Compromising Industrial Processes using Web-Based Programmable Logic Controller Malware

Anonymous Author(s)

ABSTRACT

We present a novel approach to developing programmable logic controller (PLC) malware that proves to be more flexible, resilient, and impactful than current strategies. While previous attacks on PLCs infect either the control logic or firmware portions of PLC computation, our proposed malware exclusively infects the web application hosted by the emerging embedded web servers within the PLCs. This strategy allows the malware to stealthily attack the underlying real-world machinery using the legitimate web application program interfaces (APIs) exposed by the admin portal website. Such attacks include falsifying sensor readings, disabling safety alarms, and manipulating physical actuators. Furthermore, this approach has significant advantages over existing PLC malware techniques (control logic and firmware) such as platform independence, ease-of-deployment, and higher levels of persistence. Our research shows that the emergence of web technology in industrial control environments has introduced new security concerns that are not present in the IT domain or consumer IoT devices. Depending on the industrial process being controlled by the PLC, our attack can potentially cause catastrophic incidents or even loss of life. We verified these claims by performing a Stuxnet-style attack using a prototype implementation of this malware on a widely-used PLC model by exploiting zero-day vulnerabilities that we discovered during our research[1]. Our investigation reveals that every major PLC vendor (80% of global market share [5]) produces a PLC that is vulnerable to our proposed attack vector. Lastly, we discuss potential countermeasures and mitigations.

CCS CONCEPTS

• **Security and privacy → Software and application security**.

KEYWORDS

ICS, Programmable Logic Controller, Malware, Malicious JavaScript

ACM Reference Format:

Anonymous Author(s). 2023. Compromising Industrial Processes using Web-Based Programmable Logic Controller Malware. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3460120.3484581

[1]These issues were disclosed to the vendor and fixed as CVE-2022-45137, CVE-2022-45138, CVE-2022-45139, and CVE-2022-45140.

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3484581

1 INTRODUCTION

Industrial Control Systems. Industrial control systems (ICSs) can be abundantly found in many critical infrastructure sectors including the electric grid, pharmaceutical, and manufacturing industries [58]. ICSs integrate IT capabilities such as monitoring and communication with physical system control [9]. This integration has resulted in today's "smart" industrial technologies such as the smart electric grid and smart manufacturing, which provide operational convenience and increased sustainability [58]. Unfortunately, the rise of smart industrial technologies has also expanded the ICS attack surface. The ICS cybersecurity market is projected to grow from $13.20B USD in 2019 to $18.05B USD by 2024 [2].

Programmable Logic Controllers. Programmable logic controllers (PLCs) are considered the core component of ICSs because they monitor sensors and manipulate actuators using local automatic control. PLCs take raw data from sensors, perform calculations based on control logic, and send commands to physical actuators to control the real-world processes [32]. Process Engineers are responsible for programming PLCs using an IEC 61131-3 compliant control language such as ladder diagram (LD) through proprietary engineering software on an engineering workstation (EWS). These EWSs compile the written PLC programs into binary executables that can be run by the processors of the PLCs in a user-code sandbox. PLCs also utilize a firmware layer to provide the low-level interface between the hardware and the control logic.

In recent years, this firmware layer has also begun to include a customizable embedded web server, which provides customers with a convenient method for accessing both administrative configurations and physical process monitoring/control via standard web browsers. This emerging trend has transformed the ICS ecosystem in profound and irreversible ways. Unfortunately, our research has uncovered that this transformation has also introduced new web-oriented security concerns that are specific to ICS environments. These security concerns are not simply the standard baggage caused by web technology in the IT domain, but rather are issues unique to the conditions caused by industrial control environments and hierarchical network architecture (i.e. Purdue Enterprise Reference Architecture [PERA]).

Real-World Attacks. While it may seem that ICSs, and PLCs in particular, are impossible targets for attackers because they are mostly disconnected from the public Internet, this notion is simply not true as demonstrated by the emergence of recent severe attacks in this domain [1, 16, 17, 20, 31, 35]. The Stuxnet [20, 35] worm targeted Iranian Uranium enrichment facilities in 2010. A few years later in 2015 and 2016, the Ukrainian power grid experienced two widespread blackouts caused by the BlackEnergy 3 malware [16, 31]. More recently, Triton (i.e., "the world's most murderous malware") [17] targeted a Saudi petrochemical plant in 2017, where the malware disabled safety instrumented systems (SISs) of the plant to cause sabotage in the underlying physical

process. These real-world examples of successful ICS attacks show that persistent bad actors are able to infiltrate segregated industrial networks using a variety of techniques (e.g., Out of Band malware infections, malicious USB drives, insider threats, etc).

Existing PLC Malware and Shortcomings. The ultimate goal of many ICS attacks is to somehow infect PLCs with malicious software (i.e., "PLC malware"). This is usually done via the final payload of an advanced ICS attack (e.g., Stuxnet [35]). PLCs are typically thought to only run software at two different levels: firmware and control logic. This preconception has engendered numerous research works and real-world attacks exploring malware implemented at both levels with firmware rootkits (e.g., HARVEY [21]) implemented in assembly code and control logic malware (e.g., LLB [28]) implemented in LD or another PLC control language. Fortunately for ICSs, both of these approaches have substantial drawbacks that make them impractical for casual adversaries. Such drawbacks include infection difficulty (e.g., requiring physical or network access), fully-offline operation (e.g., trapped in segregated industrial networks), platform dependence (e.g., requiring model-specific payloads), and low-persistence (e.g., trivially erased with factory resets).

Proposed Web-Based PLC Malware. In this paper, we introduce a new strategy for developing PLC malware that infects the front-end web layer with malicious JavaScript code. This malware, which we call Web-Based (WB) PLC malware, is fundamentally different than prior approaches and overcomes all of the drawbacks of those strategies. Our WB PLC malware resides in PLC memory, but ultimately gets executed by various browser-equipped devices throughout the ICS environment. From there, the malware uses ambient browser-based credentials to interact with the PLC's legitimate web APIs to attack the underlying real-world machinery. Our paper demonstrates that this type of malware is much easier to deploy against a real-world ICS, is capable of online operations, is largely platform independent, and achieves extremely high levels of persistence.

Consumer vs Industrial Embedded Web Servers. Prior work has shown that consumer "Internet of Things" (IoT) devices such as printers and home routers may also incorporate embedded web servers for ad-hoc administrative control [13]. While these household embedded web servers do introduce their own security concerns, they are primarily limited to basic entry-point attacks such as weak authentication and default passwords because these web servers typically only host simplistic vendor-authored setup wizards used for an initial 1-time configuration [27]. On the contrary, embedded web servers in the ICS domain are used for continuous monitoring and control via programmable web applications consumed by dedicated client hardware (e.g., WAGO eDisplay! 7300 Microbrowser). This unique utilization of embedded web technologies introduces a new attack vector not applicable to consumer devices - *persistent and covert front-end code execution*. In the ICS domain, malicious front-end code can be pushed to a programmable controller through the legitimate channels discussed in Section 4.2 and perpetually executed on a multitude of browser-equipped devices throughout the industrial network [47]. Table 1 summarizes the key differences between embedded web technology in the IT domain vs ICS domain and illustrates how PLCs are uniquely susceptible to web-based malware attacks.

Table 1: Embedded Web Technology in IT vs. ICS Domains

	Web Server Purpose	Front-End Code Author	Web Client	Web Attack Vector
IoT	Initial 1-time Setup	Device Manufacturer	Browser on Personal Device	Standard Web Vulnerabilities
PLC	Continuous Monitoring & Control	Customer & Device Manufacturer	Dedicated Hardware	Persistent and Covert Code Execution

Contribution. Our main contributions are as follows:

(1) We introduce the concept of WB PLC malware, which proves to be more flexible (e.g., platform-agnostic), resilient (e.g., persists across factory resets), and impactful (e.g., physical sabotage w/ real-time Command & Control) than prior PLC malware infections (control logic or firmware);

(2) We developed a cross-platform framework that outlines how methodically compromising embedded PLC web servers can sabotage industrial processes;

(3) We implemented a prototype of WB PLC malware, dubbed *Iron Spider*, on a widely-used PLC model to show its effectiveness compared to existing PLC malware techniques in a Stuxnet-style attack. *Iron Spider* exploited four zero-day vulnerabilities that we discovered during our research (CVE-2022-45137, CVE-2022-45138, CVE-2022-45139, and CVE-2022-45140);

(4) We propose practical countermeasures to mitigate the risk of the developed attacks or significantly reduce their damaging consequences;

Furthermore, we experimentally verified that every PLC model included in our study, namely Siemens S7-1200, Schneider TM241C, Allen-Bradley MicroLogix 1400, Mitsubishi MELSEC-F, GE/Emerson RX7i, and WAGO 750 (these vendors account for over 80% of global PLC market share [5]), is vulnerable to some sort of WB PLC malware. The rest of this paper is organized as follows. Section 2 discusses background information about ICS networks and PLCs operations. Section 3 introduces our proposed WB PLC malware and compares it to related work. The details about our multi-stage attack method are given in Section 4. Section 5 presents our experimental results and performance evaluations. Finally, Section 6 is the conclusion of the paper.

2 BACKGROUND

ICS Network Architecture. An understanding of ICS network architecture is needed to appreciate the unique characteristics of our proposed malware. Figure 1.A shows the most common architecture, PERA, which represents the various layers of ICS networks, separated into functionally distinct groups [67].

At the top, we have Level 4/5 where the primary business functions occur. This layer is typically contained within the IT/business network and is connected to the public Internet through a firewall. This level provides business direction and orchestrates manufacturing operations.

In Level 3, the production workflow is managed in remote control centers. This layer consists of data historians to record operations

data as well as EWSs and remote Human-Machine Interfaces (HMIs) that program and monitor local controllers (e.g., PLCs). This layer poses a significant challenge to network isolation because it often contains "dual-homed" devices with simultaneous connectivity to both the IT/Business network and the segregated industrial network [11].

In Level 2, Supervisory Control and Data Acquisition (SCADA) software and local HMIs are located within a geographically close distance to the physical plant. These HMIs are again used to monitor and control the underlying physical processes of PLCs. Devices in this layer and below are exclusively connected to the industrial network, thus typically do not have any connection to the public internet.

In Level 1, local controllers such as PLCs perform sensing and manipulation of physical processes using sensors and actuators with a closed-loop control structure. Finally, Level 0 defines the actual physical processes.

Emerging Web-Based ICS Services. Modern PLCs run a variety of network services in their firmware layer such as a Modbus TCP Server, a DHCP client, and an SNMP agent. Some of these services are utilized by complimentary ICS systems (e.g., SCADA) while others are available for basic networking configuration (to establish an IP address, etc).

A relatively recent addition to the list of PLC network services is the embedded web server. In the early 2010's, PLCs hosted modest admin portal websites that consisted of mostly static HTML pages to display configuration details. Nowadays, PLCs host complex and customizable Single Page Applications (SPAs) with a suite of web-based APIs to manage nearly all PLC operations, including physical process monitoring and control. This functionality has become so ubiquitous in the ICS ecosystem that virtually every major PLC vendor today includes an embedded web server in their flagship product [57], and these web servers tend to gain additional capabilities with every firmware update [52, 64]. A clear advantage of this web-based architecture is that any browser-equipped device can now configure and control the PLC (a task that previously required proprietary engineering software and clunky HMI clients [47]). This design has resulted in web browsers being abundantly found in all layers of modern ICS environments [48]. Legacy ICS equipment that communicated over serial protocols have been replaced with single-purpose microbrowser touch screens and even tablets or smartphones. Many of these devices display the web application 24/7 using a mounted display panel [56].

While the advantages of hosting web-based services are undeniable, they do blur the lines between the conceptual layers proposed by the original 1990's PERA model. Today's PLCs still live in level 1, however their web interfaces are utilized by HMIs in level 2 and EWSs in level 3. Furthermore, external front-end dependencies, such as Content Delivery Network (CDN) hosted JavaScript/CSS files, are loaded from the public internet, making them traverse level 4/5.

Empirically Measured ICS Web Trends. To give the reader confidence that web technology in the ICS domain is in fact gaining attention from both PLC vendors and customers, we performed a series of empirical studies. Our studies, combined with published data by industry leaders [48] and Internet-wide statistics [7], demonstrate that web applications do provide a realistic attack surface in modern PLCs.

We independently confirmed that embedded PLC web servers are indeed gaining functionality over time by analyzing the total Source Lines of Code (SLOC) and cyclomatic complexity [24] of the web-related codebases of multiple unpacked WAGO firmware update images. Source code size and program complexity are common metrics in software analysis to gauge the capabilities/functionality of a given application [40]. Firmware version 3.0.39 (released May 2019) contained 2,194 total SLOC (587 JS; 1,607 PHP) and an aggregate cyclomatic complexity score of 199 (82 JS; 117 PHP) while version 03.09.05 (released in March 2022) contained 15,086 total SLOC (12,471 JS; 2,615 PHP) and an aggregate cylomatic complexity of 3,779 (3,657 JS; 122 PHP). This data shows that over the past 3 years, the web application codebase has grown by over 688% and increased in complexity by over 7,581%. All source code analyzed in our study is used solely by the embedded web application, thus corroborating the claim that PLC vendors are actively introducing additional functionality to their on-board web applications.

We also experimentally verified that customers are increasingly using these embedded web servers by performing a modest longitudinal survey of internet-facing devices using the Shodan [49] search engine. We analyzed the publicly-reachable population of three widely-used PLCs (WAGO 750, Siemens S7-1200, and AB MicroLogix 1400) from June 2017 to Septemeber 2022. We discovered that on average, embedded web server usage has increased 212.66% over the past 5 years, even though overall PLC population has only increased by 12.15%. We came to this conclusion by observing the rate at which the web servers became internet-facing (discovered using web fingerprints such as SSL issuers and favicon hashes) and comparing it to the rate at which the SNMP services appeared online (discovered using keywords on the 161 UDP port). This data provides strong evidence that customers are indeed enabling and using these web servers. Our results are in-line with previously published data regarding the adoption of web technology [7, 48] and further supports the intuitive claim that PLC customers are embracing web-based app design. The remainder of this paper will investigate the new attack vector enabled by this emerging trend.

3 RELATED WORK & WB ADVANTAGES

This section compares our proposed WB PLC malware to existing PLC malware categories. We aim for this paper to provide compelling evidence that due to the emergence of powerful PLC web services, system-level compromise of the PLC is no longer necessary to successfully attack ICSs.

Traditional PLC Malware and Shortcomings. We use the term "traditional PLC malware" to describe malicious PLC control logic (CL) programs (e.g., LLB [28]) and malicious PLC firmware (FW) images (e.g., HARVEY [21]). As discussed in Section 1, these two strategies are the only publicly known methods of infecting a PLC with malicious software. Figure 1.B illustrates traditional PLC malware's infection scenarios and execution environment in the context of the PERA model.

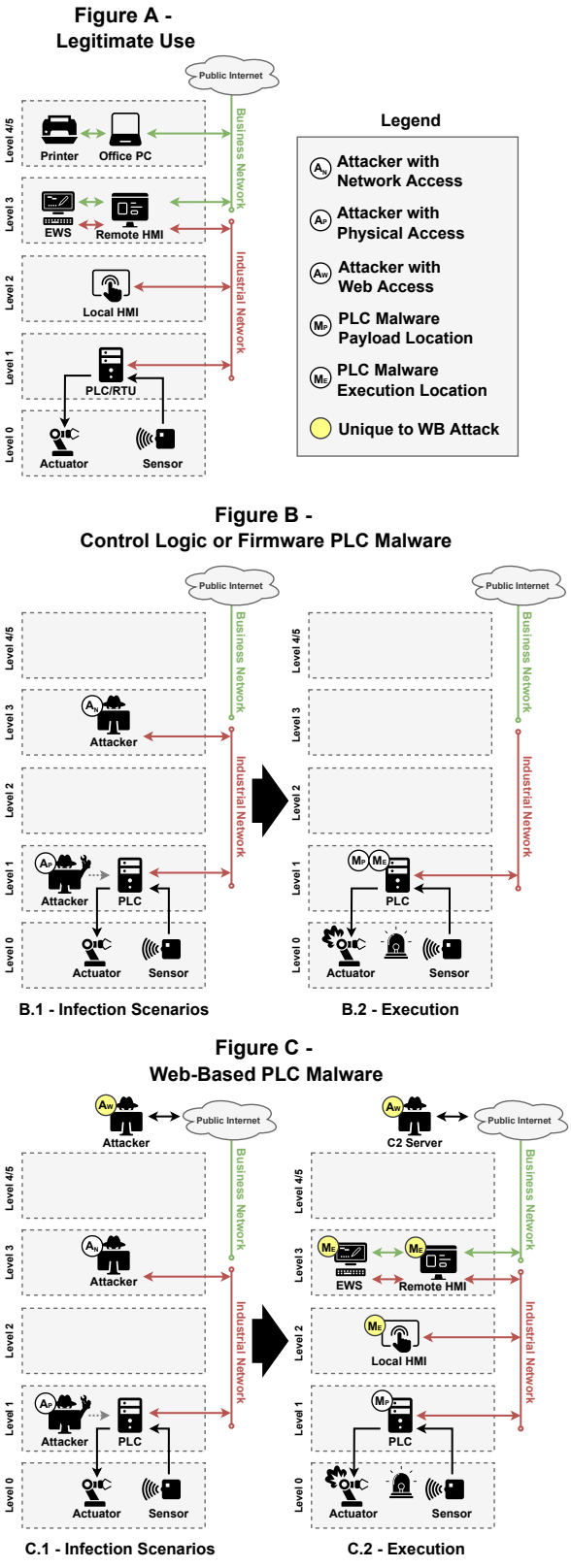

Figure 1: PLC malware in the PERA Model

Traditional PLC malware infections are possible from two distinct vantage points - *network access* (levels 1-3) and *physical access* (level 1), as shown in Figure 1.B.1. For example, a malicious control logic program may be downloaded via a compromised EWS (à la Stuxnet) or a malicious firmware update may be initiated using physical access to an exposed JTAG port (à la HARVEY [21]). Both of these scenarios require sizable prerequisites to be successful in-practice (e.g., coupled with Windows malware or launched by human assets).

Once the target PLC has become infected with traditional PLC malware, the code is both *stored and executed* on the PLC device within level 1, as shown in Figure 1.B.2. This constraint requires traditional PLC malware to abide by the strict hardware requirements of a real-time operating system (RTOS) with a modest CPU and limited network connectivity. For example, the firmware malware, HARVEY, required tedious model-specific firmware reverse engineering and binary instrumentation to carefully inject instructions in subroutines outside of the time-critical scan cycle and conform to real-time expectations of the control loop [21]. Even more restricted, control logic malware runs as user-code contained in an execution sandbox (often referred to as "jail") and only has access to specific memory regions and limited control logic APIs provided by the vendor [30]. In either case, the code exclusively runs in level 1, trapped in the segregated industrial network without a public internet connection.

Proposed WB PLC Malware and Benefits. Following the recent growing trend of web-based ICS functionalities, we present a new method for infecting PLCs with malicious software that results in a radically different type of PLC malware than the previous approaches. This malware, which we call *Web-Based (WB) PLC malware*, compromises the web application hosted by PLCs' embedded web servers with malicious JavaScript code. This code ultimately gets executed by various browser-equipped devices throughout the ICS environment (not the PLC itself as in the case for CL and FW malware). During execution, the malware uses ambient browser-based credentials to interact with the PLC's legitimate web APIs to attack the underlying real-world machinery. Figure 1.C illustrates WB PLC malware's infection scenarios and execution environment in the context of the PERA model.

As shown in Figure 1.C.1, WB malware introduces a new infection scenario not possible with previous attacks. In this scenario, which we call "*Web Access*," the attacker lures a dual-homed ICS operator within level 3 to view a malicious website. This scenario does not require the EWS to be compromised (i.e., running a malicious binary) but rather simply *viewing* an attacker-controlled website. This scenario originates from the public internet, above level 4/5, and uses web technologies to pivot into the private industrial network. An example attack from this scenario is a malicious website that exploits a Cross-Origin Resource Sharing (CORS) misconfiguration vulnerability to transfer a malicious User-defined Web Page (UWP) to the PLC's embedded web server. Additionally, the two access levels used by traditional PLC malware (*network & physical*) are also viable infection methods for WB PLC malware. For example, a malicious UWP can be downloaded via an ICS Protocol or a malicious web-based GUI may be installed via an SD Card. More details about WB infection mechanisms are presented in Section 4.2.

Table 2 summarizes the viability of different access levels per PLC malware category.

Figure 1.C.2 illustrates a fundamental difference between our proposed WB malware and traditional PLC malware - WB malware decouples where the malware resides and where it executes. WB malware resides in PLC memory but executes in web browsers (e.g., Microsoft Edge on an EWS, Chromium-based Microbrowsers in local HMIs, etc) located in levels 2 and 3, physically detached from the PLC. These level 3 browsers are also connected to the business network to enable public internet access [11]. As a result from this architecture, WB malware can utilize a web-based Command & Control (C2) connection, where all C2 communication, initiated from level 3 browsers, traverses the business network in level 4/5 and escapes to the public internet (see Section 4.4).

Table 2: Viability of Access Levels PLC malware

	Web-Based	Control Logic	Firmware
Physical Access	✓	✓	✓
Network Access	✓	✓	✓
Web Access	✓		

✓= Viable Access Level for malware Infection

Advantages of WB PLC Malware. In addition to the extra infection scenario and C2 capability mentioned above, WB PLC malware has several other advantages over traditional PLC malware, as discussed in this subsection.

1) One benefit to developing PLC malware using browser-compatible JavaScript (instead of vendor-specific control logic or hardware-specific binary instructions) is that our proposed malware is largely architecture-agnostic and cross platform. WB malware that has been developed for a specific PLC model and ICS plant configuration can be easily modified to attack any other PLC in any other ICS. From our experiments, we estimate that only 5%-10% of WB malware code needs to be altered to change attack targets. Much of WB malware functionality such as screenshot exfiltration, virtual HMI interactions, and C2 channel communications do not need any alteration whatsoever (resulting in cheaper and faster malware development for adversaries). On the other hand, the payload development for control logic and firmware malware is specific to a certain PLC model, hardware architecture, and ICS configuration (I/O pin layout, actuator settings, etc.). Thus, traditional PLC malware requires significant time, effort, and domain knowledge to expand to other scenarios.

2) In Section 4.3, we explore how WB malware can be extremely resilient compared to traditional strategies. Using a combination of web technologies such as service workers and browser cache, we developed a WB PLC malware sample, called *Iron Spider*, that is capable of surviving PLC factory resets and even physical hardware replacement. This persistence level is much higher than CL malware, which is erased during control logic updates, and FW malware, which is erased during firmware updates.

3) In Section 4.4, we describe various methods for WB PLC malware to perform malicious activities against the ICS environment. These actions are more impactful than CL malware because they include altering administrative configurations such as user passwords and on-board firewalls, which is not possible using a control logic program. They are also more impactful than FW malware because they allow for real-time data exfiltration through level 3 browsers, which is not possible using firmware running in a segregated network.

4) Section 5.3 discusses why existing prevention, detection, and removal strategies are largely ineffective against WB PLC malware. Generally speaking, most countermeasures today focus on protecting the control logic and firmware portions of PLC computation, which are unmodified during our attack. Furthermore, because our proposed technique executes in web browsers, physically detached from the PLC even physical side channel anomaly detection strategies fail to detect WB PLC malware.

Considering the advanced capabilities of WB PLC malware, we claim that this emerging attack surface provides the best environment to stealthily attack ICSs. Table 3 summarizes the main differences between WB malware vs. previously studied PLC malware categories and shows that WB PLC malware is indeed more *flexible*, *resilient*, and *impactful*.

Table 3: WB vs. CL vs. FW PLC malware.

	Web-Based	Control Logic	Firmware
Infectability (F)	High	Medium	Low
Platform Independence (F)	High	Medium	Low
Persistence (R)	High	Low	Medium
Malicious Activities (I)	High	Low	Medium
Prevention Evasion (R)	High	Medium	Low
Detection Evasion (R)	High	Medium	Low

(F) = Flexibility; (R) = Resilience; (I) = Impact;

4 WB PLC MALWARE STAGES

This section introduces the stages of our proposed WB PLC malware using a vendor-agnostic framework as well as an example implementation on a widely-used PLC model in a real-world Stuxnet-style attack.

Vendor-Agnostic Framework. We developed a general-purpose framework for building and analyzing WB PLC malware. This framework explains the malware lifecycle using four distinct stages, as shown in Figure 3: *Initial Infection*, *Persistence*, *Malicious Activities*, and *Cover Tracks*. This framework explores each stage using widely applicable strategies that can be used against most modern PLC models. At a high level, the framework presents an overview of how malicious front-end code can subvert the integrity of ICS environments by methodically compromising PLCs' web properties. This framework can be used as a benchmark in future studies across any PLC vendor and model.

Example Implementation. Additionally, we implemented each step of this framework using an example malicious program, which we call *Iron Spider*. This program was designed to illustrate the effectiveness of the WB malware by performing a Stuxnet-style attack on a popular PLC model (WAGO 750) in a real-world ICS testbed. The testbed's main objective is to precisely spin a three-phase 220VAC industrial motor, representative of the ones used to power gas centrifuges during the uranium enrichment process.

We used this testbed to demonstrate the core functionality of *Iron Spider*, however modern ICSs of any size and complexity are equally as susceptible to this emerging threat. A detailed description of the testbed equipment and configurations can be found Section 5.

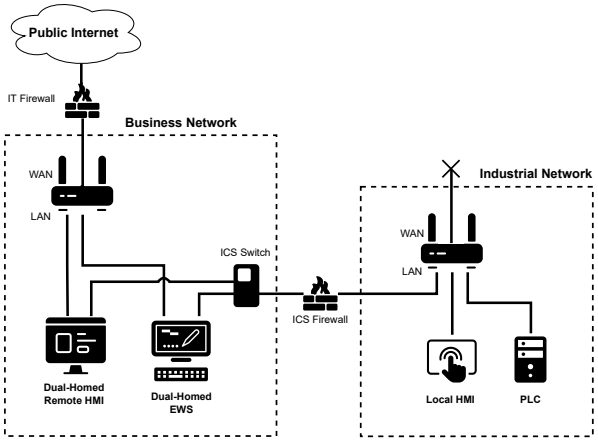

Figure 2: ICS Network Topology

4.1 Threat Model & Assumptions

In this attack, we assume that ICS operators use EWSs that have simultaneous access to both the business network and the industrial network and that both networks are secured using tightly configured firewalls. Figure 2 gives a detailed view of the network topology, which is a typical implementation of the PERA model in critical infrastructure environments according to surveys conducted by UK's Centre for the Protection of National Infrastructure [11]. We also assume that the EWSs are secured using standard IT best-practices (e.g., up-to-date operating system and browser, sufficiently strong system password, continuous anti-virus scans, etc.) and that the PLC is using the most secure settings possible (e.g., password-protected services, encrypted protocols, up-to-date firmware, etc.). Lastly, we assume that the adversary has knowledge of the PLC model being used in the ICS, however does not know its exact location in the network nor any details about other ICS/networking devices (data historians, RTUs, SCADA, routers, switches, etc). Note that while our testbed accomplishes the multi-network EWSs via dual-homed Ethernet adapters, this same attack can be accomplished in *any* networking configuration as long as the EWS can simultaneously access both the public web and the private PLCs, which is a typical case for ICS networks in-practice [11].

4.2 Initial Infection

The initial infection stage of our proposed WB Malware occurs when the attacker successfully plants malicious JavaScript code in a context where it will be executed in the same web origin as the PLC's admin portal (oftentimes referred to as the "system website"). Once deployed, our malware will have the same authority as the PLC's system website, and thus, identical functionality.

4.2.1 Initial Infection - Framework. Injecting code into the system website can be accomplished via several different methods including malicious User-defined Web Pages (UWPs), hijacked PLC GUI files, and ICS Cross Channel Scripting (XCS), as explored in this section. Each injection mechanism has its own strengths and weaknesses with varying degrees of practicality. The numerous injection mechanisms, spanning various technologies, is one of the reasons why we claim this malware is so flexible.

Malicious User-defined Web Page (UWP). PLC vendors such as Siemens, Allen Bradley, and Mitsubishi allow customers to write their own HTML code to augment the web application hosted by the PLC's embedded web server [3, 4, 51]. These custom HTML files are referred to as "User-defined Web Pages" (UWPs) and are leveraged to create specialized HMI dashboards. These UWPs are considered a distinct web property from the system website and have a limited set of advertised capabilities. UWPs typically have read-only access to PLC inputs/outputs, and depending on the vendor, may also have limited write-access to a subset of control logic variables.

The supposed restrictions imposed on UWPs would make most users assume that the impact of a malicious UWP is quite limited, however due to an unintended consequence of Same-Origin-Policy (SOP), the permission boundary of UWPs is actually defined by the *user* viewing the UWP and not an intrinsic property of the UWP itself, despite official vendor documentation stating otherwise [8]. This (seemingly misunderstood) relationship between the system website and UWPs allows a malicious UWP, when viewed by an administrator, to take full administrative control over the PLC. Thus, a malicious UWP is a viable injection mechanism to plant WB PLC malware. To help mitigate the impact of malicious UWPs, PLC vendors should consider *sandboxing* untrusted front-end code to an isolated origin (e.g., Facebook's *fbsbx.com* [19]). A detailed explanation of how this potential mitigation strategy can be adapted to the ICS domain is presented in Section 5.3.

UWPs can be downloaded to PLCs via proprietary ICS protocols (e.g., CIP PCCC for Allen Bradley [43] and ISO-TSAP for Siemens [50]) or via non-ICS download methods (e.g., FTP for GE [23] or SD card for Mitsubishi [4]). Furthermore, some vendors allow a full project image, including any UWPs, to be downloaded to the PLC over HTTP(s) via a "Restore from Backup" web API exposed by the system website. The flexibility of download methods gives an attacker multiple paths to planting a malicious UWP.

In lieu of personally downloading the malicious UWP, a bad actor can also trick an authorized user into installing a trojan UWP, as many UWPs are actually authored and sold by third-parties (e.g., Elmi Elettromeccanica [18]). The lack of web/front-end subject matter expertise by ICS operators combined with the misconception about the impact of malicious UWPs makes this injection path particularly feasible. This point is a compelling argument for enforcing domain sandboxing because without it, a UWP is equally as enticing of a target for attackers as the system website is. Furthermore, PLC vendors cannot guarantee the security of UWPs (as they are not authoring them), so the only way to mitigate their compromise is to isolate it from the administrative portion of the web application.

Hijacked PLC GUI Files. Instead of manually writing HTML code, WAGO and Schneider customers can choose to use software that generates web-based graphical user-interface (GUI) from a high-level visual description [46, 63]. The most common example of such software is the *WebVisu* application licensed from *CODESYS* [12]. This software allows an operator to drag-and-drop

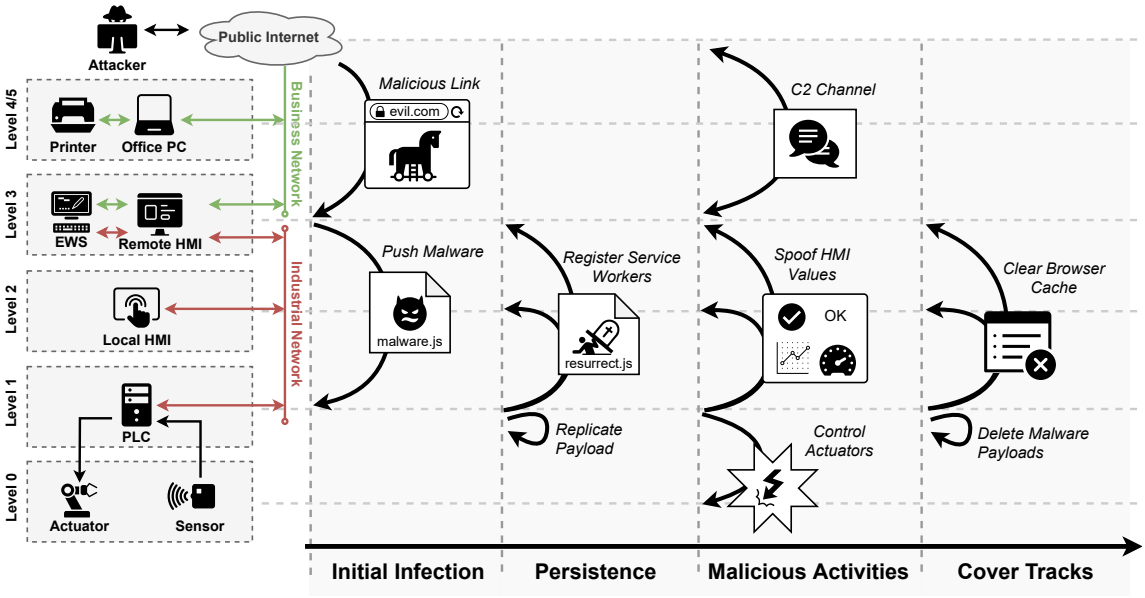

Figure 3: Lifecycle of WB PLC Malware

GUI elements to build an interface that gets transpiled into front-end files (HTML, JavaScript, CSS). This software helps ICS operators build rich, although less customized, HMI dashboards using pre-build elements, without needing any web subject matter expertise. If these files are hosted on the same embedded web server as the system website (without any sandboxing considerations), they may be a feasible infection mechanism for WB malware. In most cases, simply modifying the transpiled files in-transit during the download process or overwriting them in the filesystem after download is all that is needed to compromise the device. We verified this technique by SSH'ing into a WAGO 750 PLC in our lab and overwriting the transpiled front-end files.

ICS Cross-Channel Scripting. In addition to the two legitimate channels for pushing front-end code to the PLC discussed above, an attacker may also be able to exploit a Cross Channel Scripting (XCS) vulnerability to inject WB malware. XCS is an obscure variant of Cross-Site Scripting (XSS) where the malicious payload is transferred to the web server via a non-web protocol such as SNMP or FTP [36]. We discovered that this vulnerability classification is particularity common in the ICS domain because real-time constraints force industrial equipment to utilize low-latency proprietary protocols. While investigating this project, we observed that these protocols provide an effective method for sending malicious JavaScript payloads to the embedded web servers inside PLCs. We believe that XCS is an understudied vulnerability in the ICS domain as the analysis by our team revealed multiple zero-day vulnerabilities across several different vendors (e.g., CVE-2022-46670).

Discovering these injection bugs required manual effort with custom-written clients because traditional IT-oriented scanning tools (e.g., Burp Suite [42]) are unable to inspect the industrial protocols that PLCs regularly utilize to accept user-input (e.g., CIP, Modbus, Profibus, EIP, etc). Our research shows that PLCs are uniquely difficult to protect from JavaScript injection because their web servers often render user-input that was ingested from a plethora of specialized non-web protocols. Configuring web vulnerability scanners to interrogate ICS protocols may make for interesting future work.

4.2.2 Initial Infection - Example Implementation. After approximately one week of testing, our team identified four zero-day vulnerabilities in WAGO 750's latest firmware. These issues were disclosed to the vendor and fixed as CVE-2022-45137, CVE-2022-45138, CVE-2022-45139, and CVE-2022-45140. The methodology and testing procedures used to identify these issues are outside the scope of this paper, however we believe that similar vulnerabilities can be found in most PLC admin portal web applications. Next, we created a malicious website that, when viewed by anybody within levels 1-3 of PERA, exploited these vulnerabilities to automatically plant *Iron Spider* in the homepage of the WAGO system website. Our malicious website also used basic JavaScript resonance methods (e.g., websocket Favicon sweeping internal IP ranges [34]) to automatically locate the WAGO PLC within the private industrial network. Pseudocode for the malicious website is included below.

```
window.onload = function() {
    sweepForLocalServers().then((ips)=>{
        filterToWagoPLCs(ip).then((wagoIP)=>{
            exploitWebBugs(wagoIP).then(()=>{
                location.assign('//' + wagoIP);
            });
        });
    });
};
```

Listing 1: JavaScript code to locate and attack the PLC

We emailed a link to this website to the EWS in our testbed and manually opened it in the default web browser. Alternatively, an attacker looking to indiscriminately launch the attack at-scale can perform a *watering hole attack* [33] by simply purchasing an ad

banner on a popular PLC help forum. Note that the attacker did not need any prior knowledge of the EWS hardware, operating system, or web browser. After 3.7 seconds of viewing the webpage, *Iron Spider* was successfully downloaded to the target PLC without any user notification or firewall intervention. We emphasize that this attack did not actually *compromise* the EWS, but rather simply used it as a pivot point to gain network access to the PLC. None of the typical EWS security measures such as anti-virus scans and patch management were able to prevent this attack.

4.3 Persistence

The next stage of our framework is *persistence*. In this stage, the malware will employ several techniques to hide its presence and become resilient to typical removal methods.

4.3.1 Persistence - Framework. In our experiments, we found that the exact steps needed to accomplish persistence will vary depending on the model of the infected PLC, however the following sections will provide insight into the most common strategies.

Resurrection Code in HMIs and EWSs. A shockingly effective strategy to achieve persistence in the ICS domain is to leverage a web service worker to cache "resurrection code" in multiple web browsers throughout the ICS network (levels 2 and 3 in the PERA model). Service workers are a relatively new addition to the HTML5 specification that lets scripts run in the background, detached from any single web page. This powerful feature of web browsers is used to build rich offline experiences that include functionality such as push notifications and background sync [22]. We propose that this functionality be repurposed in the ICS domain to secure a foothold in the segregated industrial network by caching secondary malware payloads throughout the ICS environment. These secondary payloads will execute directly from EWS/HMI browser cache for up to 24 hours [60] after the primary malware payload has been completely removed from the PLC device. These service workers can periodically check for such removal, and when detected, use methods from the previous section to re-infect the device, as shown in Figure 4.

In addition to re-infecting a factory-reset PLC, the proposed strategy also allows the malware to infect any replacement PLC, thus giving the malware the ability to survive even after the PLC hardware has been completely rebuilt. This advanced "resurrection" technique helps the WB malware withstand even the most stringent eradication steps set by the Cybersecurity and Infrastructure Security Agency's (CISA) "Cybersecurity Incident & Vulnerability Response Playbooks" (i.e., reimage PLC to "gold" source and rebuild PLC hardware) [15]. Note that this malicious utilization of service workers is unique to the ICS domain because securing a web-based foothold in a segregated private network is not needed to communicate with a web server in the IT domain.

Self-Replication via Downgraded PLC Firmware. As discussed in Section 4.2, there are numerous infection mechanisms for WB malware. This is especially true if the malware uses the system website APIs to downgrade the firmware version to reintroduce known security issues. The malware may utilize these different infection mechanisms to provide redundancy (i.e., store copies of itself in different sections of the PLC memory), which enables the malware to survive intentional, or accidental, actions

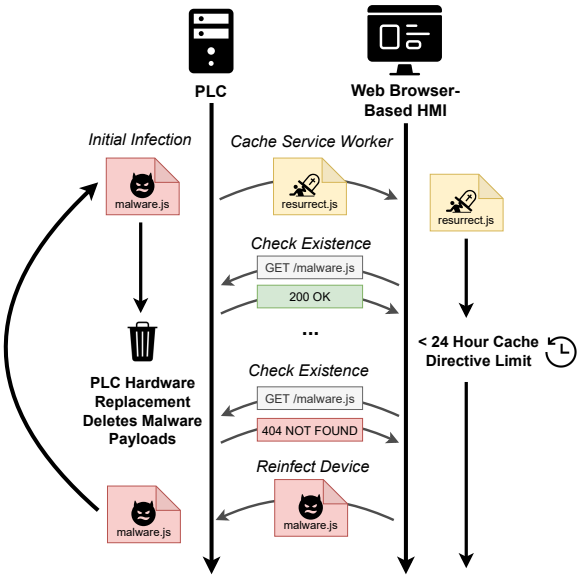

Figure 4: Service Worker used for WB malware re-infection

by the PLC operator that may delete the payload (e.g., updating configuration settings or power-cycling the device). We experimentally verified with Siemens S7-1200 that using web APIs to downgrade the firmware version then exploiting known file upload vulnerabilities was an effective method of self-replication in that PLC model. Note that the technique discussed here is unique to the ICS domain, as front-end code in the IT domain does not typically have the ability to control versioning of server-side code.

4.3.2 Persistence - Example Implementation. Recall that *Iron Spider* was installed on the homepage of the PLC's system website in the previous stage. In this stage, *Iron Spider* employed various strategies to ensure continuous execution in the ICS environment. Firstly, *Iron Spider*, utilized the same zero-day bugs from the Initial Infection step to overwrite the transpiled PLC GUI files, thus spawning new execution processes in both the local and remote HMIs. Next, *Iron Spider* registered service workers in all three of the browsers rendering its payload (Microsoft Edge on EWS, Chromium on remote HMI, WAGO microbrowser on local HMI). These service workers became cached in the browsers and periodically checked for the existence of the main malware payload. If/when the main payload was found to be missing, the service worker used the same methods from the first step to re-infect the device. We emphasize that this strategy enables *Iron Spider* to survive PLC hardware replacement.

4.4 Malicious Activities

The impact of the WB malware can be measured by its ability to sabotage real-world machinery, abuse PLC admin settings, and exfiltrate data. Recall from Section 3 that traditional PLC malware is trapped within segregated industrial network without an internet connection and that CL malware runs in a user-code sandbox with limited functionality. For these reasons, we claim that WB malware is capable of performing more impactful malicious activities than

prior work. Table 4 compares the capabilities of each PLC malware category.

Table 4: Malicious Capabilities per Malware Category

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

✓= Possible Malicious Activity

4.4.1 Malicious Activities - Framework. The malicious capabilities of the malware are directly mapped to features of the system website. All actions capable of being performed by a human using the system website normally can be accomplished programmatically by WB malware (e.g., virtually *"click"* buttons, virtually *"type"* into forms, and utilize all legitimate HTTP APIs). Therefore the impact of WB malware will depend on which PLC model has been infected. The following sections contain the general approach that can be applied to most models.

Sabotage Machinery for Physical Damage. A key component of any PLC malware is its ability to influence real-world physical events. Our proposed WB malware is capable of performing such control by utilizing the legitimate system website APIs to sabotage the industrial processes. Because WB malware executes on the same web origin as the PLC's system website, it can leverage the ambient browser credentials (e.g., cookies) needed to interact with authenticated web APIs. This can be done directly via JavaScript-initiated network requests (e.g., *fetch()* or *XMLHttpRequest*) or indirectly via JavaScript-initiated simulated user input (e.g., virtually "clicking" buttons in the UI). The specific steps needed to sabotage the real-world machinery will depend on the functionality provided by the vendor of the infected PLC. Some PLCs, such as the Allen Bradley MicroLogix 1400, expose web APIs to directly modify I/O values by overwriting the data stored in CPU memory addresses via the "*Editable Data Table Memory Map*" (even when the data itself is not tied to an HMI-controlled variable [3]). Other PLCs, such as the Schneider TM241, expose web APIs to overwrite control logic variables with arbitrary data [46]. Furthermore, some PLCs, such as Siemens S7-1200, even expose APIs to have the entire PLC project overwritten via the *"Restore from Backup"* functionality [55]. This backup can contain new set points, user configuration, and safety settings. A bad actor can abuse these powerful web APIs to maliciously control actuators and cause catastrophic damage to the underlying physical processes. Note that modifying control logic variable values via web-based APIs does not actually change the compiled control logic binary and will therefore not trigger any attestation systems such as *PLCDefender* [45], further illustrating how this attack is materially different than CL malware.

Due to the intentional human-readability of web-based HMIs, little-to-no prerequisite information regarding the underlying physical domain (level 0) is needed to launch a successful sabotage attack. An adversary can deduce unsafe states by visually inspecting screenshots of the HMI UI (exfiltrated using the techniques discussed later in this section) and modifying the controls accordingly (e.g. virtually "turn the knob" to change motor speed set points). This type of casual control is not possible using traditional (CL or FW) PLC malware, which requires intimate knowledge of I/O pin layout configuration and downstream actuator settings. Thus, physical sabotage via WB malware requires significantly less reverse engineering effort and prerequisite intelligence compared to existing strategies.

In addition to compromising the PLC actuators, WB malware can also sabotage industrial processes by spoofing values displayed in the system website and web-based HMIs. This can be accomplished by simply modifying the DOM of the displayed webpage using the standard JavaScript interface (e.g., *document.body.innerHTML*) or by overlaying fabricated displays (e.g., adding a screenshot to a top layer full-page *img* tag). For example, stealthy WB malware may record sensor values in browser storage (e.g. local storage [66]) and display them later during the actuator compromise to hide the attack.

Abuse Admin Settings for Further Compromise. Another malicious action that the WB malware can perform is to modify the administrative PLC configuring via the web-based APIs exposed by the system website. These APIs allow an operator to control admin settings on the device through a feature called "*Web-Based-Management (WBM)*" [54] [62]. An adversary can abuse these APIs to aid in future attacks or enable further compromise of the device. The specific settings available for modification depend on the PLC vendor and firmware version, however a typical attack may include editing the on-device firewall, creating new users, and enabling/disabling certain network services. We emphasize that this type of control is not possible with CL malware due to the user-code sandbox and is extraordinarily difficult with FW malware due to the tedious nature of binary instrumentation in a real-time embedded device (and is sometimes not possible at all depending on the chipset isolation in the motherboard [21]). With WB malware, this control is easily accomplished by simply calling the legitimate HTTP APIs with JavaScript. Note that our malicious JavaScript code leverages the same authentication mechanism as the user rendering its payload, which in the case of the EWS, will likely be the cookies belonging to the PLC system administrator because the primary purpose of the EWS is to perform administrative device configuration [14]. Table 5 lists common WBM admin settings and example consequences of their malicious misuse.

Data Exfiltration for Industrial Espionage. As mentioned in Section 2, the unique execution characteristics of our proposed malware allows it to utilize a public Internet connection even when the PLC itself is located in an isolated private network. Web browsers in which the malware executes (e.g., Microsoft Edge on an EWS and/or Chromium-based Microbrowsers in remote HMIs) are typically simultaneously connected to both the target PLC network and other less-critical networks [11]. For example, EWSs are usually connected to both the level 2 network (to communicate with the PLC) and the level 3 network (to perform online tasks like sending emails and viewing forum websites to troubleshoot devices, e.g., http://support.industry.siemens.com [53]). This simultaneous connectivity can be accomplished using a variety of networking setups common in large ICSs such as dual NICs, VLAN tagging, and firewall-managed enclaves. The perimeter of these networks are often secured using domain-specific firewalls that attempt to block abnormal traffic [11]. Our WB malware can bypass this scrutiny by

only utilizing protocols allowed in level 3 traffic, such as DNS or HTTPS, when communicating to the C2 server and only using the legitimate PLC APIs when communicating to the PLC.

Performing this exfiltration using WB malware does not require any intimate familiarity with the target PLC or the underlying physical process. Generic, but powerful, web-based exfiltration strategies such as canvas screenshots, event-listener keylogging, and full DOM dumps can be applied to virtually all PLC models across every vendor. These techniques allow WB malware to intercept sensitive ICS information such as physical process characteristics, plaintext usernames and passwords, and plant configuration details. This stolen data can be covertly exfiltrated using front-end network requests such as JavaScript *fetch()* and URI parameters to an HTML *img* tag *src*. Browser-based exfiltration strategies like this are notoriously difficult to detect or prevent using firewalls [10] because benign web applications often communicate with a variety of third party servers via encrypted protocols [44] (e.g., HTTPS and WSS), which commonly causes nefarious browser-based connections to go unnoticed, especially if the C2 infrastructure is built on top of a reputable third party web service such as Google Analytics or Facebook pixels [38]. Note that the C2 stream will be completely unaffected by any *on-device PLC firewalls* because the PLC-to-browser communication is utilizing the legitimate APIs provided by the PLC vendor. We emphasize that neither CL nor FW PLC malware can perform real-time data exfiltration in-practice because they typically execute in segregated industrial networks and cannot utilize a public internet connection.

Table 5: Consequences of Misusing WBM Admin APIs

Admin Setting	Example Misuse
On-Device Firewall	Modify EWS MAC Address whitelist to allow rogue connections to arbitrary hosts.
User Management	Create new "backdoor" user to ensure continuous control.
Network Configuration	Modify IP address to circumvent downstream firewall routing rules.
Network Services	Disable SNMP to prevent network operators from noticing IP reconfiguration.
Image Backup	Create and exfiltrate a full project backup, including current CL programs, device metadata, and historical logfiles to aid in espionage.
Security Settings	Disable IPsec, OpenVPN, and TLS to let other devices to MiTM PLC traffic.

4.4.2 Malicious Activities - Implementation Example. The goal of *Iron Spider* was to perform a Stuxnet-style attack using web technologies that circumvent modern defenses. To prepare for the attack, a real-time websocket C2 channel was established by all three execution environments. While only the EWS and remote HMI had direct internet access, the local HMI process was still able to achieve C2 communications by using a covert channel within the PLC device as a proxy. During the attack, *Iron Spider* modified the web-based HMIs' DOMs to display falsified sensor values (recorded the previous day and saved in browser local storage). Next, the malware virtually interacted with the HMI's UI to covertly change the set-point for the motor's speed control. Our testbed used an emergency light to indicate that a critical failure occurred (i.e., the attack was successful and it is too late to intervene). Approximately 7.4 seconds after the attack began, the emergency system tripped indicating that the centrifuge was damaged. The HMIs displayed fake readings throughout the entire attack.

4.5 Cover Tracks

The final stage of the framework is to remove any traces of the infection. This stage is aimed to impede incident response and forensics postmortem efforts. We emphasize that this self-contained removal strategy is not possible using CL malware due to the user-code sandbox in which it executes.

4.5.1 Cover Tracks - Framework. The following section contains the general approach to removing the payload and resetting the device.

Delete PLC Malware Payload. Once the malicious activities have been accomplished and the malware is no longer needed, it should attempt to remove the payload from PLC storage. The exact removal techniques will depend on how the payload was initially implanted onto the device (see Section 4.2), however in general, the process will involve repeating stage 1 using a blank, or otherwise benign, payload to overwrite the malware file. Additionally, the malware will need to invalidate any resurrection code by *unregistering* the service workers using the exposed HTML5 browser APIs.

Restore from PLC Backup Image. Even after the payload has been removed from PLC storage, traces of its existence may still reside in access logs and/or obscure memory caches. To finish the removal process, WB PLC malware may use the legitimate system website APIs to restore the PLC program from a prior backup image. Doing this will overwrite the set points, reset all configurations, and flush all caches. This drastic step gives the malware full control of the current PLC image, which is likely where forensics teams will begin their investigation. This type of absolute control over raw server-side memory states is unique to the ICS domain and allows WB PLC malware to self-destruct in a much more complete manner than malicious JavaScript in the IT domain.

4.5.2 Cover Tracks - Implementation Example. After *Iron Spider*'s successful attack, it began the process of cleansing the PLC of any traces of the infection. This was a non-trivial exercise because our zero-day vulnerabilities (CVE-2022-45137, CVE-2022-45138, CVE-2022-45139, and CVE-2022-45140) planted the malware payload in a section of the PLC's filesystem that was unaffected by factory resets. *Iron Spider*'s first step in covering its tracks was to re-use these vulnerabilities to overwrite both the system website homepage and compiled GUI files back to their original content. Then, the malware (while still executing on any open tabs) unregistered the service workers and flushed browser cache. Lastly, the malware refreshed all pages, thus killing all execution processes.

5 EVALUATION

As demonstrated in the previous section, *Iron Spider*, is capable of sabotaging industrial processes using a radically different approach than the ones used by existing PLC malware. Our malware can exfiltrate sensitive data, spoof HMI displays, maliciously control PLC

actuators, and self-destruct all without system-level compromise. Furthermore, *Iron Spider*, does not need to hijack any peripheral systems such as EWS or Engineering Software library dependencies. Recall that implementation details for *Iron Spider* were included in each subsection of Section 4. This section provides an analysis of the experimental results obtained from running *Iron Spider* in a real-world ICS testbed to perform a Stuxnet-style attack using web technologies that circumvent modern defenses. This section also discusses the efficacy of existing and proposed countermeasures.

5.1 Experimental Setup

The previous section explained how *Iron Spider* implemented each stage of our WB PLC malware framework in a real-world ICS testbed constructed to precisely spin a three-phase 220VAC industrial motor. This scenario was inspired by the real-world configuration for controlling uranium enrichment centrifuges during the Stuxnet attack [69]. This subsection outlines the key hardware components and networking details that comprised our testbed.

Networking Details. We developed a real-world ICS network environment by segregating the industrial network from the business network, as shown in Figure 2. The industrial network consisted of a *WAGO 750* PLC, a *WAGO eDisplay! 7300* local HMI, and a dedicated LAN port to connect to certain devices in the business network. Firewall rules only permitted legitimate PLC traffic to traverse the industrial network and the WAN port was sealed to prevent direct routable access to other networks. The business network consisted of a Raspberry-Pi remote HMI and a Microsoft Windows EWS. The business network was connected to the public Internet through a firewall that only allowed standard office traffic. The remote HMI and EWS were dual-homed to enable simultaneous access to both the business and industrial networks. We emphasize that while we chose to use dual-homing for its simplicity, *any* network configuration where EWS can simultaneously access both the public web and the private PLCs (a typical setup in practice [11]) would suffice for this attack.

PLC Configuration. The WAGO 750 PLC controlled a *Dayton 11W366* industrial motor with a 0-10v analog signal to a *Schneider ATV12* Variable Frequency Drive. The PLC also read the actual rotor speed using a *Compact Instruments Tachoprobe A2108* tachometer, which outputted a 0-6V analog signal. A web-based HMI was developed using WAGO's *WebVisu* integration that allowed operators to both view the tachometer readings and change the setpoint for the motor speed. This HMI was displayed in the *WAGO eDisplay! 7300* Microbrowser local HMI and in Chrome on the remote HMI.

Failure Indication. Lastly, an emergency light was configured to trip if the tachometer read values over a certain threshold. This system was designed to indicate that a critical failure has occurred, similar to a standard smoke detector.

5.2 Execution & Results

Iron Spider performed Stuxnet-style sabotage by orchestrating an end-to-end attack where the EWS, HMIs, and PLC all worked together to covertly set the motor speed setpoint to a dangerous level. This subsection summarizes the data flow during the attack and analyzes the outcome from the sabotage.

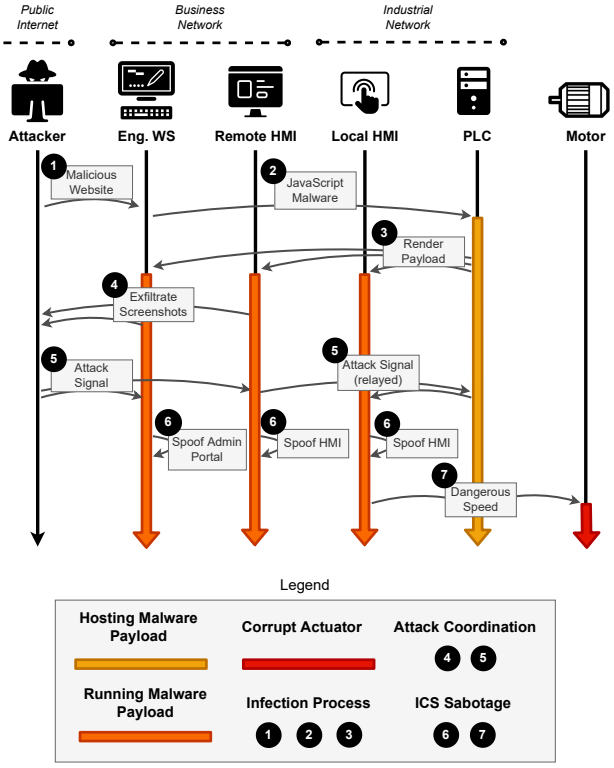

Figure 5: Overview of *Iron Spider*'s Data Flow

Execution. During this attack, *Iron Spider*, covertly interacted with several components of the ICS environment. The EWS was used as a pivot point to gain entry into the industrial network through JavaScript-initiated network requests by the malicious website. The PLC was used to host the malware payload, relay C2 messages to the segregated local HMI, and physically interact with the sensors/actuators. Both the HMIs and the EWS were used to execute the malware payload in their respective browsers. And of course, the motor was used to sabotage the industrial process. Figure 5 illustrates the high-level data flow that occurred during this attack. We emphasize that neither CL nor FW malware alone could perform data exfiltration in our testbed due to the realistic network segregation controls.

Results. Stuxnet sabotaged Iranian nuclear facilities by modifying the analog output signal to variable-frequency drives that controlled uranium enrichment centrifuges [69]. A direct result from this sabotage was the physical destruction of over 1,000 centrifuges and a 30% reduction in operational capacity at the facilities [6]. Our prototype malware, *Iron Spider*, was able to achieve a fundamentally similar attack (covertly altering the rotor speed of an industrial motor) using a radically different approach.

Stuxnet attacked PLCs via CL malware that it deployed via compromised EWS (these EWS were compromised using a Microsoft Windows worm and Trojan Step7 DLLs [29]). *Iron Spider*, however, used WB malware that it deployed using a malicious website without needing to compromise any peripheral systems. While both

Table 6: 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

attacks achieved the same outcome (sabotaged motor), *Iron Spider*'s approach has all of the advantages discussed in Section 3.

Figure 6 shows the true tachometer reading during the *Iron Spider* attack. Recall that a spoofed tachometer reading, with incorrect values, was visibly shown on the HMIs while the attack took place. As displayed in this figure, shortly after the attack began, the industrial motor from the testbed started spinning at a speed above the critical safety threshold[2].

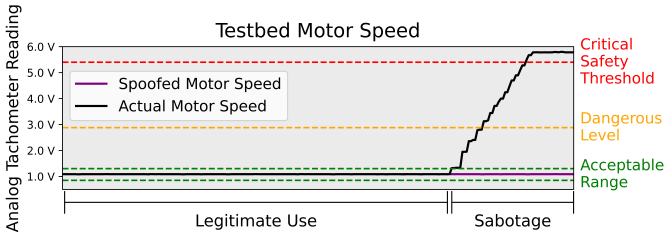

Figure 6: Tachometer Readings During the *Iron Spider* Attack

5.3 Countermeasures

Existing Countermeasures. The unique strategy employed by *Iron Spider* (i.e., circumventing the need for system-level compromise by targeting the web application hosted by the embedded web server) allowed it to bypass virtually all modern ICS protections, both ones actively used by industry and proposed countermeasures in recent academic papers. This includes defences such as ICS-configured firewalls, PLC intrusion prevention systems (e.g., *Reditus* [41]), PLC control logic attestation systems (e.g., *PLCDefender* [45]), and PLC control logic formal verification (e.g., *TSV* [37]). Table AI in the Appendix lists modern countermeasures and a brief explanation of their ineffectiveness. Generally speaking, most countermeasures today focus on protecting the control logic and firmware portions of PLC computation, which are unmodified during our attack.

Proposed Countermeasures. We are now faced with a bleak reality - there is no silver bullet for fully preventing WB PLC malware attacks. That being said, this emerging threat can be partly mitigated using a combination of protections implemented by web browser developers, PLC vendors, and third-party ICS products. These protections can serve as layers for a defence-in-depth strategy to reduce the likelihood and impact of an attack. Table 6 lists the proposed countermeasures, the protections provided by each defense, and the practicality of deploying them in real-world ICSs. Notable W3C-draft/proposed browser improvements include "*Private Network Access*" and the "*Content-Security-Policy (CSP) Confidentiality Directive*." Potential protections by PLC vendor include utilizing a read-only CDN used in conjunction with the *CSP src-script Directive* and *Subresource Integrity (SRI)* as well as domain-sandboxing for untrusted JavaScript code (i.e., isolating customer-authored UWPs and GUIs from the system website). Unfortunately, the domain-sandboxing techniques used in the IT domain are not directly usable in the ICS domain, as ICS environments typically use IP addresses instead of domain names to access servers. A slightly different approach to domain sandboxing that is more practical in industrial environments is to either use a different port of the same IP address or to utilize the CSP *sandbox* directive to virtually render untrusted content on an isolated opaque origin. Finally, a potential protection that does not rely on browser or PLC vendors could be a PLC-configured Web App Firewall (WAF) (i.e., a WAF configured to inspect non-web PLC protocols such as SNMP, Modbus, and CIP). We verified the efficacy of these protections using a series of simulations in Section Appendix I.1 of the Appendix.

6 CONCLUSION

Contrary to popular belief, firmware and control logic are not the only levels of PLCs computation. Modern PLCs also utilize an embedded web server that has become increasingly more powerful over the past 10 years. So powerful, in fact, that hijacking it using web-based malware proves to be more fruitful for attackers than hijacking its firmware or control logic. We believe that WB PLC malware is an emerging threat to ICS environments and will have devastating real-world consequences if not adequately defended against. We demonstrated with *Iron Spider* that a Stuxnet-style attack is not only possible despite modern defences, but is actually easier to perform today than it was back in 2005 when the actual malware was developed. Web applications are now deeply integrated into the modern ICS landscape, with embedded web servers talking directly to embedded web browsers, and this trend is only expected to increase over time. Appropriate precautions should be taken to ensure that WB PLC does not undermine all other ICS defences.

[2]As a safety precaution, we set the threshold of our testbed motor to a modest speed that did not actually cause any physical damage.

REFERENCES

[1] . [n.d.]. Five Myths of industrial control systems security. https://media.kaspersky.com/pdf/DataSheet_KESB_5Myths-ICSS_Eng_WEB.pdf

[2] . [n.d.]. Industrial Control Systems (ICS) Security Market by Solution (Firewall, Antimalware/Antivirus, IAM, Encryption, Whitelisting, Security Configuration Management, DDoS, and IDS/IPS), Service, Security Type, Vertical, and Region - Global Forecast to 2023. https://www.marketsandmarkets.com/Market-Reports/industrial-control-systems-security-ics-market-1273.html

[3] [n.d.]. MicroLogix 1400 Embedded Web Server. https://literature.rockwellautomation.com/idc/groups/literature/documents/um/1766-um002_-en-p.pdf

[4] [n.d.]. Web Server Function Application Guide. https://dl.mitsubishielectric.co.jp/dl/fa/document/catalog/plcf/l08643/l08643-a.pdf

[5] Arizton Advisory and Intelligence. [n.d.]. PLC Market - Global Outlook and Forecast 2020-2025. https://www.arizton.com/market-reports/plc-market-analysis

[6] David Albright, Paul Brannan, and Christina Walrond. 2010. *Did Stuxnet take out 1,000 centrifuges at the Natanz enrichment plant?* Institute for Science and International Security.

[7] Aplitude. [n.d.]. 2022 App vs. Website Trend Report. https://amplitude.com/2022-app-vs-website-report#key-takeaways

[8] Rockwell Automation. [n.d.]. MicroLogix 1400 Embedded Web Server. https://literature.rockwellautomation.com/idc/groups/literature/documents/um/1766-um002_-en-p.pdf

[9] Martín Barrère, Chris Hankin, Demetrios G Eliades, Nicolas Nicolau, and Thomas Parisini. 2019. Assessing cyber-physical security in industrial control systems. *arXiv preprint arXiv:1911.09404* (2019), 1–10.

[10] Kenton Born. 2010. Browser-based covert data exfiltration. *arXiv preprint arXiv:1004.4357* (2010).

[11] Centre for the Protection of National Infrastructure. [n.d.]. Firewall Deployment for SCADA and Process Control Systems. https://www.energy.gov/sites/prod/files/Good%20Practices%20Guide%20for%20Firewall%20Deployment.pdf

[12] Codesys. [n.d.]. CODESYS WEBVISU. https://www.codesys.com/products/codesys-visualization/webvisu.html

[13] Andrei Costin, Apostolis Zarras, and Aurélien Francillon. 2016. Automated dynamic firmware analysis at scale: a case study on embedded web interfaces. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. 437–448.

[14] Cybersecurity and Infrastructure Security Agency. [n.d.]. Control System Engineering Workstation. https://www.cisa.gov/uscert/ics/Control_System_Engineering_Workstation-Definition.html

[15] Cybersecurity and Infrastructure Security Agency. [n.d.]. Cybersecurity Incident & Vulnerability Response Playbooks. https://www.cisa.gov/sites/default/files/publications/Federal_Government_Cybersecurity_Incident_and_Vulnerability_Response_Playbooks_508C.pdf

[16] Defense Use Case. 2016. Analysis of the cyber attack on the Ukrainian power grid. (2016), 1–29.

[17] Alessandro Di Pinto, Younes Dragoni, and Andrea Carcano. 2018. TRITON: The first ICS cyber attack on safety instrument systems. In *Proc. Black Hat USA*. 1–26.

[18] Elmi Elettromeccanica. [n.d.]. PLC Siemens TIA Portal - Controller per turbine eoliche - Automazione e robotica windmill. http://www.elmielettromeccanica.it

[19] Facebook. [n.d.]. Out of scope & false positive XSS. https://www.facebook.com/whitehat/education/false-positives/

[20] Nicolas Falliere, Liam O Murchu, and Eric Chien. 2011. W32. Stuxnet dossier. *White paper, Symantec Corp., Security Response* 5, 6 (2011), 29.

[21] Luis Garcia and Saman A Zonouz. 2017. Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit.. In *Network and Distributed System Security (NDSS) Symp.* 1–15.

[22] Matt Gaunt. [n.d.]. Service Workers: an Introduction. https://developers.google.com/web/fundamentals/primers/service-workers

[23] GE. [n.d.]. PACSystems* RX7i & RX3i TCP/IP Ethernet Communications User Manual. https://www.manualslib.com/manual/1258748/Ge-Rx3i.html?page=22

[24] Geoffrey K Gill and Chris F Kemerer. 1991. Cyclomatic complexity density and software maintenance productivity. *IEEE transactions on software engineering* 17, 12 (1991), 1284–1288.

[25] Github Security Engineering. [n.d.]. GitHub's post-CSP journey. https://github.blog/2017-01-19-githubs-post-csp-journey/

[26] Google. [n.d.]. Content hosting for the modern web. https://security.googleblog.com/2012/08/content-hosting-for-modern-web.html

[27] Baptiste Gourdin, Chinmay Soman, Hristo Bojinov, and Elie Bursztein. 2011. Toward secure embedded web interfaces. In *20th USENIX Security Symposium (USENIX Security 11)*.

[28] Naman Govil, Anand Agrawal, and Nils Ole Tippenhauer. 2017. On ladder logic bombs in industrial control systems. In *Computer Security*. 110–126.

[29] ICS Alert (ICS-ALERT-14-281-01E). [n.d.]. Ongoing Sophisticated Malware Campaign Compromising ICS (Update E). https://us-cert.cisa.gov/ics/alerts/ICS-ALERT-14-281-01B

[30] Tal Keren. [n.d.]. THE RACE TO NATIVE CODE EXECUTION IN PLCS. https://claroty.com/2021/05/28/blog-research-race-to-native-code-execution-in-plcs/

[31] Rafiullah Khan, Peter Maynard, Kieran McLaughlin, David Laverty, and Sakir Sezer. 2016. Threat analysis of blackenergy malware for synchrophasor based real-time control and monitoring in smart grid. In *4th Int. Symp. ICS & SCADA Cyber Security Research*. 53–63.

[32] Eric D Knapp and Joel Thomas Langill. 2014. *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other industrial control systems*. Syngress.

[33] N Krithika. 2017. A study on wha (watering hole attack)–the most dangerous threat to the organisation. *Int. J. Innov. Sci. Eng. Res.(IJISER)* 4 (2017), 196–198.

[34] Jorge Lajara. [n.d.]. JS-Recon detailed Analyzing the internal network with a XSS. https://jlajara.gitlab.io/js-recon

[35] Ralph Langner. 2011. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy* 9, 3 (2011), 49–51.

[36] R Madhusudhan et al. 2018. Cross Channel Scripting (XCS) Attacks in Web Applications: Detection and Mitigation Approaches. In *2018 2nd Cyber Security in Networking Conference (CSNet)*. IEEE, 1–3.

[37] Stephen E McLaughlin, Saman A Zonouz, Devin J Pohly, and Patrick D McDaniel. 2014. A Trusted Safety Verifier for Process Controller Code.. In *NDSS*, Vol. 14.

[38] MITRE. [n.d.]. Exfiltration Over Web Service. https://attack.mitre.org/techniques/T1567/

[39] Mozilla Firefox. [n.d.]. Security/CSP/Confidentiality. https://wiki.mozilla.org/Security/CSP/Confidentiality

[40] Alberto S Nuñez-Varela, Héctor G Pérez-Gonzalez, Francisco E Martínez-Perez, and Carlos Soubervielle-Montalvo. 2017. Source code metrics: A systematic mapping study. *Journal of Systems and Software* 128 (2017), 164–197.

[41] Syed Ali Qasim, Jared M Smith, and Irfan Ahmed. 2020. Control logic forensics framework using built-in decompiler of engineering software in industrial control systems. *Forensic Science International: Digital Investigation* 33 (2020), 301013.

[42] Sagar Rahalkar. 2021. Extending Burp Suite. In *A Complete Guide to Burp Suite*. Springer, 131–145.

[43] Rockwell Automation. [n.d.]. Delivery of CIP Over RA Serial DF1 Links. https://www.rockwellautomation.com/content/dam/rockwell-automation/sites/downloads/pdf/CIPandPCCC_v1_1.pdf

[44] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. 2012. Detecting and Defending Against {Third-Party} Tracking on the Web. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. 155–168.

[45] Mohsen Salehi and Siavash Bayat-Sarmadi. 2021. PLCDefender: Improving Remote Attestation Techniques for PLCs Using Physical Model. *IEEE Internet of Things Journal* 8, 9 (2021), 7372–7379.

[46] Schneider. [n.d.]. How to configure Web Visualization in TM241 using Ecostruxture Machine Expert? https://www.se.com/in/en/faqs/FAQ000191672/

[47] Schweitzer Engineering Laboratories Inc. [n.d.]. Interface HMI Touchscreens to SEL Devices Using Modbus Protocols. https://cms-cdn.selinc.com/assets/Literature/Publications/Application%20Notes/AN2013-24-20130701.pdf?v=20211013-235342

[48] Schweitzer Engineering Laboratories Inc. [n.d.]. Web-based HMI: An emerging trend? https://www.automation.com/en-us/articles/2003-1/web-based-hmi-an-emerging-trend

[49] Shodan. [n.d.]. Shodan Search Engine. https://shodan.io/

[50] Siemens. [n.d.]. Basic Examples for Open User Communication: ISO-on-TCP. https://cache.industry.siemens.com/dl/files/710/109747710/att_923140/v6/109747710__IsoOnTcp_BaseComm_V1_en.pdf

[51] Siemens. [n.d.]. Creating Userdefined Web Pages on S7-1200 / S7-1500. https://cache.industry.siemens.com/dl/files/496/68011496/att_917318/v3/68011496_S7-1200_1500_Webserver_DOC_v22_en.pdf

[52] Siemens. [n.d.]. Firmware update for CPU 1214C, DC/DC/DC, 14DI/10DO/2AI. https://support.industry.siemens.com/cs/document/107539750/firmware-update-for-cpu-1214c-dc-dc-dc-14di-10do-2ai?dti=0&lc=en-US

[53] Siemens. [n.d.]. Industry Online Support. https://support.industry.siemens.com/cs/start?lc=en-US

[54] Siemens. [n.d.]. SCALANCE XM-400/XR-500 Web Based Management (WBM). https://cache.industry.siemens.com/dl/files/663/109798663/att_1070766/v1/PH_SCALANCE-XM-400-XR-500-WBM_76.pdf

[55] Siemens. [n.d.]. Simatic Web Server. https://cache.industry.siemens.com/dl/files/560/59193560/att_898124/v1/s71500_webserver_function_manual_en-US_en-US.pdf

[56] Spider Control. [n.d.]. (Micro-) Browser Solution. https://spidercontrol.net/spidercontrol-products/micro-browser-solution/?lang=en

[57] statista. [n.d.]. Global PLC market share as of 2017, by manufacturer. https://www.statista.com/statistics/897201/global-plc-market-share-by-manufacturer/

[58] Keith Stouffer, S Lightman, V Pillitteri, Marshall Abrams, and Adam Hahn. 2014. NIST special publication 800-82, revision 2: Guide to industrial control systems (ICS) security. *National Institute of Standards and Technology* (2014).

[59] Steven Van Acker, Daniel Hausknecht, and Andrei Sabelfeld. 2016. Data Exfiltration in the Face of CSP. In *Proc. of the 11th ACM on Asia Conf. on Computer and*

Communications Security. 853–864.
[60] W3C. [n.d.]. Service Workers. https://www.w3.org/TR/service-workers/
[61] W3C Community Group. [n.d.]. Private Network Access. https://wicg.github.io/private-network-access/
[62] WAGO. [n.d.]. PFC200 CONTROLLER. https://www.wago.com/us/pfc200
[63] WAGO. [n.d.]. WAGO WEBVISU APP. https://www.wago.com/us/software/webvisu
[64] WAGO. 2022. *WAGO PFC Firmware Releases*. https://github.com/WAGO/pfc-firmware/releases
[65] Lukas Weichselbaum, Michele Spagnuolo, Sebastian Lekies, and Artur Janc. 2016. CSP is dead, long live CSP! On the insecurity of whitelists and the future of content security policy. In *Proc. ACM SIGSAC Conf. Computer and Communications Security*. 1376–1387.
[66] whatwg.org. [n.d.]. HTML Living Standard Web storage. https://html.spec.whatwg.org/multipage/webstorage.html
[67] Timothy Williams. 1998. The Purdue enterprise reference architecture and methodology (PERA). *Handbook of life cycle engineering: concepts, models, and technologies* 289 (1998).
[68] Hyunguk Yoo, Sushma Kalle, Jared Smith, and Irfan Ahmed. 2019. Overshadow PLC to detect remote control-logic injection attacks. In *Int. Conf. Detection of Intrusions and Malware, and Vulnerability Assessment*. 109–132.
[69] Kim Zetter. [n.d.]. An Unprecedented Look at Stuxnet, the World's First Digital Weapon. https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/
[70] Mu Zhang, Chien-Ying Chen, Bin-Chou Kao, Yassine Qamsane, Yuru Shao, Yikai Lin, Elaine Shi, Sibin Mohan, Kira Barton, James Moyne, and Z. Morley Mao. 2019. Towards Automated Safety Vetting of PLC Code in Real-World Plants. In *IEEE Symp. Security and Privacy (S&P)*. 522–538.

Appendix I APPENDIX

Appendix I.1 Countermeasures Discussion

Browser Improvements. The first area of potential improvement is *limiting communication between the public Internet and private Intranet*. Providing this defence will greatly reduce the possibility of a "Web Access" infection (see Section 3) because it would restrict public websites' access to PLCs' embedded webservers using HTML and/or JavaScript network requests. In theory, this prevention should cause many PLC web bugs such as rXSS and CSRF to become mostly un-exploitable from malicious websites originating from the public web. Google Chrome engineers have already proposed this countermeasure in the form of the "*Private Network Access*" specification (previously known as *CORS-RFC1918*) [61], however this defence has not yet been implemented in any major browser. While a full adoption of this protection will definitely help in mitigating web-based infections, it is unfortunately not a perfect solution in all cases. Currently, the draft does not apply to local HTML files (even when delivered via a malicious USB drive). This draft also will not prevent an existing WB malware infection from spreading to other PLCs because existing WB malware will already have a private IP address, which is permitted to interact with other private IP addresses according to the spec.

Another area of potential improvement for browsers is *adding built-in exfiltration defences*. There are currently no browser mechanisms that prevent rogue JavaScript code from covertly connecting to, and exfiltrating sensitive data to, a third party remote server. This lack of protection is what enables our proposed malware to establish a C2 channel with the public internet. While some websites attempt to use Content-Security Policy (CSP) as a makeshift exfiltration defence [25], prior work has shown that this is an inadequate protection because methods such as DNS resolution, subresource prefetching, and navigation redirects easily defeat it [59]. In 2012, Firefox developers proposed a new CSP directive called "*Confidentiality*" [39] designed to fully eliminate exfiltration attacks, however it was shelved shorty after creation due to reprioritization. We hope that this paper will inspire browser developers to reevaluate that decision.

ICS Protections. In addition to improving the web browsers in which WB PLC malware executes, protecting the embedded web server itself is also a reasonable mitigation strategy. As discussed in Section 4.2, there are numerous techniques to infect a PLC with WB malware, including XSS, malicious UWP, and hijacked GUIs.

Defending against XSS attacks has been a long-standing cat-and-mouse game in the IT domain, with countermeasures constantly being optimistically developed and woefully defeated [65]. Unfortunately, no XSS defences have proven to be fully effective in all scenarios. The most successful approach is a defence-in-depth strategy where software development best practices such as contextually aware input filtering and output encoding are utilized in tandem with browser defences such as CSP and script attestation. Additionally, network-based defences such as Web App Firewalls (WAFs) can also be deployed as another layer of defense. Unfortunately, in the ICS domain WAFs will only provide partial coverage because XSS payloads can be transferred to PLCs using a plethora of non-web protocols such as SNMP, Modbus, and CIP, which ultimately get rendered in the system website. Configuring a WAF to inspect non-web protocols commonly used by PLC network services could make for interesting future work.

Lastly, all JavaScript code authored by the PLC customer (either in UWPs or GUI files) should be fully isolated from the front end properties authored by the PLC vendor. As mentioned in Section 4.2, the IT domain has a well-established method for performing origin isolation called "domain sandboxing." Using this method, untrusted front end files are hosted on a different web origin, usually with a different domain name, than the main website. For example, Facebook hosts customer-written code on *fbsbx.com* [19] and Google hosts customer-written code on *googleusercontent.com* [26]. This origin isolation prevents untrusted code from having access to the ambient browser credentials associated with the main origin, which are needed to access sensitive web APIs.

In ICS environments, where IP addresses are typically used instead of DNS to access network devices, a sandboxing solution may seem infeasible, however can still be accomplished using a slightly different approach. Instead of using a different domain name, PLC vendors can either host untrusted front end files on a different port of the same IP address or utilize the CSP *sandbox* directive to virtually render untrusted content on an isolated *opaque* origin.

Site-sandboxing is an effective strategy to prevent untrusted, customer-authored, code from hosting full-fledged WB PLC malware because it will be unable to perform many of the malicious actions outlined in Section 4.4.1 without access to the ambient browser credentials used by the system website. Until all PLC vendors natively add these protections, interesting future work could be to retroactively add the CSP *sandbox* header to untrusted code using an HTTP proxy via a bump in the wire.

Prevention Strategy	Effectiveness	Explanation
Network Isolation	✗	Web traffic can use JS-based network requests to pivot into private networks
EW Antivirus Software	✗	Engineering Workstation was not compromised
EW Patch Management	✗	Engineering Workstation was not compromised
IT Firewall	✗	Exfiltration and C2 only used typical web protocols (DNS, HTTPS, WSS)
ICS Firewall	✗	PLC communication only used legitimate PLC APIs
Network-Based ICS IPS (e.g., *Reditus* [41] or *Shade* [68])	✗	Control Logic was not altered
PLC Control Logic Attestation (e.g., *PLCDefender* [45])	✗	Control Logic was not altered
PLC Control Logic Formal Verification (e.g., *TSV* [37])	✗	Control Logic was not altered
PLC Control Logic Static/Dynamic Analysis (e.g., *VETPLC* [70])	✗	Control Logic was not altered
PLC Firmware Attestation	✗	Firmware was not altered
PLC Patch Management	✗	*Iron Spider* exploited zero-day webapp bugs in latest firmware
PLC Password Protection	✗	*Iron Spider* uses ambient browser credentials (e.g., cookies) of infected web application
Protocol-Layer Encryption	✗	*Iron Spider* does not rely on MiTM (of ICS or IT protocols)

✓= Effective; ✗= Not Effective

Table AI: Existing ICS Countermeasures and their Effectiveness at Preventing *Iron Spider*