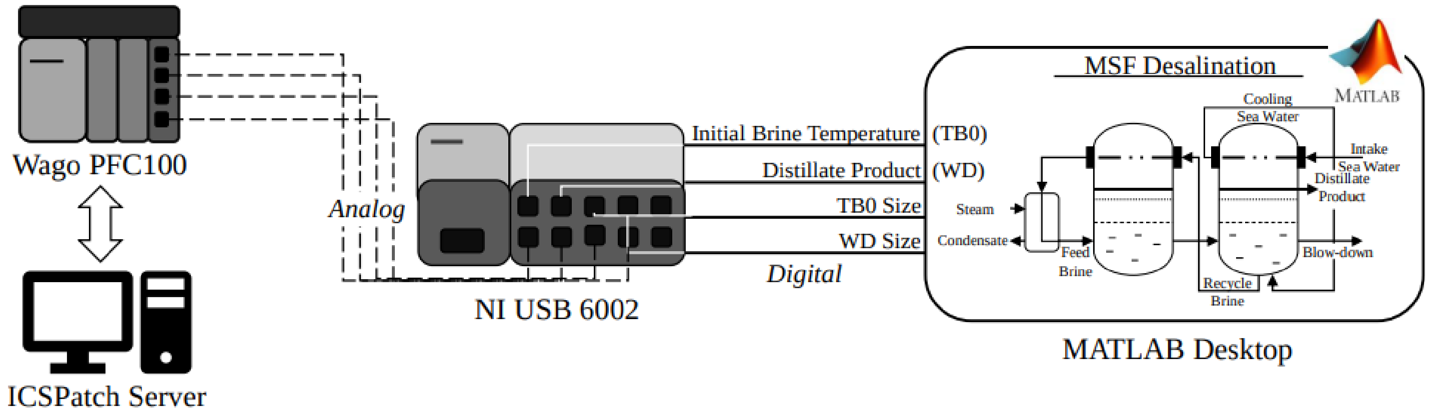


Fig. 1: hardware-in-the-Loop setup of MSF desalination plant.

- Experimental Setup
  - MATLAB Simulink model for a Multi-Stage Flash desalination plant validated against the Khubar II plant in Saudi Arabia
  - NI USB 6002, a DAQ device connects Simulink model to WAGO PFC100 PLC
  - ICSPatch server connects to the PLC

Case Study

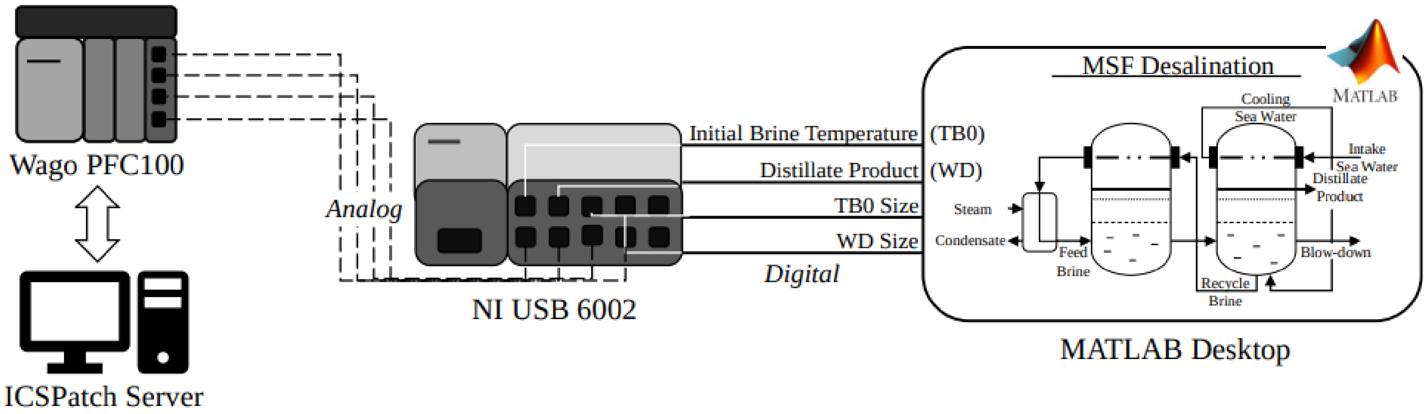


Fig. 1: hardware-in-the-Loop setup of MSF desalination plant.

- **Experimental Setup**
  - MATLAB Simulink model for a Multi-Stage Flash desalination plant validated against the Khubar II plant in Saudi Arabia
  - NI USB 6002, a DAQ device connects Simulink model to WAGO PFC100 PLC
  - ICSPatch server connects to the PLC

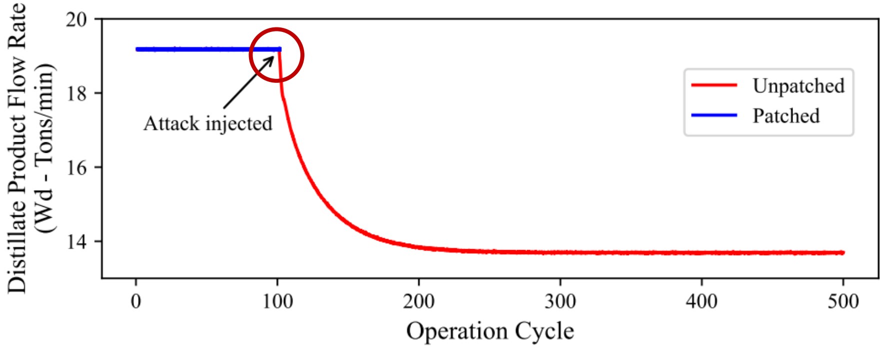


Fig. 2: Distillate product flow rate before and after patching.

Case Study

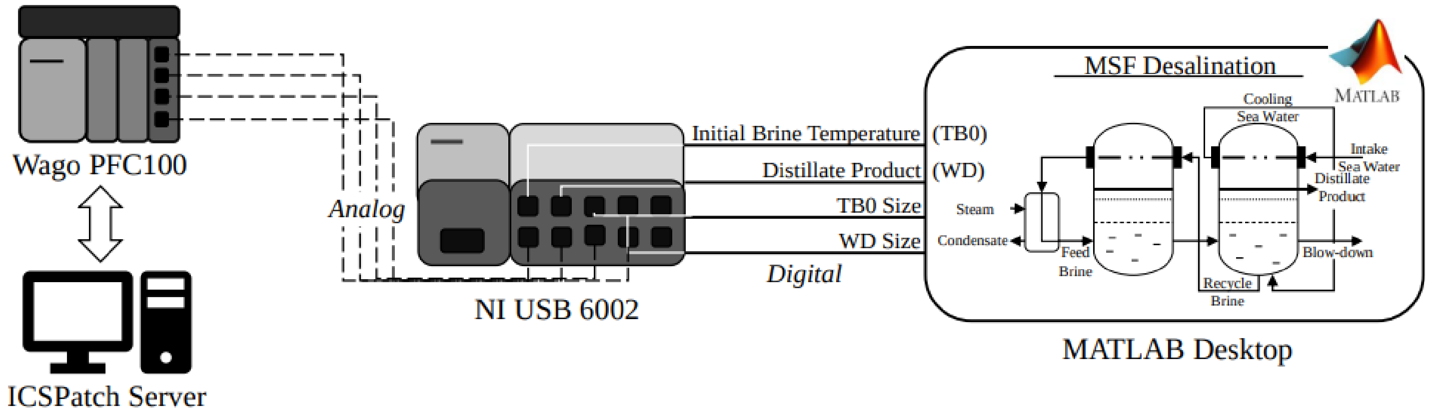


Fig. 1: hardware-in-the-Loop setup of MSF desalination plant.

- **Experimental Setup**
  - MATLAB Simulink model for a Multi-Stage Flash desalination plant validated against the Khubar II plant in Saudi Arabia
  - NI USB 6002, a DAQ device connects Simulink model to WAGO PFC100 PLC
  - ICSPatch server connects to the PLC

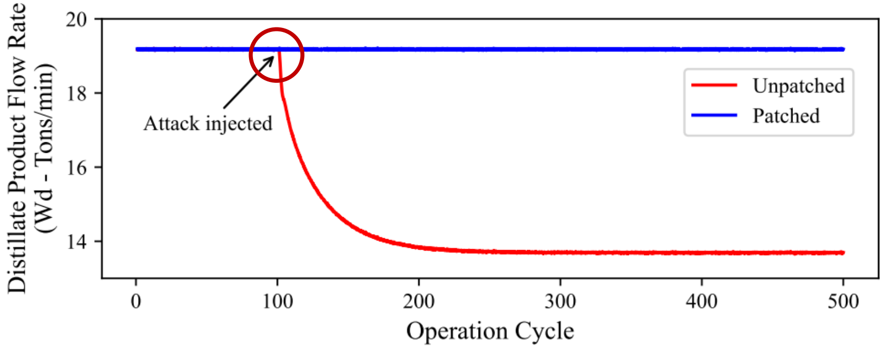


Fig. 2: Distillate product flow rate before and after patching.

MoMA LAB | nyuad.nyu.edu/momalab

Thank you. Questions?

@starlordphr
@c_smokeson
@realMoMAlab

https://wp.nyu.edu/momalab/

Paper

Code