An Experimental Analysis of Security Vulnerabilities in Industrial IoT Devices

XINGBIN JIANG, MICHELE LORA, and SUDIPTA CHATTOPADHYAY, Singapore University of Technology and Design

The revolutionary development of the Internet of Things has triggered a huge demand for Internet of Things devices. They are extensively applied to various fields of social activities, and concerning manufacturing, they are a key enabling concept for the Industry 4.0 ecosystem. Industrial Internet of Things (IIoT) devices share common vulnerabilities with standard IoT devices, which are increasingly exposed to the attackers. As such, connected industrial devices may become sources of cyber, as well as physical, threats for people and assets in industrial environments.

In this work, we examine the attack surfaces of a networked embedded system, composed of devices representative of those typically used in the IIoT field. We carry on an analysis of the current state of the security of IIoT technologies. The analysis guides the identification of a set of attack vectors for the examined networked embedded system. We set up the corresponding concrete attack scenarios to gain control of the system actuators and perform some hazardous operations. In particular, we propose a couple of variations of Mirai attack specifically tailored for attacking industrial environments. Finally, we discuss some possible

CCS Concepts: • **Security and privacy → Distributed systems security**; **Embedded systems security**; **Vulnerability management**; • **Hardware → Safety critical systems**;

Additional Key Words and Phrases: Industrial IoT security, mirai attack, stealthy attack

ACM Reference format:
Xingbin Jiang, Michele Lora, and Sudipta Chattopadhyay. 2020. An Experimental Analysis of Security Vulnerabilities in Industrial IoT Devices. *ACM Trans. Internet Technol.* 20, 2, Article 16 (May 2020), 24 pages.
https://doi.org/10.1145/3379542

1 INTRODUCTION

The constant need for the development of intelligent industry and smart factories forced the systematic exploitation of Internet of Things (IoT) devices into industrial applications. Every day, many tasks, such as production monitoring and control, or communication and data analysis for logistics, are more demanded to millions of devices spread around the world. Traditional logic

This project was partially supported by Keysight Technologies grant no. RTKS171003.
Authors' address: X. Jiang, M. Lora, and S. Chattopadhyay, Information Systems Technology and Design (ISTD), Singapore University of Technology and Design, 8 Somapah Rd, Singapore, 487372; emails: {xingbin_jang, michele_lora, sudipta_chattopadhyay}@sutd.edu.sg.

1533-5399/2020/05-ART16 $15.00
https://doi.org/10.1145/3379542

controllers are replaced by more advanced Cyber-Physical Systems (CPSs), usually connected to the Internet. This transformation is the key to enable what is now known as Industry 4.0 or smart manufacturing [11].

In the meantime, security issues have not yet received sufficient attention from manufacturers and users. Therefore, this vast amount of Industrial Internet of Things (IIoT) devices are becoming attractive targets for attackers to do malicious operations or obtain sensitive information. A query using Shodan or similar searching engines shows that there exists a large number of IIoT devices that are exposed on the Internet without being adequately secured [6].

Although consequences of cyber attacks may be generally dramatic, they may be catastrophic when affecting industrial applications. Effects may range from stolen classified information to causing physical damages to the production systems. For instance, an attacker gaining control of the system actuators may also cause physical harm to workers [7]. Any of these scenarios may be catastrophic. Stuxnet, for instance, has been advocated to be capable of causing a "new Chernobyl" [24] by compromising the industrial control systems of nuclear plants. Indeed, this would be an extreme scenario. Still, even the less critical attack may cause substantial economic losses due to production downtime [37] or leak of information related to intellectual properties. This is further worsened by recent manufacturing trends, where a leak of a connected device configuration may cause the leak of information about the product being produced by a reconfigurable manufacturing system [35].

Much work has been done to improve the security of the IoT devices [44]. However, industrial scenarios introduce further constraints that are not considered in classic IoT applications. IIoT applications are usually safety critical and time constrained (e.g., controlling robotic arms operating alongside a human). Hence, attack detection techniques must be performed in place, as delegating remote servers or cloud systems will introduce variability in the system timing, thus affecting real-time constraints [10]. At the same time, such solutions might expose new attack surfaces. Thus, solving security and privacy issues on IIoT systems requires consideration of a higher number of dimensions with respect to targeting the same problem for classic IoT applications. Some articles [3, 37, 47] have discussed the vulnerabilities that can potentially afflict an IIoT device, as well as potential protection methods [47]. However, none of them presented systematic research of the attack surfaces, attack vectors, and prevention methods for a typical IIoT device. In this article, we systematically analyze the feasibility of attacking an IIoT device by exploring concrete attack surfaces and attack vectors.

In particular, we came up with a variation of Mirai attack. Classic Mirai attack is usually used to perform Distributed Denial of Service (DDoS) attacks; however, the proposed variation aims at gaining control of the actuators to create dangerous situations, and stealing reserved information from the connected devices. Then, we extend our attack by making it stealthy to make it less identifiable by defense mechanisms. The experimental analysis is based on a networked embedded system, whose structure is depicted in Figure 1: it is composed of two micro-controllers connected to the network and controlling a physical actuator each. The devices we decided to use in this work are two i.MX 6SoloX Smart Application Blueprint for Rapid Engineering (SABRE) boards from NXP, being representative of a wide class of IIoT devices. The i.MX boards are then connected to a couple of Arduino devices for further controlling the servo motor actuators. During our vulnerability analysis, we will gain control of the actuators and make them perform some hazardous actions, such as moving to unsafe positions:

- We analyze the vulnerabilities in a typical IIoT device to characterize its attack surfaces and vectors. We use the device as the central core to build a scenario involving distributed actuation, which is typical of industrial control systems.

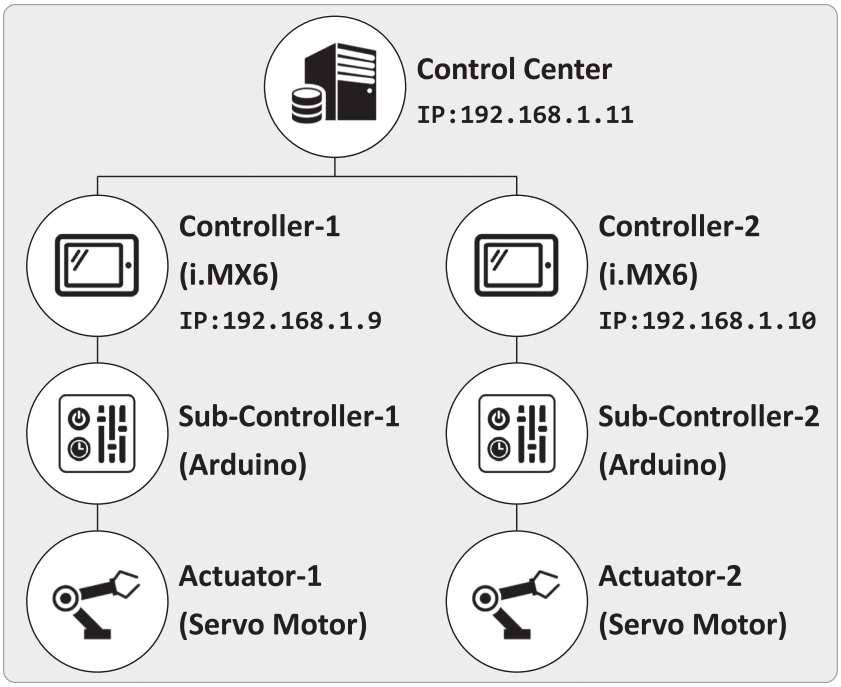


Fig. 1. Overview of the system target of our attacks. The system is composed of two i.MX6 micro-controllers connected to the LAN, with each of them controlling a servo motor actuator through an Arduino device.

- We analyze the security of the IIoT devices connected to the Internet today. We get inspired by previous analysis [2, 6, 43] to update the findings and relate them to the IIoT device used for the presented experimental analysis.
- We set up a set of attack scenarios to show how the attack on a common IIoT device may impact on the safety of an industrial actuator system.
- We propose a novel version of Mirai attack, aiming at impacting on the physical actuator system. Furthermore, we present a method to make Mirai attack stealthy and consequently difficult to be identified by general protection mechanisms, thus providing the attacker more time to inflict more physical damages to the actuation system.

The rest of the article is organized as follows. Section 2 introduces the necessary background, the system used as case study in the article, and the state of the art about the different techniques used in this work. Section 3 defines the system target of our analysis, identifying its attack surfaces, and introduces the attack scenarios. Furthermore, we present the updated analysis of security threats for IIoT devices we carried on, and we relate the finding of such analysis to the attack scenarios explored in our experimental analysis for the target system. Sections 4 through 6 detail the different attack scenarios and present their results. Finally, in Section 7, we discuss some possible countermeasures before drawing some conclusions in Section 8.

2 BACKGROUND

This section introduces the main concepts related to the introduction of IoT within industrial environments. Then, it details the architecture of the system used as a case study in this work. Finally, it presents the related work available in the literature.

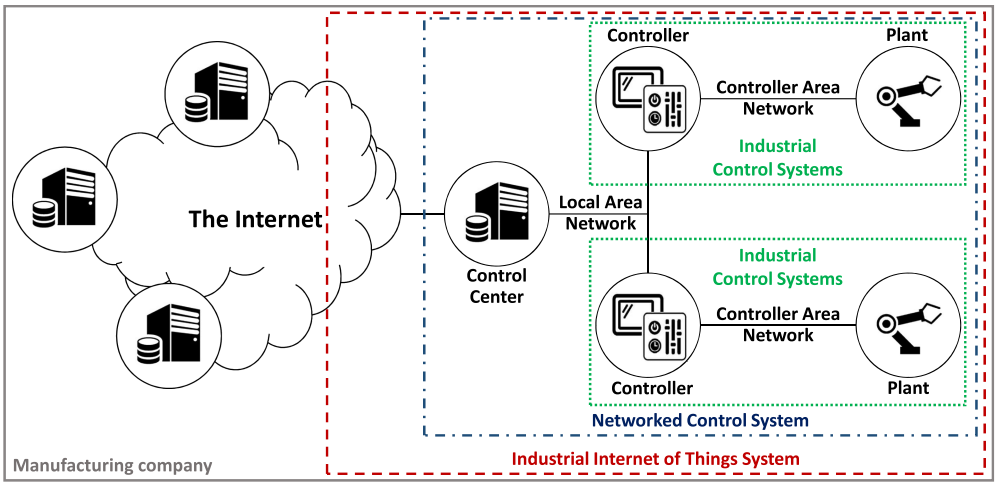


Fig. 2. Placement of an IIoT system within a manufacturing company system, highlighting the differences among the industrial control system, networked control system, and IIoT system.

2.1 IoT in Industry and Case Study

With the rising of modern manufacturing trends [9], many other terms emerged on the side of traditional *industrial control systems*. From a network perspective, *networked control systems* [13] first appeared to indicate production systems where the actuators composing the production plant are distributed and controlled through a communication network. Then, the growth of the IoT led to introducing such technology into industrial production systems, consequently introducing the concept of IIoT. At the current state of the practice, these concepts are co-existing within the same production systems. Figure 2 aims at clarifying the borders among the industrial control system, networked control system, and IIoT system. It shows a generic structure of a modern manufacturing company: its production plants may be distributed geographically and connected through the Internet. Each production plant is connected to its controller(s), usually through an ad hoc network, often creating a Controller Area Network (CAN), composing an industrial control system. Multiple industrial control systems connected through a Local Area Network (LAN) compose a networked control system, traditionally isolated from the external world. However, when such systems are connected to the external world through the Internet, they become IIoT systems.

To perform our experimental analysis, we built a low-cost system presenting the typical characteristics of an IIoT system. It is based on two i.MX6 micro-controllers produced by NXP. Each micro-controller integrates two CAN buses; it provides two Gigabit Ethernet interfaces and many other features, making it ideal to be deployed for industrial applications.[1] An Ethernet network connects the two micro-controllers to a control center. The control center is connected to the Internet, and it acts as gateway for the networked control system composed by the two i.MX6 micro-controllers.

On the controlled plant side, the two micro-controllers are connected to an Arduino device each. The Arduino devices act as an interface to a couple of servo motor actuators. These actuators must operate within a specific range of positions. Even though the physical plant is not built using hardware typically used in concrete industrial scenarios, it is meant to present the same

[1]NXP i.MX6 fact sheet: https://www.nxp.com/docs/en/fact-sheet/IMX6SRSFS.pdf.

vulnerabilities of a real-life industrial system. For instance, Arduino devices may be used to prototype a CPS, such as a production plant. However, it is not realistic to deploy an Arduino device in the final system. For this reason, our analysis does not consider the attack vectors of the Arduino devices. On the contrary, the i.MX6 micro-controller is meant to be deployed when implementing an Internet-enabled industrial control system and IoT systems in general. Furthermore, its attack surface shares several characteristics with the devices discovered in the Shodan-based analysis about the worldwide state of the practice presented in Section 3.2. As such, our case study is a low-cost prototype representative of many IIoT systems.

2.2 Related Work

Plenty of work has been done in the past few years to study the security and privacy issues of IoT applications and devices, both from the technical and legislative points of view [42, 46]. Several groups of researchers presented systematic reviews of threats in the IoT field by analyzing the security and privacy challenges, such as network security and identity management [44], or enumerating possible attack surfaces, such as devices memory, web interfaces, and device network services [1]. Zhou et al. [51] analyzed the effects of the most recent features of IoT devices and applications on security and privacy. It characterizes the emerging features of IoT systems, and it describes various scenarios, including industrial applications, and discusses the privacy and security issues that may arise. However, it is more a systematic review of the literature, and it does not provide any concrete analysis of the attack surfaces of such systems.

In the context of industrial control systems, Sadeghi et al. [37] gave an introduction to the security and privacy challenges in IIoT and an outlook on possible solutions toward a holistic security framework. They identified the attack surfaces of a typical smart factory: the potential attack vectors might be invasive hardware attacks, side-channel attacks, and reverse-engineering attacks. The software can be compromised by malicious code, such as Trojans, viruses, and runtime attacks. Communication protocols are subject to protocol attacks, including Denial of Service (DoS) attacks. The general potential solutions they discussed include introducing security architectures, verifying the integrity of a system's software configuration, and securing user interfaces or suitable communication interfaces. Sisinni et al. [41] analyzed the differences between the features of IIoT and consumer IoT. They specifically described a three-tier IIoT architecture and discussed the possible challenges and directions. In terms of security, they pointed out several security features to consider when designing secure IIoT infrastructure, including the use of appropriate encryption algorithms, reasonable identification, and authentication mechanisms.

A comprehensive set of statistics about the major threats for IIoT devices and networked industrial control systems was released in 2016 by Kaspersky Lab [4]. The study categorizes the different vulnerabilities identifiable into industrial control systems connected to the Internet, showing a wide range of different types of threats exploitable by potential attackers. A more recent analysis [2] analyzes instead the number of devices being used in industrial plants and accessible to the Internet. The analysis shows the types of devices and the services exposed by them. Thus, it identifies the possible threats due to the known vulnerabilities of devices and services. However, the analysis is restricted to a very reduced geographical area (i.e., Jordan) and a specific set of device producers. Furthermore, none of these works present a concrete, experimental scenario showing how devices can be compromised.

Quarta et al. [36] gave a comprehensive security analysis of an industrial robot controller, reporting possible attack vectors that could also share characteristics with generic IIoT devices, including insecure network surface and command injection, weak authentication, naive cryptography, and missing code signing. They showed different possible vulnerabilities. Then, they discussed how to

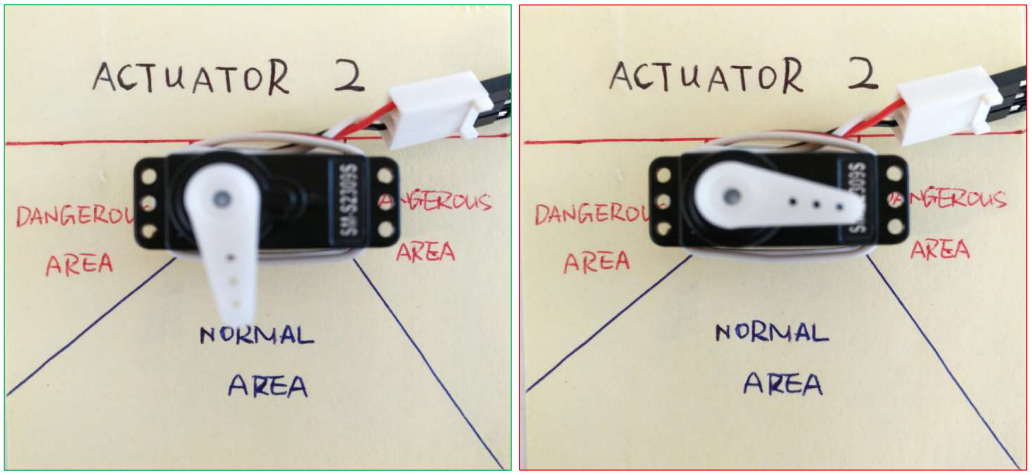
establish a security mechanism in an industrial robot system case study. Wurm et al. [47] elaborated a detailed security analysis on consumer and IIoT devices, such as a home automation system and a smart meter. Then, they discussed security solutions and mitigation methods. They selected the Itron Centron smart meter as the case study and discovered its vulnerabilities. Although they have given vulnerability analysis on some specific industrial or consumer IoT devices, there is still lack of comprehensive analysis of the attack surfaces, attack vectors, and prevention methods for a typical IIoT device.

Attacks to industrial control systems are a major threat of modern cyber warfare [24]. After Stuxnet [29] was revealed, new attacks against industrial control systems continued to occur. Havex [28] is a remote access Trojan discovered in 2013. It is part of a wide range of espionage activities for industrial control systems that are widely used in various industries. Havex has impacted up to 2,000 infrastructure sites, specifically for energy grid operators, large power generation companies, oil pipeline operators, and industrial equipment providers [28]. BlackEnergy [26] first appeared as a DDoS tool in 2007. Its improved version—BlackEnergy3 (BE3)—participated in a cyber attack in Ukraine in 2015, causing power outages [23]. Although BE3 has no direct effect on powering down, BE3 was used in the early stages of the attack to gather information about the industrial control systems environment and was likely to be used to compromise the network operator's user credentials. CRASHOVERRIDE [27], also known as Industroyer, is the first malware specifically designed for the electric grid. It was used in a cyber attack on December 17, 2016, which powered off the Ukrainian transit station on the outskirts of Kiev, cutting off about one fifth of the power. TRISIS [30], also known as TRITON or HatMan, is a malware variant of the Schneider Electric's Triconex Safety Instrumented Systems (SIS) controller. It was discovered in December 2017 by FireEye's network security company Mandiant. FireEye determined that TRITON was specifically designed to interact with these SIS controllers and believed that the participants or organizations behind the attack may be trying to develop the ability to physically damage and stop operations on the organization's equipments.

Other than exploring already well-known attacks in the attacking surface of a concrete case study, in this work we are also going to refine a particular kind of attack—Mirai—to tailor it to target industrial control systems more effectively. Mirai is malicious software that turns networked devices running Linux into *bots* and creates *botnets* of these compromised IoT devices, aiming at launching large-scale DDoS attacks. It has been used to infect a large number of online consumer devices such as IP cameras and home routers, and to launch serious attacks on servers of the DNS provider Dyn, resulting in inaccessibility of several high-profile websites such as GitHub, Twitter, and Reddit [14]. The proliferation of Mirai fully illustrated the serious insecurity problem of IoT devices. A lot of research work analyzing the working mechanism of Mirai and the harm of such kind of botnets has been studied since then [5, 22, 40]. However, to the best of our knowledge, we have not found related work considering that Mirai and its variants may be applied to IIoT systems. Moreover, how to make Mirai attack a stealthy attack and discuss defenses is also a unique contribution of our work.

3 OVERVIEW OF VULNERABILITIES

The focus of this article is an experimental analysis that aims at attacking the system introduced in Section 2.1, with the target of gaining control of its actuators and stealing reserved information from the device. Figure 3 shows the two actuators. Figure 3(a) shows the actuator whose arm is operating in the safe area of operation. Figure 3(b) depicts the actuators whose arm is in a critical configuration. The experimental analysis aims at violating the system and making its actuators working in the latter configuration (i.e., Figure 3(b)).



(a) (b)

Fig. 3. (a) A servo actuator in our system during normal operation. (b) The actuator reaching a hazardous condition once the system has been compromised.

3.1 Vulnerability Analysis and Attack Scenarios

The experimental analysis starts by identifying the main vulnerabilities of the target system, guided by *Open Web Application Security Project* (OWASP) *Top 10 IoT Vulnerabilities 2018* [31], which categorizes the vulnerabilities of a typical IIoT device. The IoT vulnerabilities identified by Open Web Application Security Project (OWASP) since 2014 provides a summary of attack surfaces associated with the top IoT vulnerabilities. In this work, we address all of the vulnerabilities identified in the latest edition (i.e., the 2018 edition).

The IIoT devices we focus on are mainly used as smart controllers for industrial applications. In this case, the primary objective of the attackers is to misuse the IIoT devices to harm the system. In fact, the possibility of physically damaging the system through a cyber attack is a peculiarity of IoT that is even more critical for IIoT systems. We also analyze privacy concerns, as one of the attack scenarios we are going to carry on will also steal information from the device. However, we do not consider some attack surfaces that are more important in the consumer IoT field, such as *Insecure Mobile Interface*. To this end, we focused on *Insecure Web Interface*, *Insufficient Authentication*, *Lack of Transport Encryption*, *Insecure Network Services*, and *Insecure Software/Firmware*. Then, for each attack surface, we identify its attack vectors based on our goal of gaining control of the actuators. Then, we define different attack scenarios that use the identified attack vectors. Table 1 summarizes the results of our analysis in the first two columns. The third column reports, for each attack surface, the concrete attack scenarios that will be applied to the reference device.

The final target of our experimental analysis is to gain control of the system, interfering with the servo motors moving their arms out of normal operation modes and stealing restricted information. Figure 4 shows the different path we will follow to compromise the system. We exploit the five attack surfaces identified in Table 1 throughout three attack scenarios, namely, *Web Management System*, *Remote Software Update*, and *Mirai Attack*. In each scenario, we alternatively exploit different attack vectors to gain control of the system, to provide a concrete example of threats affecting industrial devices. During our experiments, we will try to gain control of the *Web Management System* by exploiting weak authentication, SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). It is worth noticing that all of these techniques will, however,

Table 1. Summary of the Attack Surfaces, Vectors, and Scenarios Identified in the Case Study Presented in This Work with Respect to the 2018 OWASP IoT Top 10 Vulnerabilities

Attack Surfaces (OWASP IoT Top10 2018)	Attack Vectors	Attack Scenario
1. Weak, Guessable, or Hard-Coded Passwords	Use weak password to login	Web Management System
2. Insecure Network Services	Compromise the device through vulnerable network services (e.g., Telnet, HTTP, SSH)	Mirai attack
3. Insecure Ecosystem Interfaces	SQL injection	Web Management System
	XSS	
	CSRF	
4. Lack of Secure Update Mechanism	Upload malicious packages in absence of authentication	Remote Software Update
5. Use of Insecure or Outdated Components	Compromise the device through vulnerable network services (e.g., Telnet, HTTP, SSH)	Mirai Attack
	SQL injection	Web Management System
	XSS	
	CSRF	
6. Insufficient Privacy Protection	Capture data via network due to lack of encryption	Remote Software Update
	Leak of reserved information	Mirai attack
7. Insecure Data Transfer and Storage	Capture data via network due to lack of encryption	Remote Software Update
8. Lack of Device Management	Upload malicious packages in absence of authentication	
9. Insecure Default Settings	Use of well-known default username and password	Web Management System
10. Lack of Physical Hardening	Lead actuators to unsafe states	Mirai attack

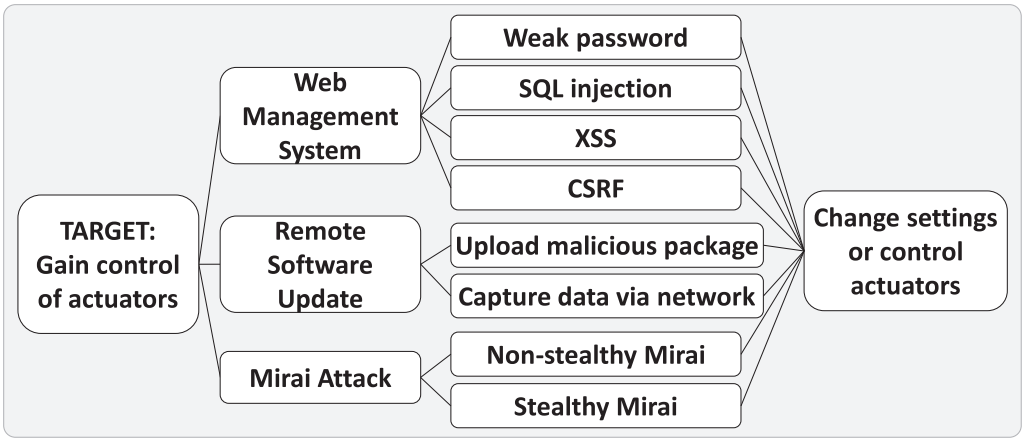


Fig. 4. Tree-based representation of our attack strategy. The root represents our final target (i.e., to attack the system actuators), whereas each node represents a possible action. Each path from the root to the leaves identify a specific attack.

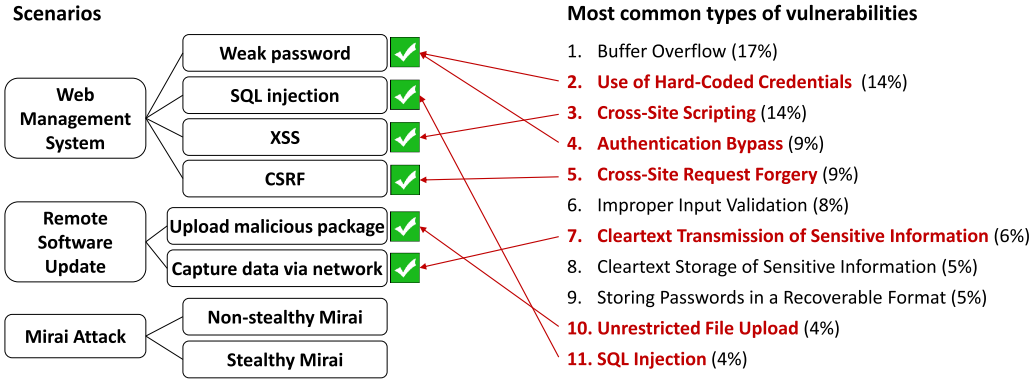


Fig. 5. Correlation between the scenarios explored in this work and the most common types of vulnerabilities identified in Andreeva et al. [4]. The types of vulnerability are sorted according to the percentage (reported in brackets) of vulnerabilities identified and belonging to the type.

require either a poor system configuration or to access the control settings of the system (e.g., inside job). The *Remote Software Update* scenario will require to upload a malicious package and to capture system information for further compromising the system. Finally, we will present a variation of Mirai attack targeting actuators instead of performing DDoS on the system. Since a defender can fairly easily identify this kind of attack, we will extend the attack to make it stealthy by utilizing both *Linux Kernel Module* (LKM) *rootkits* and *Stunnel*.

3.2 Scenarios Effectiveness

The scenarios described previously share the necessity of gaining access to the system, with sufficient privileges for the attacker to send commands to actuators. As such, it is necessary to evaluate whether there exists a concrete possibility of such an eventuality to happen, such as to evaluate if "real-world" IIoT and industrial control systems may be accessed by malicious users, as in our experimental analysis. Although our case study seems to be "synthetic," it is actually inspired by concrete vulnerabilities. In the following, we provide the first contribution of this article: an analysis of the vulnerabilities currently exploitable on Internet-connected IIoT devices and industrial control systems. Furthermore, we illustrate how such vulnerabilities shares the same principles of those described in the scenarios considered in our experimental analysis. Indeed, any of the presented scenarios may be implemented whenever the attacker can have physical access to the system to control. For instance, the attacker may be an employee or a technician hired by the attacked company. The attacker may as well exploit social engineering techniques to gain physical or remote access to the system. However, we do not consider such cases in our threats analysis.

Industrial control systems are made of devices managed by complex software stacks. Such a complex infrastructure is often a source of exploitable vulnerabilities. Even though in recent years much more attention has been given to security in such types of applications, new vulnerabilities are constantly found in industrial control systems. Kaspersky Lab provided some statistics about the vulnerabilities found in industrial control systems in 2015 [4]. They identified 189 vulnerabilities published in 2015: more than one new vulnerability found every 2 days. Of such vulnerabilities, 49% have been considered critical, whereas the percentage of medium severity vulnerabilities was around 42%. In addition, for many vulnerabilities (e.g., hard-coded credentials), an exploit code is not needed at all to obtain unauthorized access to the vulnerable system. The study also pointed out the high differentiation of vulnerabilities affecting industrial control systems. Figure 5 reports the most common types of vulnerabilities identified by the study, and it correlates them to the

Table 2. Results Obtained Through the Shodan Search Engine While Searching for Industrial Control Systems and IIoT Devices Connected to the Internet and Currently Exposing Insecure Services

IIoT Device Producer	Devices Found	Insecure Services Found				
		HTTP	Telnet	SSH	Total	Percentage
Moxa	10,462	169	435	9	613	5.86
Latronix	15,441	5	3,140	0	3,145	20.37
Ubiquiti	618,078	175	682	0	857	0.14
Others	459	231	7	1	239	52.07
Total	**644,440**	**580**	**4,264**	**10**	**15,301**	**2.37**

The results refer to a set of queries perfomed on March 15, 2019.

scenarios presented in our experimental analysis. The attack vectors used in our scenarios cover 60% of the vulnerabilities identified in Andreeva et al. [4].

Even though the analysis in Andreeva et al. [4] is 4 years old, many of the identified vulnerabilities are still exploitable today. Furthermore, attack surfaces similar to those identified in our experimental analysis have been very recently found in another study presented by Kaspersky Lab [18]. This latter study identified seven vulnerabilities in a well-known IIoT middleware. The identified vulnerabilities allow a remote attacker to execute any command on an IIoT device managed by the affected middleware, as well as gaining root access to the system.

It is easily noticeable that the third scenario considered in our experimental analysis (i.e., Mirai attack) is not related to any of the identified vulnerabilities. This is because Mirai attack exploits insecure network services. Of course, Mirai attack may be performed by an inside malicious user or after acquiring root access to the system. Thus, it may be performed as a variation of the first two scenarios and by exploiting the same vulnerabilities. However, Mirai attack may be performed also by exploiting vulnerabilities of insecure network services that may be exposed to the Internet by an IIoT device. For this reason, we searched for how many IIoT devices today are exposing insecure network services to the Internet. A similar analysis was carried out in Al-Alami et al. [2]; however, it was restricted only to a very limited geographical area (i.e., Jordan). We performed a similar search but extended it to the entire world. We relied on the Shodan search engine to discover devices and services on the Internet [43]. To restrict our search to only IIoT devices, we filtered devices by the producer, keeping only those producing IoT devices specialized for industrial applications. Then, we refined our search to identify how many of these devices were exposing three of the most notable insecure network services: HTTP, Telnet, and SSH. Table 2 reports the results of our scanning. We were able to found more than 600,000 IIoT devices connected to the Internet. Of these, 2.37% were using insecure services. As such, they are potential victims in the third scenario (cf. Figure 4) presented in our experimental analysis.

The presented data highlights how the case study shares its attack surface with many devices currently deployed in industrial plants. In the following, we show application of the three scenarios reported in Figure 4, each of them exploiting the attack vectors just discussed.

4 WEB MANAGEMENT SYSTEM VULNERABILITIES

Embedded web servers are often integrated into smart networked devices for remote configuration and operation, such as embedded web configuration interfaces in IP cameras or routers. For industrial applications, embedded web management systems can be used to configure or control IIoT devices, or to operate the subordinate actuators. The IIoT devices in our reference system are equipped with a web management system integrated into the controllers (i.MX6 devices) by

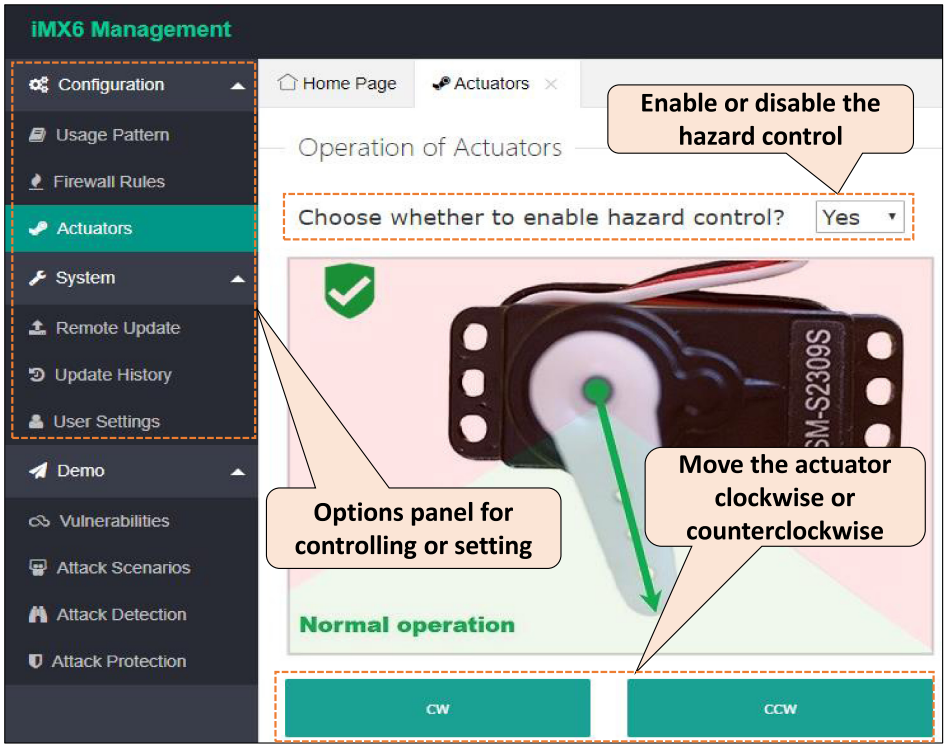


Fig. 6. Web interface of the management system, which can be used to set or control the actuator.

using the Boa embedded web server and the SQLite3 database. The web interface contains three functional modules: the *main* module to control the login and to display the home page, the *config* module for the configuration and operation, and the *system* module to change the user account information. The database is composed of six tables. Table `t_user` stores the user accounts information; table `t_session` stores the current session's information; `t_log` stores the recorded revision information of the web management system; and tables `t_pattern`, `t_firewall`, and `t_servo` store the information about the configuration of the usage patterns, firewall, and actuator.

We identified four types of attack vectors available for accessing the web interface and then to gain control of the actuators, as introduced in Figure 4.

4.1 Use a Weak Password for the Login

A large amount of web interfaces adopt similar user names and weak passwords, such as `admin|admin`, `admin|123456`, and `system|system`. For company servers, strong user authentication is usually enforced, and statistics show [20] that 15% of IoT devices are accessible using the producer default credentials. Furthermore, 14% and 9% of IIoT devices were found to have hard-coded credentials and authentication bypass issues, respectively (cf. Figure 5).

Devices in our case study have two user accounts: `admin|admin` and `imx|123456`. Assuming an attacker accesses our web management system for the first time, it can easily guess the user name and password. Once the attacker successfully logs into the system, it can access the administration page shown in Figure 6. The tabs on the left show options that the attacker can utilize to configure or control the system, such as setting the usage pattern or firewall, changing the user name or password, or even forcing the actuator to move into dangerous areas.

4.2 SQL Injection

Whenever a website asks the user to input data, SQL injection attacks are possible, and 4% of the industrial control systems exposed to the Internet are vulnerable to such threat [4]. In SQL injection, the attacker inputs an SQL statement rather than justifiable data. Then, the website will unknowingly run the given statement on the database. If there is no method to validate the legality of the user input, the attacker can enter a malicious input instead of a well-formed username and password. Thus, for instance, it might input the following.

```
username: admin | password: 1'OR `1'=`1
```

Then, the SQL statement will become as follows.

```
SELECT id,username,password
FROM t_user
WHERE username=`admin'
AND password=`1' OR `1'=`1'
```

The preceding SQL statement is valid and returns *all rows* from the table `t_user`, since `OR‘1’=‘1’` is always TRUE. The attacker can successfully login to the web interface without knowing the correct user credentials. Then, it can gain control of the actuators as described in the preceding attack vector.

4.3 Cross-Site Scripting

As introduced previously, 14% of the industrial control systems are vulnerable to Cross-site Scripting (XSS) attacks where a malicious attacker exploits vulnerabilities of a website that does not discard the data submitted by the user or filters the data insufficiently. The attacker embeds its code in the web page, forcing other users to execute the corresponding embedded code when they access the same website. The attacker may steal user accounts, obtain enterprise data, transfer funds illegally, hang Trojans on websites, and so on [15].

```
1 <script>
2   var Str=document.cookie;
3   var a =document.createElement('a');
4   a.href= "javascript:alert(Str)";
5   a.innerHTML="<img src='../page/xss.jpg'>";
6   document.body.appendChild(a);
7 </script>
```

Listing 1. Implementation of an XSS attack. Line 2 retrieves the section identifier from the browser. Line 4 displays it to the attacker whenever the link described in line 5 is clicked.

In our experiment, the attacker injects malicious JavaScript (Listing 1) to the web management system with a hyperlink able to read the visitor's Session Identifier. If a legal user visits the web management system and clicks the image with a hyperlink, its Session Identifier can be stolen (shown in Figure 7). The attacker may utilize this Session Identifier to access the web management system and launch attacks to the actuators.

4.4 Cross-Site Request Forgery

CSRF is an attack mode that holds users to perform unintended operations on web applications that are already logged in. Around 9% of industrial control systems are reportedly vulnerable to such attack [4]. Compared to XSS, CSRF exploits the system's trust in the page browser, whereas

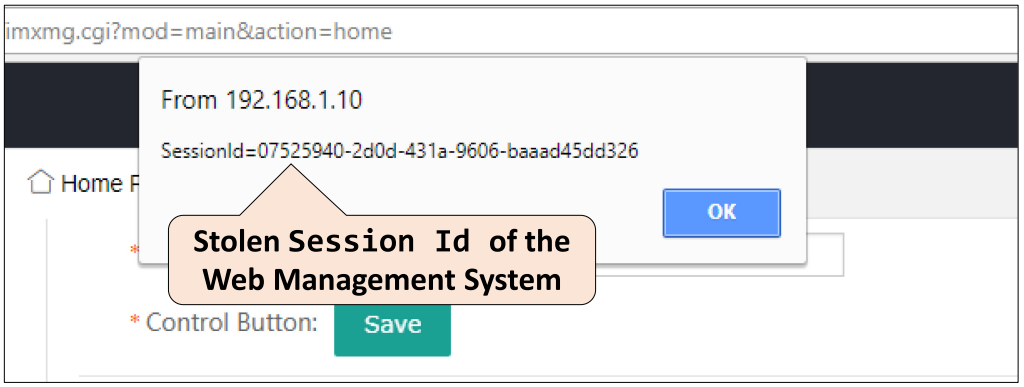


Fig. 7. Injecting malicious Javascript code, the attacker can retrieve another user's Session Identifier, using it to access the system.

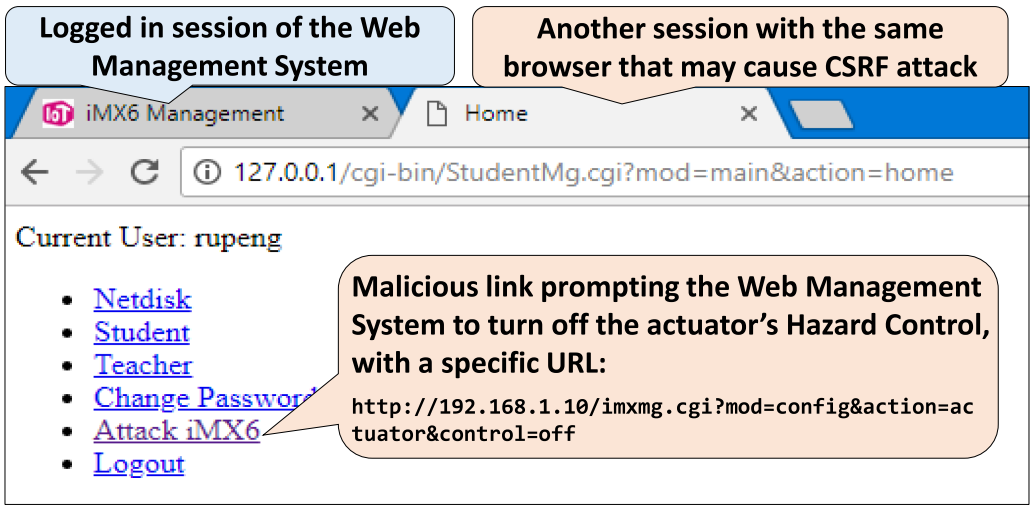


Fig. 8. Leaving a web management system logged in session in a browser exposes the system to CSRF attacks.

XSS takes advantage of the system's trust on the user. As shown in Figure 8, an authorized user starts a tab and logs into the web management system. It will get a Session Identifier from the web management system for keeping its login status. Then it makes a new tab with the same browser and logs into another website (attacker) with a malicious link that can hack the web management system. For example, the link `Attack iMX6` in Figure 8 prompts the web management system to disable its hazard control of the actuator with a specific URL. Since the browser already obtained the Session Identifier from the web management system, the preceding URL will be executed successfully without requesting to login.

5 REMOTE SOFTWARE UPDATE VULNERABILITIES

The second scenario requires that the IIoT devices firmware and software may be updated regularly for maintaining security and stability. Some vulnerabilities affect the update mechanisms, and they can be exploited by the attackers, such as lack of authentication or encryption, as listed in Table 1.

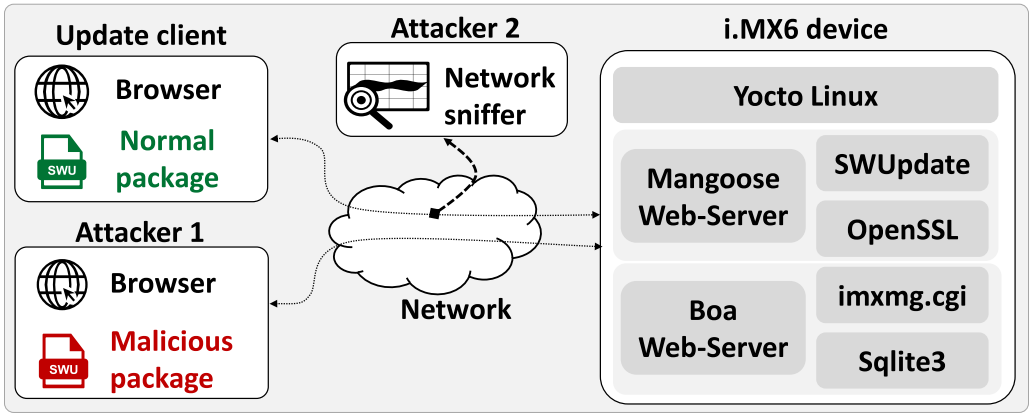


Fig. 9. Overall setup for the remote software update experiments. (1) The attacker can upload a malicious package if the update mechanism of the system is without authentication. (2) The attacker can capture sensitive data via the network if the transport is not encrypted.

We set up a remote software update scenario for i.MX6 and explored the possible vulnerabilities that can be exploited to attack the actuators. The device uses *SWUpdate*[2] as the updating engine to implement the remote software update system. SWUpdate is a Linux update agent aiming at providing an efficient and safe way to update embedded systems. It supports local and remote updates and multiple update strategies. The setup used to experiment this scenario is shown in Figure 9. The i.MX6 device is running Yocto Linux, a Mangoose web server for remote access to the update system, the SWUpdate tool, and an OpenSSL module for authentication and encryption. It is connected to a LAN accessible by both the attacker and non-malicious users. The updating target is the *imxmg.cgi* program used in the web management system for manipulating the settings and operations.

5.1 Upload Malicious Packages in Absence of Authentication

A device must be safely updated. As such, it must be able to verify that any update comes from known sources and was not corrupted to introduce some malware. SWUpdate provides a signature to the updated package and verifies the hash code of every single image. This protection mechanism ensures that the installer can accept only an update package generated by a verified source. However, SWUpdate does not automatically enforce it. Whenever the device is deployed on a network without adopting such a solution, attackers will be free to upload any potentially dangerous package. Data reported in Section 3.2 show that 4% of IIoT devices suffer vulnerabilities allowing unrestricted file upload.

A malicious package of the program aiming at moving the actuators to dangerous areas has been successfully uploaded to i.MX6 through the remote update system due to no authentication check. Consequently, the attacker can attack actuators in the way of uploading malicious packages.

5.2 View Data via Network Due to Lack of Transport Encryption

The update file might be only authenticated but not encrypted, as may potentially happen for 4% of IIoT systems (cf. Figure 5). In this case, the attacker can still capture the file over the network. The attacker only needs a simple network sniffer to monitor the connection between the update client and the IIoT device. Figure 10 shows non-encrypted/encrypted packages with the same payload

[2]https://github.com/sbabic/swupdate.

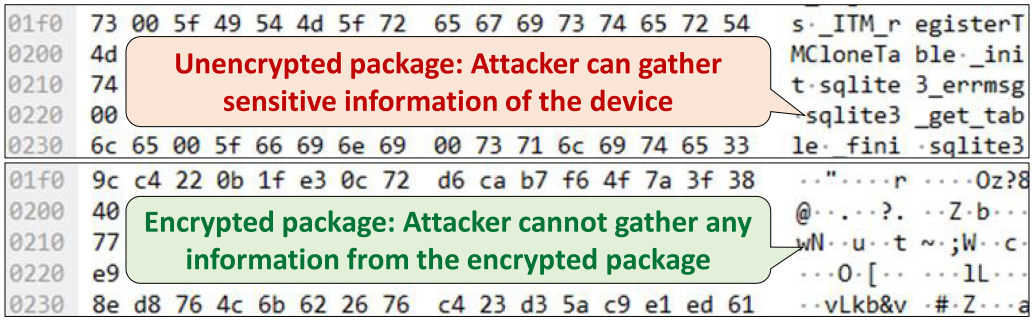


Fig. 10. The attacker can gather sensitive information from the unencrypted package simply sniffing, for example, the database SQLite3 used in the device.

captured by sniffing the network. The unencrypted package allows the attacker to gather information about the target system, such as its database used by the device. Attackers can further exploit this information to attack other parts of the device, such as performing SQL injection attacks on the web management system, and thus gain control of actuators as discussed in Section 4.

6 MIRAI ATTACKS FOR INDUSTRIAL SCENARIOS

Mirai attack exploits open and insecure network services (i.e., Telnet or SSH) in poorly protected devices to turn them into *bots* controlled by the attacker. Insecure services, exposing production systems to the Mirai attack, have been found running on more than 15,000 devices spread around the world by our search on the Shodan search engine (cf. Table 2). Mirai's self-propagating abilities make it extremely effective against many IoT devices [40]. It can infect tens of thousands of devices and exploit them by setting up DDoS attacks against a target victim. In an industrial scenario, connected devices may act as controllers. Once Mirai infects one of the devices, it can quickly spread Mirai to the entire network. Since the source code of Mirai is public, it is possible to modify it to target a specific manufacturer's device for other abuses, different than DDoS. We hereby first describe the variation we propose to Mirai attack. Then, we present our strategy to make the proposed attack stealthy. Finally, we evaluate the effects of the proposed attack considering the system's availability, integrity, and confidentiality. We released the code used in this section via open source[3] (i.e., the code extending the Mirai attack) and the code necessary to perform the analysis of the case study described in the following.

6.1 Gaining Control of Actuators and Stealing Information Through Mirai Attack

We modified the source code of Mirai, making it able to gather sensitive information from the device and to gain control of the system actuators other than performing DDoS attack against the system remote control center. The actuators are attacked to disturb the production tasks and cause business losses. Then, attacking the remote control center can increase the difficulty of troubleshooting the attack, thus making it more effective. Still, DDoS does not avoid the attack to be identified. Thus, we turn Mirai into a stealthy attack. The stealthy Mirai attack is less likely to be detected and mitigated.

Figure 11 reports the main steps of the Mirai attack targeting the two actuators and the remote control interface of the system. Red dashed arrows and numbers indicate the operations to perform non-stealthy Mirai attack; green solid lines and numbers identify the operations necessary to make

[3]Source code of the Mirai attack for IIoT is available at https://gitlab.com/asset-sutd/public/mirai4iiot.

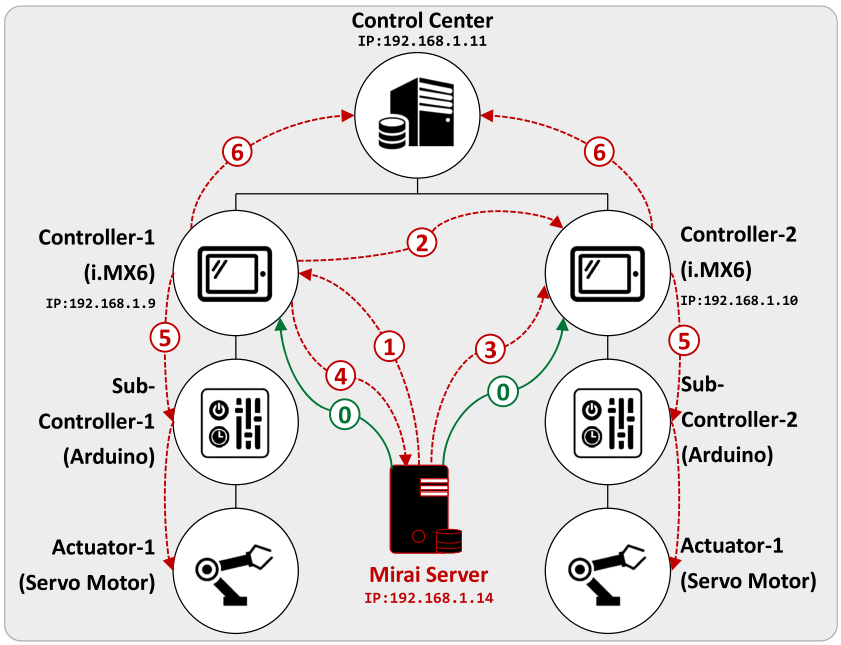
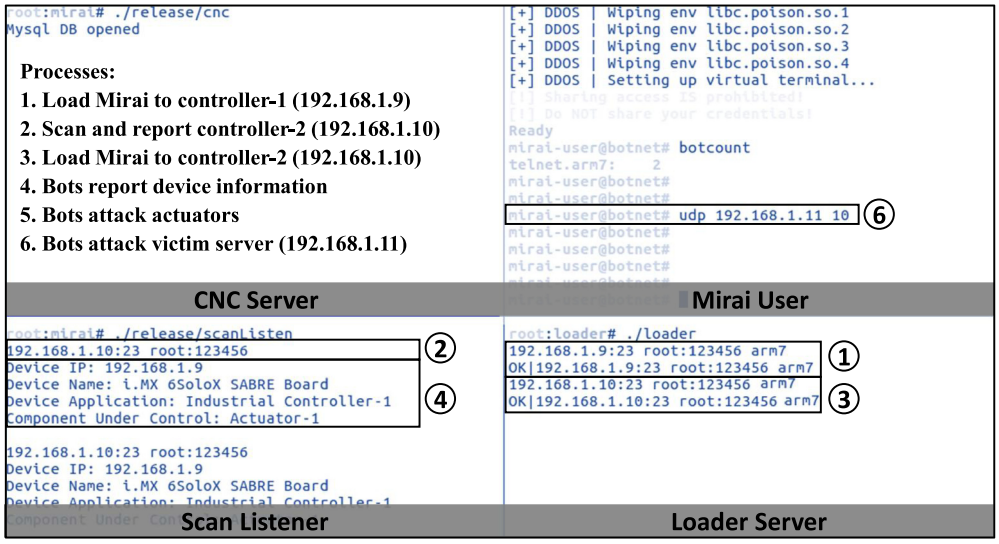


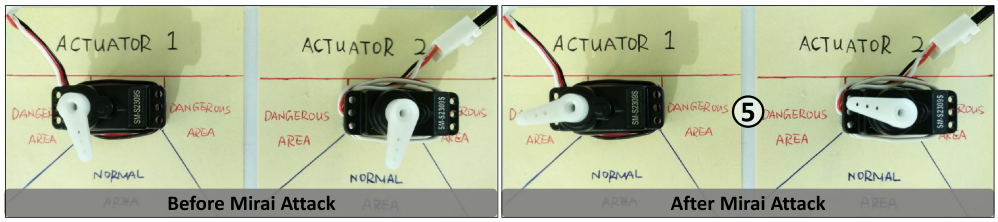
Fig. 11. Steps of the Mirai attack performed to gain control of the actuators. The red dashed arrows and numbers depict non-stealthy Mirai attack. The green arrows show the actions to be performed before a Mirai attack to hide it.

Mirai attack stealthy. The non-stealthy Mirai attack in our case study proceeds as follows. Figure 12 shows the execution of the attack step by step:

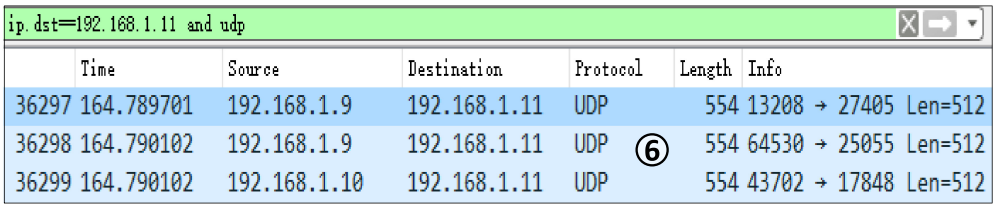
(1) *Load Mirai to Controller-1*: The *Loader Serve*r tries to login to the IIoT device (Controller-1, 192.168.1.9) by using the default username and password (e.g., `root|123456`) or by obtaining the username and password by brute forcing the IIoT device. However, even if IIoT devices do not use the weak authentication credentials, the attacker may also invade them by other means, such as insider threats or phishing attacks. After successfully logging in, the Loader Server asks the device to download the Mirai main program from the attacker's File Server by using TFTP. The Loader Server then prompts the device to run the Mirai main program, thus transforming the device into a bot.
(2) *Scan and report Controller-2*: The bot starts using Telnet to connect to the Command and Control (CNC) Server. Meanwhile, the bot scans other devices in the LAN, reporting the obtained result (Controller-2, 192.168.1.10 with `root|123456`) to the Scan Listener.
(3) *Load Mirai to Controller-2*: The Loader Server retrieves the IP address, username, and password of Controller-2 from the Scan Listener. Then, it injects the Mirai botnet to Controller-2 using the same process used to infect Controller-1.
(4) *Bots report device information*: Both controllers in the system turned into bots. Then, they can gather sensitive information about the devices to report to the Mirai server. In our case, the Mirai botnet reads the *configuration file* of the device and report the detailed device configuration and its usage information, as shown in ④ in Figure 12(a).
(5) *Bots attack actuators*: The bots can send commands to the actuators, forcing them to perform unsafe actions. In our case, the Mirai botnet moves the robot arm of the servo motor to the dangerous region, as shown in Figure 12(b).



(a)



(b)



(c)

Fig. 12. Experimental results of non-stealthy Mirai attack. The numbers are consistent with the ones in Figure 10. (a) Servers controlled by the attacker, including the CNC server, scan listener, and loader server. (b) Actuators working in safety before Mirai attack, on the left. Dangerous configuration of actuators forced by the Mirai attack on the right. (c) The network package captured on the victim (192.168.1.11) after launching a DDoS attack shows that the victim was suffering from UDP flood from the bots (192.168.1.9 and 192.168.1.10).

(6) *Bots attack victim server*. The attacker can promote the bots to mount DDoS attacks targeting the remote control interface (192.168.1.11), as shown in Figure 12(c). A considerable number of DDoS packages will make the remote control interface not work correctly, increasing the difficulty of troubleshooting. This, in turn, provides more time to Mirai attack for compromising the physical actuators.

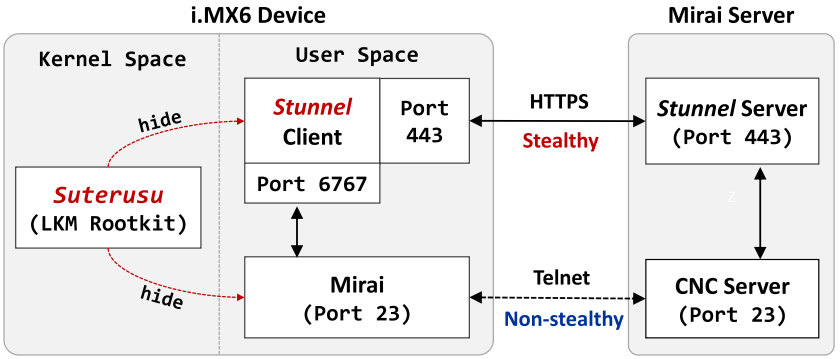


Fig. 13. Architecture to make Mirai attack stealthy. The non-stealthy Mirai attack adopts Telnet to communicate with the CNC server, which will show Telnet traffic on the network. Stunnel hides the Telnet traffic in the HTTPS traffic. Meanwhile, the Suterusu rootkit hides the traces of Mirai and Stunnel to potential intrusion detection mechanisms.

Although Mirai attack can spread widely, it can be easily detected by monitoring anomalous running processes, established network connections, and network traffic [40]. Properly hiding these three kinds of trace information allows transforming Mirai attack into a persistent stealthy attack. This kind of attack will grant more time to the attacker to perform more physical damage to the system. Such extra time may be crucial for the success of the attack, as physical processes usually evolve more slowly than cyber processes. Thus, the effects of an attack to a physical process require more time to manifest.

6.2 Making Mirai Attack Stealthy

A stealthy attack operates quietly, hiding evidence of attackers' actions, including processes, network connections, and traffic. A *rootkit* is a kind of special malicious software that can operate stealthy [50]. They can be used in combination with Trojans, back-doors, and other malicious programs. LKMs allow extending the operating system (OS) kernel dynamically. LKMs have the same permissions as code compiled into the kernel, thus giving them a lot of flexibility and power. However, maliciously written kernel module can subvert the entire OS. A kernel module has the same privileges as any other piece of kernel code: it may hijack Linux systems by redirecting system calls or by replacing system call handlers with its own wrappers to hide files, processes, and network connections. We employ the LKM rootkit (*suterusu*) to hide the processes and network connections used by Mirai attack. An LKM rootkit, such as *suterusu*, can be inserted into the Linux kernel as a module. It is designed to hide its own existence. Consequently, the system users cannot discover the rootkit presence through any system tools, kernel module lists, or kernel logs. The rootkit runs in the kernel space covertly and provides the ability to hide information for the Mirai process running in the user space, as shown in Figure 13. The existence of Mirai is then hidden to the system users. Thus, the system utilities cannot find the running process of Mirai, and `netstat` cannot find the established network connections of Mirai.

Hiding the process and the connection information is not enough to make the attack sufficiently concealed. The Telnet service used by the communication between the bot and the CNC server exposes the traffic information that the system can use for anomaly detection. We adopt *Stunnel*, an open source tool used to provide a universal TLS/SSL tunneling service. It converts an ordinary, unencrypted Telnet connection to a secure, encrypted HTTPS connection, as shown in Figure 13, obtaining the behavior depicted in Figure 14: HTTPS traffic will be evaluated as a normal data

ip.src==192.168.1.10 or ip.src==192.168.1.14

No.	Time	Source	Destination	Protocol	Length
5	0.006922807	192.168.1.10	192.168.1.14	TCP	66
6	0.008498645	192.168.1.10	192.168.1.14	TELNET	70
7	0.008536267	192.168.1.14	192.168.1.10	TCP	66
8	0.010279325	192.168.1.10	192.168.1.14	TELNET	79

(a)

ip.src==192.168.1.10 or ip.src==192.168.1.14

No.	Time	Source	Destination	Protocol	Length
5	0.024173578	192.168.1.14	192.168.1.10	TCP	74
6	0.024273891	192.168.1.14	192.168.1.10	TCP	74
7	0.025992068	192.168.1.10	192.168.1.14	TCP	66
8	0.028708330	192.168.1.10	192.168.1.14	TLSv1.2	313

(b)

Fig. 14. Behavior observed by sniffing the network during our experiments. (a) Unhidden Telnet traffic captured by the network sniffer. (b) Hidden traffic in HTTPS captured by the network sniffer.

stream, and the firewall will not block it. In addition, its encrypted nature makes attacks subtler and less noticeable.

Combining with LKM rootkit and Stunnel, we completely hide Mirai's process, network port, and network traffic information. In addition, information about the existence of rootkits and Stunnel can also be hidden. Since the LKM rootkit runs at the same level as the kernel, it is difficult to detect by using general detection methods [49]. Subsequently, the stealthy Mirai attack may be hidden in the IIoT system for long-lasting attacks, providing all of the time the attacker may need to cause physical damages to the industrial system.

6.3 Evaluation of the Proposed Mirai Attack

We base our evaluation of the proposed Mirai attack variations described earlier on the *Availability*, *Integrity*, and *Confidentiality* (AIC) triad by considering the three issues separately.

The issues caused to system *availability* by the proposed variation of Mirai attack are twofold: one issue is strictly related to security, whereas the second concerns the safety of the system. From the security point of view, Mirai attack floods the system, making it unavailable to the legitimate user. This provides more time for the attacker to perform malicious actions involving physical parts of the IIoT system. This is necessary because the reaction time of physical (e.g., mechanical) processes is significantly slower than cyber actions. As such, the attacker may need extra time to carry out the attack. The proposed attack is able to provide such extra time by compromising the availablity of the system interfaces through flooding. However, Mirai can compromise the system availability from the safety point of view by forcing operations that cause physical damage to the system, such as by moving the actuators into unsafe configurations, as shown in our experimental analysis. It is important to note that the safety and security system availability issues are connected to each other. In fact, flooding the system (security issue) gives the attacker time to bring the physical parts of the system into an unsafe configuration (safety issue). This is due to the strong correlation between security and safety, which is a peculiar characteristic of IIoT systems.

The *integrity* of the system is compromised by the proposed Mirai attack. As long as the actuators are controlled by a malicious user, the real-time constraints characterizing industrial systems can no longer be guaranteed. Furthermore, gaining control of the physical components of the system, forcing them to unsafe states, violates the integrity of the system. It is worth noting also that the integrity issues are located at the intersection of the system's safety and security, highlighting once more the strong connections between the two concepts in IIoT systems.

During our experimental analysis, Mirai attack allowed stealing reserved information from the attacked devices, thus violating the *confidentiality* of the system. However, in our experimental analysis, the stolen information was the device configuration, and devices may contain largely more sensible information. This is particularly true for systems embracing the most recent manufacturing trend. For instance, when the IIoT system implements agile or reconfigurable manufacturing paradigms, the configuration of the production system carries information about the product being produced [35]. Consequently, an attacker able to steal the device configuration may also end up stealing information about the product. Such type of leak may lead to significant economic losses, especially when the products are covered by intellectual property, patent, and so forth.

The proposed variation of Mirai attack is able to violate the Availability, Integrity and Confidentiality (AIC) triad. Furthermore, all violations are specifically oriented to the specific requirements typical of indutrial systems and thus of IIoT systems.

7 PROTECTION STRATEGIES

We can classify the preceding vulnerabilities into three categories: defective system design, malware injection, and stealthy attack. Therefore, the protection strategies are discussed from these three aspects, respectively.

7.1 Defective System Design

The vulnerabilities targeting the *web management system* and the *remote software update* (i.e., weak credentials, authentication bypass, SXX, CSRF, SQL injection, unrestricted file upload, and cleartext transmission) are mainly due to a defective system design. In other words, these attack vectors can be reduced by improving system integrity. For example, developers may adopt a mechanism for enforcing users to use strong passwords or can optimize the code of the database operations to defend devices from SQL or JavaScript injection. Ensuring adequate authentication and encryption is an effective means of protecting the *Remote Software Update* from attacks.

7.2 Malware Injection

Embedded devices generally do not have a mature malware detection mechanism, already used on conventional computing platforms, such as servers and desktops [38]. This allows Mirai-like malware to easily infect a large number of IoT devices without being detected. Implementing a set of malware detection and protection mechanisms for IoT devices is becoming compulsory. Commonly used malware detection methods are software-based solutions using anti-virus software to detect malicious files. Detection is carried on by signature-based techniques or semantics-based approaches [21]. These solutions typically work at the OS level and effectively discover and remove most of the malware. We experimentally tested that it is possible to identify non-stealthy Mirai attacks as abnormal system processes by adopting a simple whitelist-based detection method.

7.3 Stealthy Attack

The non-stealthy Mirai attack is able to elude Operating System (OS)-level anti-virus software by using an LKM rootkit running at the same level as the OS. Hypervisor-level detection methods,

such as Virtual Machine Monitor (VMM)-based hidden process detection, have been proposed to overcome the effects of the LKM rootkit [16]. Such methods introduce a hypervisor isolated from the actual OS of the device to prevent OS-level rootkit attack. Concerning the protection of IoT devices discussed in this article, there are two limitations to this solution. First, adding a VMM to each embedded device negatively affects the design complexity, making it unsuitable for practical applications. Second, the hypervisor-level VMM itself can be the attack target and can be compromised by the rootkit [34].

The disadvantages of software-based solutions lead to the introduction of hardware-based methods as promising solutions that are less likely to be eluded [48]. State-of-the-art hardware-based malware or rootkit detection solutions can be placed into two categories: *data-centric* and *program-centric* [50] solutions. Data-centric approaches detect the integrity of data at the hardware level by generating and verifying fingerprints of selected features. A common solution is to verify whether the redirections in the program control flow are legal through Control Flow Integrity (CFI) check [17]. A kernel rootkit detection solution using custom hardware components to sign the system call routines has been proposed [49]. The method has been proved to detect a variety of LKM rootkits effectively. These kinds of static signature-based methods provide low overhead and are easy to implement. Nevertheless, the existence of indirect jump instructions and inconsistencies caused by OS upgrades will affect their effectiveness [49]. Program-centric techniques focus on using low-level features such as micro-architectural events collected by hardware performance counters to model dynamic program behavior to distinguish between malicious and benign programs. By combining various machine learning methods, researchers [8, 33, 39] have shown that malware can be effectively detected by gathering and modeling hardware performance counter data. Alternative hardware low-level architectural information (e.g., memory address references, instruction opcodes) can also be used by machine learning modeling methods for detecting malware [19, 32, 48].

Although hardware-based detection techniques are considered to be more efficient and less likely to be evaded, they typically require the addition of hardware components at the processor level that are not usually available in the micro-controllers typically used in industrial applications. Thus, such methods will impose redesign and increasing costs. For this reason, they have not yet been practically applied in industrial contexts. The most recent processors for IoT devices are equipped with hardware performance counters that can be used for malware detection. However, most of the detection methods relying on such hardware features require virtualization mechanisms to protect the detection [45]. For this reason, most of the work in this area targeted the x86-based system, whereas devices used in industrial contexts usually adopt microprocessors, such as ARM architecture processors. These microprocessor-based systems behave very differently and usually have severe resource constraints as compared to x86-based ones. Therefore, developing specific detection mechanisms for IIoT devices that satisfy multiple objectives, including efficiency, system availability, and security, will be crucial future research directions. As such, the overhead required by virtualization technologies makes such methods impractical in the industrial context.

A valid alternative to virtualization may be provided by Trusted Execution Environments (TEEs) equipping different architectures, including ARM architectures used to build IIoT systems [12]. TrustZone is the TEE equipping ARM architectures. It is a technology that guarantees confidentiality and integrity of data and code running in a processor [25]. It is designed to be isolated from the traditional system, establishing a secure and trusted execution area in the processor. Such an environment is a promising alternative to VMMs for implementing detection methods based on the analysis of the hardware events, such as those proposed for general-purpose computing systems [45]. Such a solution should allow keeping the overhead low, as required by IIoT systems, while increasing the chances of detecting stealthy malware.

7.4 Limitations of Our Analysis

Our research considered most of the security risks in IIoT disclosed by OWASP and Kaspersky Lab. The aspects that were not analyzed by our system include the privacy concerns (i.e., the cleartext storage of sensitive information vulnerability disclosed by Kaspersky Lab), the insecure cloud/mobile interface, and the buffer overflow defect (i.e., the highest proportion of vulnerabilities in industrial control systems listed by Kaspersky Lab), and meanwhile, the poor physical security is only partially addressed. Most of these vulnerabilities are due to the designers' inadequate consideration of system security, and they should be fully deliberated and avoided during specific system design. Concretely speaking, all security-sensitive information should be stored correctly in encrypted form to avoid disclosure. The cloud/mobile service and all software components require security testing to avoid defects such as buffer overflows. Physical security needs to be clarified in specific application scenarios, such as ensuring that the data storage medium cannot be easily removed or reset, and ensuring that unsafe USB media are not used to access the system and the information within it.

8 CONCLUDING REMARKS

In this article, we presented an experimental analysis about the possible threats of a device representative for the IIoT field. We identified its attack surfaces to define some attack scenarios. Then, we presented a new version of Mirai attack, tailored explicitly for industrial scenarios, and we extended it to make it not easily identifiable to the defender. The experimental analysis highlighted the vulnerability of modern production systems. By penetrating the system, we were able to control actuators, thus potentially endangering the system itself, and, eventually, the people interacting with it. Finally, we summed up some possible state-of-the-art protection and mitigation strategies that may be taken into account for further research in the IIoT context to prevent the scenarios analyzed during our experiments.

REFERENCES

[1] Jinesh Ahamed and Amala V. Rajan. 2016. Internet of Things (IoT): Application systems and security vulnerabilities. In *Proceedings of the 5th International Conference on Electronic Devices, Systems, and Applications (ICEDSA'16)*. IEEE, Los Alamitos, CA, 1–5.

[2] Haneen Al-Alami, Ali Hadi, and Hussein Al-Bahadili. 2017. Vulnerability scanning of IoT devices in Jordan using Shodan. In *Proceedings of the 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes and Systems (IT-DREPS'17)*. IEEE, Los Alamitos, CA, 1–6. DOI: https://doi.org/10.1109/IT-DREPS.2017.8277814

[3] Cristina Alcaraz, Rodrigo Roman, Pablo Najera, and Javier Lopez. 2013. Security of industrial sensor network-based remote substations in the context of the Internet of Things. *Ad Hoc Networks* 11, 3 (2013), 1091–1104.

[4] Oxana Andreeva, Sergey Gordeychik, Gleb Gritsai, Olga Kochetova, Evgeniya Potseluevskaya, Sergey I. Sidorov, and Alexander A. Timorin. 2016. *Industrial Control Systems Vulnerabilities Statistics*. Technical Report. Kaspersky.

[5] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, et al. 2017. Understanding the Mirai botnet. In *Proceedings of the USENIX Security Symposium*. 1092–1110.

[6] Roland Bodenheim, Jonathan Butts, Stephen Dunlap, and Barry Mullins. 2014. Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices. *International Journal of Critical Infrastructure Protection* 7, 2 (2014), 114–123.

[7] Sujit Rokka Chhetri, Nafiul Rashid, Sina Faezi, and Mohammad Abdullah Al Faruque. 2017. Security trends and advances in manufacturing systems in the era of Industry 4.0. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'17)*.

[8] John Demme, Matthew Maycock, Jared Schmitz, Adrian Tang, Adam Waksman, Simha Sethumadhavan, and Salvatore Stolfo. 2013. On the feasibility of online malware detection with performance counters. *ACM SIGARCH Computer Architecture News* 41 (2013), 559–570.

[9] Rainer Drath and Alexander Horch. 2014. Industrie 4.0: Hit or hype? *IEEE Industrial Electronics Magazine* 8, 2 (2014), 56–58.

[10] Arvind Easwaran, Anupam Chattopadhyay, and Shivam Bhasin. 2017. A systematic security analysis of real-time cyber-physical systems. In *Proceedings of the 22nd Asia and South Pacific Design Automation Conference (ASP-DAC'17)*. IEEE, Los Alamitos, CA, 206–213.
[11] Alasdair Gilchrist. 2016. *Industry 4.0: The Industrial Internet of Things*. Springer.
[12] Joffrey Guilbon. 2018. Introduction to Trusted Execution Environment: ARM's TrustZone. Retrieved October 8, 2019 from https://blog.quarkslab.com/introduction-to-trusted-execution-environment-arms-trustzone.html.
[13] Rachana Ashok Gupta and Mo-Yuen Chow. 2009. Networked control system: Overview and research trends. *IEEE Transactions on Industrial Electronics* 57, 7 (2009), 2527–2535.
[14] Scott Hilton. 2016. Dyn Analysis Summary of Friday October 21 Attack. Retrieved November 6, 2018 from https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/.
[15] Isatou Hydara, Abu Bakar Md Sultan, Hazura Zulzalil, and Novia Admodisastro. 2015. Current state of research on cross-site scripting (XSS)—A systematic literature review. *Information and Software Technology* 58 (2015), 170–186.
[16] Xuxian Jiang, Xinyuan Wang, and Dongyan Xu. 2007. Stealthy malware detection through VMM-based out-of-the-box semantic view reconstruction. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*. ACM, New York, NY, 128–138.
[17] Arun Kanuparthi, Jeyavijayan Rajendran, and Ramesh Karri. 2016. Controlling your control flow graph. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST'16)*. IEEE, Los Alamitos, CA, 43–48.
[18] Kaspersky Lab. 2019. Security Research: ThingsPro Suite—IIoT gateway and device manager by Moxa | Kaspersky Lab ICS CERT. Retrieved March 22, 2020 from https://ics-cert.kaspersky.com/reports/2019/01/22/security-research-thingspro-suite-iiot-gateway-and-device-manager-by-moxa/.
[19] Khaled N. Khasawneh, Meltem Ozsoy, Caleb Donovick, Nael Abu-Ghazaleh, and Dmitry Ponomarev. 2015. Ensemble learning for low-level hardware-supported malware detection. In *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*. 3–25.
[20] Kirill Shipulin. 2017. Practical ways to misuse a router. *Positive Technologies*. Retrieved November 8, 2018 from http://blog.ptsecurity.com/2017/06/practical-ways-to-misuse-router.html.
[21] Clemens Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiao-Yong Zhou, and XiaoFeng Wang. 2009. Effective and efficient malware detection at the end host. In *Proceedings of the USENIX Security Symposium*, Vol. 4. 351–366.
[22] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. 2017. DDoS in the IoT: Mirai and other botnets. *Computer* 50, 7 (2017), 80–84.
[23] Michael J. Lee, Robert M. Assante, and Tim Conway. 2016. *Analysis of the Cyber Attack on the Ukrainian Power Grid*. Electricity Information Sharing and Analysis Center.
[24] Jon R. Lindsay. 2013. Stuxnet and the limits of cyber warfare. *Security Studies* 22, 3 (2013), 365–404.
[25] Bernard Ngabonziza, Daniel Martin, Anna Bailey, Haehyun Cho, and Sarah Martin. 2016. Trustzone explained: Architectural features and use cases. In *Proceedings of the IEEE 2nd International Conference on Collaboration and Internet Computing (CIC'16)*. IEEE, Los Alamitos, CA, 445–451.
[26] NJCCIC. 2017. BlackEnergy. Retrieved October 3, 2019 from https://www.cyber.nj.gov/threat-profiles/ics-malware-variants/blackenergy.
[27] NJCCIC. 2017. CRASHOVERRIDE. Retrieved October 3, 2019 from https://www.cyber.nj.gov/threat-profiles/ics-malware-variants/crashoverride.
[28] NJCCIC. 2017. Havex. Retrieved October 3, 2019 from https://www.cyber.nj.gov/threat-profiles/ics-malware-variants/havex.
[29] NJCCIC. 2017. Stuxnet. Retrieved October 2, 2019 from https://www.cyber.nj.gov/threat-profiles/ics-malware-variants/stuxnet.
[30] NJCCIC. 2017. TRISIS/TRITON. Retrieved May 1, 2020 from https://njccic.squarespace.com/threat-profiles/ics-malware-variants/triton..
[31] OWASP. 2018. Top 10 IoT Vulnerabilities. Retrieved September 30, 2019 from https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project.
[32] Meltem Ozsoy, Caleb Donovick, Iakov Gorelik, Nael Abu-Ghazaleh, and Dmitry Ponomarev. 2015. Malware-aware processors: A framework for efficient online malware detection. In *Proceedings of the IEEE 21st International Symposium on High Performance Computer Architecture (HPCA'15)*. IEEE, Los Alamitos, CA, 651–661.
[33] Nisarg Patel, Avesta Sasan, and Houman Homayoun. 2017. Analyzing hardware based malware detectors. In *Proceedings of the 54th IEEE/ACM Design Automation Conference (DAC'17)*. ACM, New York, NY, 25.
[34] Diego Perez-Botero, Jakub Szefer, and Ruby B. Lee. 2013. Characterizing hypervisor vulnerabilities in cloud computing servers. In *Proceedings of the 2013 International Workshop on Security in Cloud Computing*. ACM, New York, NY, 3–10.

[35] Erik Puik, Daniel Telgen, Leo van Moergestel, and Darek Ceglarek. 2017. Assessment of reconfiguration schemes for reconfigurable manufacturing systems based on resources and lead time. *Robotics and Computer-Integrated Manufacturing* 43 (2017), 30–38.

[36] Davide Quarta, Marcello Pogliani, Mario Polino, Federico Maggi, Andrea Maria Zanchettin, and Stefano Zanero. 2017. An experimental security analysis of an industrial robot controller. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP'17)*. IEEE, Los Alamitos, CA, 268–286.

[37] Ahmad-Reza Sadeghi, Christian Wachsmann, and Michael Waidner. 2015. Security and privacy challenges in industrial Internet of Things. In *Proceedings of the 52nd IEEE/ACM Design Automation Conference (DAC'15)*. ACM, New York, NY, 54.

[38] Hossein Sayadi, Hosein Mohammadi Makrani, Onkar Randive, Sai Manoj P. D., Setareh Rafatirad, and Houman Homayoun. 2018. Customized machine learning-based hardware-assisted malware detection in embedded devices. In *Proceedings of the 17th IEEE International Conference on Trust, Security, and Privacy in Computing and Communications and the 12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE'18)*. IEEE, Los Alamitos, CA, 1685–1688.

[39] Hossein Sayadi, Nisarg Patel, Sai Manoj P. D., Avesta Sasan, Setareh Rafatirad, and Houman Homayoun. 2018. Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification. In *Proceedings of the 55th ACM/ESDA/IEEE Design Automation Conference (DAC'18)*. IEEE, Los Alamitos, CA, 1–6.

[40] Hamdija Sinanovic and Sasa Mrdovic. 2017. Analysis of Mirai malicious software. In *Proceedings of the International Conference on Software, Telecommunications, and Computer Networks*. 1–5.

[41] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. 2018. Industrial Internet of Things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics* 14, 11 (2018), 4724–4734.

[42] John A. Stankovic. 2014. Research directions for the Internet of Things. *IEEE Internet of Things Journal* 1, 1 (2014), 3–9.

[43] Andrea Tundis, Wojciech Mazurczyk, and Max Mühlhäuser. 2018. A review of network vulnerabilities scanning tools. In *Proceedings of the 13th International Conference on Availability, Reliability, and Security (ARES'18)*. ACM, Los Alamitos, CA, 1–10. DOI: https://doi.org/10.1145/3230833.3233287

[44] Emmanouil Vasilomanolakis, Jörg Daubert, Manisha Luthra, Vangelis Gazis, Alex Wiesmaier, and Panayotis Kikiras. 2015. On the security and privacy of Internet of Things architectures and systems. In *Proceedings of the International Workshop on Secure Internet of Things (SIoT'15)*. IEEE, Los Alamitos, CA, 49–57.

[45] Xueyang Wang and Ramesh Karri. 2015. Reusing hardware performance counters to detect and identify kernel control-flow modifying rootkits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 3 (2015), 485–498.

[46] Rolf H. Weber. 2010. Internet of Things—New security and privacy challenges. *Computer Law & Security Review* 26, 1 (2010), 23–30.

[47] Jacob Wurm, Khoa Hoang, Orlando Arias, Ahmad-Reza Sadeghi, and Yier Jin. 2016. Security analysis on consumer and Industrial IoT devices. In *Proceedings of the 21st Asia and South Pacific Design Automation Conference (ASP-DAC'16)*. IEEE, Los Alamitos, CA, 519–524.

[48] Liwei Zhou and Yiorgos Makris. 2016. Hardware-based workload forensics: Process reconstruction via TLB monitoring. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST'16)*. IEEE, Los Alamitos, CA, 167–172.

[49] Liwei Zhou and Yiorgos Makris. 2017. Hardware-based on-line intrusion detection via system call routine fingerprinting. In *Proceedings of the IEEE/ACM Design, Automation, and Test in Europe Conference and Exhibition (DATE'17)*. IEEE, Los Alamitos, CA, 1550–1555.

[50] Liwei Zhou and Yiorgos Makris. 2018. Hardware-assisted rootkit detection via on-line statistical fingerprinting of process execution. In *Proceedings of the IEEE/ACM Design, Automation, and Test in Europe Conference and Exhibition (DATE'18)*. IEEE, Los Alamitos, CA, 1580–1585.

[51] Wei Zhou, Yan Jia, Anni Peng, Yuqing Zhang, and Peng Liu. 2018. The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved. *IEEE Internet of Things Journal* 6, 2 (2018), 1606–1616.

Received March 2019; revised October 2019; accepted January 2020