Architectural Security Weaknesses in Industrial Control Systems (ICS)

An Empirical Study based on Disclosed Software Vulnerabilities

Danielle Gonzalez, Fawaz Alhenaki, Mehdi Mirakhorli
Rochester Institute of Technology
Rochester, NY USA
{dng2551,faa5019,mxmvse}@rit.edu

***Abstract*—Industrial control systems (ICS) are systems used in critical infrastructures for supervisory control, data acquisition, and industrial automation. ICS systems have complex, component-based architectures with many different hardware, software, and human factors interacting in real time. Despite the importance of security concerns in industrial control systems, there has not been a comprehensive study that examined common security architectural weaknesses in this domain. Therefore, this paper presents the first in-depth analysis of 988 vulnerability advisory reports for Industrial Control Systems developed by 277 vendors. We performed a detailed analysis of the vulnerability reports to measure which *components* of ICS have been affected the most by known vulnerabilities, which *security tactics* were affected most often in ICS and what are the common *architectural security weaknesses* in these systems.**

Our key findings were: (1) Human-Machine Interfaces, SCADA configurations, and PLCs were the most affected components, (2) 62.86% of vulnerability disclosures in ICS had an architectural root cause, (3) the most common architectural weaknesses were "Improper Input Validation", followed by "Improper Neutralization of Input During Web Page Generation" and "Improper Authentication", and (4) most tactic-related vulnerabilities were related to the tactics "Validate Inputs", "Authenticate Actors" and "Authorize Actors".

I. Introduction

Industrial control systems (ICS) are computers that control any automation system used in industrial environments that include critical infrastructures. They allow operators to monitor and control industrial processes and support the day-to-day operations of manufacturing, oil and gas production, chemical processing, electrical power grids, transportation, pharmaceutical, and many other critical infrastructures [1]. Security is one of the key concerns in these systems and an attack can potentially adversely impact the nation's safety and economy [2]. Therefore, it is important to design these systems from ground-up to be secure. Further, since ICS often support critical infrastructure, they cannot easily be taken down for security updates or software patches [3]. Figure 1 shows a typical architecture of ICS.

Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTUs) connect to sensors and actuators through a Fieldbus to gather real-time information and provide operational control of the equipment. These controllers also communicate to a data acquisition server (SCADA) using various industrial communications protocols [3].

ICS relies on Human-Machine Interfaces (HMI) to configure the industrial plant, troubleshoot the operation and equipment, run, stop, install and update programs, and recover from failures. The day-to-day operational data is logged by the Data Historian component for further analysis and diagnosis. Each of these components has its own interfaces and can be the target of attacks that impact industrial equipment and operations. Ever since the Stuxnet attack on modern supervisory control and data acquisition (SCADA) and PLC systems, there have been many vulnerabilities discovered and reported for these systems. Recent studies [3], [4] have shown that the documented attacks to ICS infrastructures have exponentially increased from a few incidents to hundreds per year.

While there have been various ad-hock studies of vulnerabilities in these systems [5]–[12], or approaches on how to make them more secure [3], [13]–[15], there has not been a systematic work that examined common security architectural weaknesses in ICS. Therefore, in this paper we report the results on an in-depth empirical study that examined security architectural weaknesses in ICS.

This study took several months in which we collected and analyzed 988 security advisories that disclosed ICS vulnerabilities. These vulnerabilities were mined from the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) repositories. We used these reports to answer the following research questions:

- **RQ1:** *Which ICS Components Are Most Vulnerable?* We found that HMI, SCADA and PLC were among the most vulnerable components in ICS.
- **RQ2:** *What Percentages of ICS Vulnerabilities had an Architectural Root Cause?* We found that 62.86% of the disclosed vulnerabilities had an architectural root cause. This was mainly because the ICS products were designed without security in mind and with an assumption that they will be used within a protected network.
- **RQ3:** *What are the Most Common Architectural Weaknesses in ICS?* We found that "Improper Input Validation" (26.29%) is the most common architectural vulnerability type across ICS products. This weakness can be exploited to perform a denial of service attack on control software and devices.

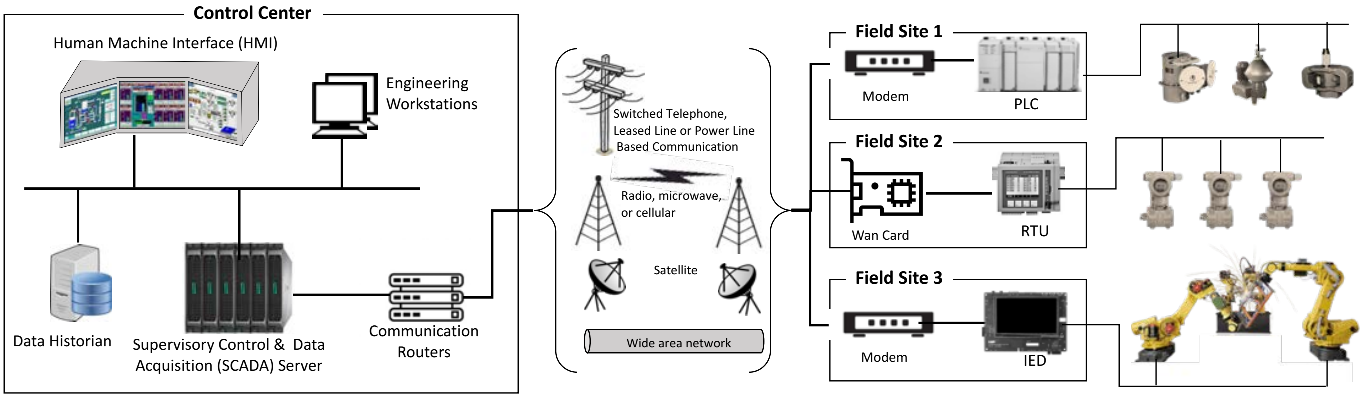
Control Center
Human Machine Interface (HMI)
Engineering Workstations
Data Historian
Supervisory Control & Data Acquisition (SCADA) Server
Communication Routers
Switched Telephone, Leased Line or Power Line Based Communication
Radio, microwave, or cellular
Satellite
Wide area network
Field Site 1
Modem
PLC
Field Site 2
Wan Card
RTU
Field Site 3
Modem
IED

Fig. 1. The figure describes the standard architecture of ICS, although implementations may vary.

- **RQ4:** *Which Architectural Tactic Implementations are Most Often Compromised in ICS?* We found the top 3 architectural security tactics compromised in ICS are Validate Inputs, Authenticate Actors, and Authorize Actors.

The **contribution** of our work is two-fold:

- An empirically grounded mapping of known vulnerabilities in ICS products to their impacted components and compromised security architectural tactics. This mapping allows us to highlight security *design* issues in ICS.
- An investigation on common architectural weaknesses in ICS systems, and their root causes.

This paper presents the *first* empirical study that includes the analysis of a large number of security advisories vulnerability reports with the aim of bringing attention to the architectural issues in ICS products.

The rest of this paper is organized as follows. Background information on Industrial Control Systems and their related architectural decisions is presented in Section II. In Section III we describe how the ICS advisory report data was mined, analyzed, and evaluated. We present our quantitative findings in Section IV, and in Section VI we interpret these findings and provide recommendations. Threats to the validity of our findings are acknowledged in Section VII and related work is described in Section VIII. We conclude in Section IX.

II. Background on Software Vulnerabilities

Vulnerabilities are security-related software defects that compromise a system's security requirements. ICS products rely on software programs to function and can become vulnerable as a result of such defects with a variety of consequences, such as leakage of data and the modification of data by unauthorized users. Software vulnerabilities are publicly disclosed and tracked in databases. Often each ICS vendor maintains advisories to report security issues within their ICS products. The **Industrial Control Systems Cyber Emergency Response Team (ICS-CERT)** compiles all these vulnerability reports and release them as soon as they are discovered. Vulnerability reports by ICS-CERT provide details about the vulnerability and are assigned an **Industrial Control Systems Advisory (ICSA)** identifier. The following excerpt from an ICS-CERT advisory report shows a vulnerability in one of the products developed by *Rockwell Co.* The overview section describes the vulnerability and impacted components of an ICS products. The **vulnerability characterization** (or **root cause**) indicates the cause of security issues. It refers to an entry from a list of known common software issues documented in the **Common Weakness Enumeration (CWE)** catalog [1].

Advisory (ICSA-16-336-06)
OVERVIEW: Alexey Osipov and Ilya Karpov of Positive Technologies have identified vulnerabilities in Rockwell Automations Allen-Bradley MicroLogix 1100 and 1400 programmable logic controller (PLC) systems. Rockwell Automation has produced new firmware versions to mitigate some of the vulnerabilities.
AFFECTED PRODUCTS:
1763-L16AWA, Series A and B, Version 14.000 and prior versions;
1763-L16BBB, Series A and B, Version 14.000 and prior versions; **[...]**
IMPACT:
Successful exploitation of these vulnerabilities may allow a remote attacker to gain unauthorized access to affected devices, as well as impact the availability of affected devices.
VULNERABILITY CHARACTERIZATION:
CLEARTEXT TRANSMISSION OF SENSITIVE INFORMATION (CWE-319)
User credentials are sent to the web server in clear text, which may allow an attacker to discover the credentials if they are able to observe traffic between the web browser and the server.
EXPLOITABILITY
These vulnerabilities could be exploited remotely.
[...]

A paper published by Santos et al. [16] revised the CWE catalog to differentiate the software weaknesses rooted in design versus those that are due coding issues such as buffer overflow. They presented the concept of **Common Architectural Weakness Enumeration (CAWE)**, a catalog which enumerates common types of vulnerabilities rooted in the architecture of a software and provides mitigation techniques to address them. The CAWE catalog organizes the architectural flaws according to known *security tactics*.

These weaknesses are are either due to an incorrect adoption of security tactics or violation of the tactic's key principle in the design or source code [17]–[19]. An example is "Exposure of Data Element to Wrong Session" [2] which occurs when a product does not sufficiently enforce boundaries between the states of different sessions, causing data to be provided to or used by the wrong session. Through this paper we rely on this catalog to reason about common architectural and

[1] https://cwe.mitre.org
[2] https://cwe.mitre.org/data/definitions/488.html

tactical issues in ICS systems. The CAWE catalog was adopted by MITRE Co and is used as the "Architectural Concept" view [3] on their CWE catalog [1]. This catalog can be used as a reference guide by developers/designers to reason about potential weaknesses in the design or implementation of their architecture [17], [20], [21]. Using the information from the ICS-CERT, the CWE/CAWE catalogs, and links between these sources of information, it is possible to study the root causes of tactical vulnerabilities in ICS.

III. Methodology and Study Design

To answer our research questions from Section I, we conducted an in-depth empirical study of ICS-related vulnerabilities based on guidelines for industrially-based multiple-case studies [22]. The unit of analysis in our study was a vulnerability advisory report for an ICS product.

A. Case Selection

We selected to study ICS products with *publicly disclosed* vulnerabilities. The U.S. National Cybersecurity and Communications Integration Center's (NCCIC) Industrial Control Systems unit collaborates with Cyber Emergency Response Teams (CERT)s to provide a central repository for vulnerability and exploit reporting specifically for Industrial Control Systems[4].

B. Data Collection

We automatically mined, processed, and analyzed text data from ICS-related vulnerability reports to extract information about *affected components* and *root causes*. This was achieved using a 5-step process combining automated data collection and processing with manual verification:

1) Mine ICS-related vulnerability advisory reports from the ICS-CERT reporting website (Section III-C).
2) Develop an automated detection system to extract impacted components from the text data in the advisory reports (Section III-D).
3) Manually verify the precision and accuracy of the automated detection (Section III-D).
4) Automatically extract the tactic-related weaknesses tagged for each advisory, and use this data to determine affected tactics (Section III-E).
5) Leverage the data from previous steps to determine components, tactics, and weaknesses with high frequency in ICS (Section III-F).

These steps are detailed in the following subsections. Each step relied on a broad range of data elements, so to help the reader understand how we utilized these data elements and artifacts, we present an information model in Figure 2. This information model shows each data element used in our study and the relationships between data elements and to various artifacts. The data elements on the top part are extracted from the reports through web-mining and natural language processing techniques, and then mapped to the data elements in the bottom part (CAWE).

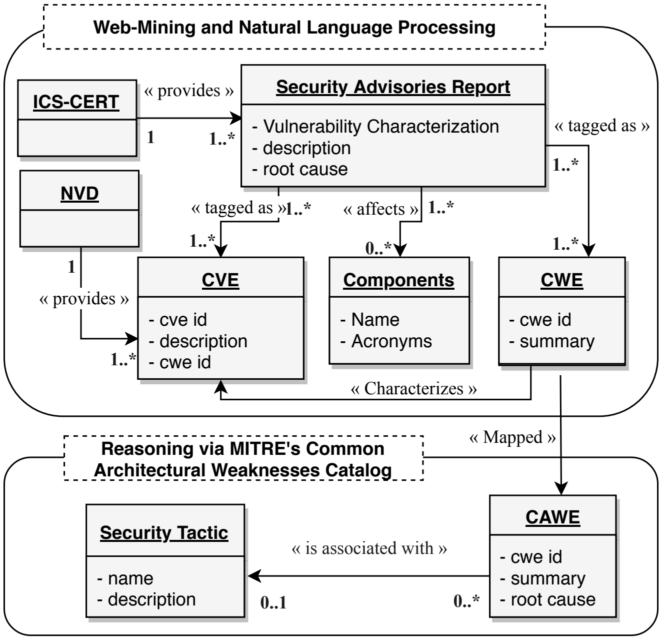

Fig. 2. Data Extraction Information Model

C. Vulnerability Report Mining

ICE related vulnerabilities are reported online. To conduct our empirical study we created a web-scraper to extract all text from each ICS-CERT report's web page and store it in a database. **988 advisory reports listed on the ICS-CERT advisory page were mined**. The dates of these reports range from March 10th, 2010 to September 24th, 2018 (the last date reports were mined).

D. Identification of Vulnerable ICS Components

Answering RQ1 required the identification of vulnerable ICS components in the advisory reports. We used an automated text analysis approach to analyze these reports and detect ICS components.

• **Step 1:** First, we developed a reference architecture for ICS products. To do so, we conducted an extensive manual search for common ICS components along with alternative terms or abbreviations for each component by reviewing online resources describing ICS systems, curating a list of components and terms. This included the review of architectural documents of ICS products that were accessible online, research papers [23], industrial products, and existing standards for ICS applications [2], [3], [7], [24]–[29]. We identified 17 components in this way. To avoid bias, each author of this paper reviewed the sources independently, and only terms with full agreement were kept.

• **Step 2:** We used the component-term data to create a *component dictionary*, with the proper name of each component as a *key*, and the list of alternative terms for each component as the *values*. Table II is a sample of the component/term dictionary. Furthermore, this list was cross-validated and revised by an engineer in a local ICS corporation. Table I provides descriptions of both "core" ICS components and "variation" components (found in *some* products).

[3] http://cwe.mitre.org/data/definitions/1008.html
[4] https://ics-cert.us-cert.gov/advisories

TABLE I
LIST OF ICS COMPONENTS USED IN THE STUDY

	Component	Specification
Common Components	Engineering workstation	Reliable software designed for configuration, maintenance and diagnostics of the control system applications and equipment.
	Data Historian (HIST)	A system for logging all data in an ICS environment, this data might be uses for analysis later.
	Human-machine Interface (HMI)	User interface that allows engineers and operators to interact with the controller.
	Intelligent Electronic Device	A smart industrial device capable of acquiring data, communicating with other devices, and automating industrial processes.
	Programmable Logic Controllers (PLC)	A solid-state control system often used in assembly lines, or robotic devices, or activities that require high reliability control, ease of programming and process fault diagnosis.
	RTU (Remote Terminal Unit)	Used to communicate with remote field equipment. PLCs with radio communication capabilities are also used in place of RTUs.
	Supervisory control software (SCADA)	A system performing control functions used to control dispersed assets using centralized data acquisition and supervisory control.
Variation	Common Industrial Protocol (CIP)	Provides a set of services & messages for control, security, synchronization, configuration, information that can be integrated into networks.
	Device Type Manager	A driver-like software that provides an interface for device configurations, maintenance, diagnostics & troubleshooting.
	Distributed Control Systems (DCS)	SACAD variant, microprocessor based units distributed functionally & geographically over the plant, situated near area where control or data gathering functions being performed.
	Distributed Network Protocol (DNP3)	a widely used protocol in electricity and/or water and waste water treatment plants with three layers (data link, application, and transport layer).
	Fieldbus	A digital, serial, multi-drop, two-way data bus between field equipment,. sensors, transducers, actuators, control room devices.
	Modbus	The de facto ICS communications protocol that uses serial communications with PLCs.
	Object Linking & Embedding (OLE) for Process Control (OPC)	A set of open standards developed to promote interoperability between disparate field devices, automation/control, and business systems.
	PAC	a "mashup" between a PC and a PLC in that it typically offers the benefits of both in a single package.
	Process Control System (PCS)	SCADA variant, typically rack-mounted, processes sensor input, executes control algorithms, and computes actuator outputs.
	Real time OS	Operating System

- **Step 3:** Next, we wrote a *keyword detector* script that used the component-term dictionary to automatically find terms in the text of each report. When a term was detected, it was logged with supporting information: the report ID, component name, and what section of the report it was found in (such as 'Background', 'Characterization', etc.) which was used to verify the accuracy of this approach (Section III-D). Our keyword detector was designed to identify the 'standalone' terms (those with white space on either side) as well as terms with non-character borders (those concatenated with as parentheses, numbers, or hyphens and etc.) We found that 768 of the 988 reports mentioned the 17 components.
- **Step 4:** The accuracy of our component detection technique was then manually verified. We checked for *false positives* by randomly selecting 50 reports that were marked as containing 1 or more components from our dataset and manually reviewing their text to confirm the match was correct. We also checked these 50 reports for any new components not found in our discovery step. We found no false positives or un-identified components.

TABLE II
SAMPLE OF COMPONENT-TERM DICTIONARY USED TO IDENTIFY COMPONENTS IN ICS ADVISORY REPORTS

ICS Component	Terms in Dictionary
Programmable Logic Controllers	Programmable Logic Controllers, PLC, control loop, logic controller
Distributed Control Systems	Distributed Control Systems, DCS, digital processor control system, process manager
Fieldbus	Fieldbus, Field Device
Supervisory Control Software	Supervisory control software, SCADA, supervisory control and data acquisition
Human-machine Interface	Human-machine Interface, HMI, Human Machine Interface, Web Interface, operator
Remote Terminal Unit	Remote Terminal Units, RTUs, RTU, Remote Terminal Unit
Data Historian	data historian, Operational Historian, HIST, process data archive, historian

E. Identifying ICS Vulnerability Reports with Architectural Root Cause

To answer research questions RQ2, RQ3, and RQ4, we needed to classify each ICS vulnerability as architectural and non-architectural. We used the CAWE catalog as a gold standard to classify vulnerabilities reported by ICS advisories (see Figure 2 for connection between vulnerability reports, CWEs, CAWEs and tactics).

- Since vulnerabilities (CVEs) reported by NVD typically contain a *CWE Tag* describing the type of software vulnerability, we could use these tags to automatically classify CVEs as tactical or non-tactical.
- In some cases the advisories reports did not have a CWE tag specified [5]. Thus, we used the *CVE tag* in the report and queried NVD database to find the CWE associated to the vulnerability. The combination of ICS-CERT and NVD databases help us to identify the missing CWE tags.

The steps described above were automated and the results have been vetted in a peer-evaluation process by the authors of the paper to ensure the mappings are 100% accurate.

To answer RQ2, RQ3 and RQ4, we used the mapping of 223 CAWEs to 12 tactics. We detected CAWEs related to 10 of the 12 tactics. Frequency data is reported in Section IV.

F. Data Analysis

RQ1 (Most Vulnerable ICS Components): To answer this question, we identified the most frequently reported components across 988 ICS-CERT Vulnerability Advisory Reports.
RQ2 (% of ICS Vulnerabilities with an Architectural Root Cause): This question was answered by calculating the percentage of ICSA reports that were tagged to a CWE ID mapped to the CAWE catalog of *architectural* weaknesses.
RQ3 (common architectural vulnerabilities in ICS): We identified the most frequently occurring CAWEs.
RQ4 (most compromised security tactics): we identified the underlying security tactic impacted by the most frequently occurring CAWEs.

[5] 129 reports did not have a CWE tag

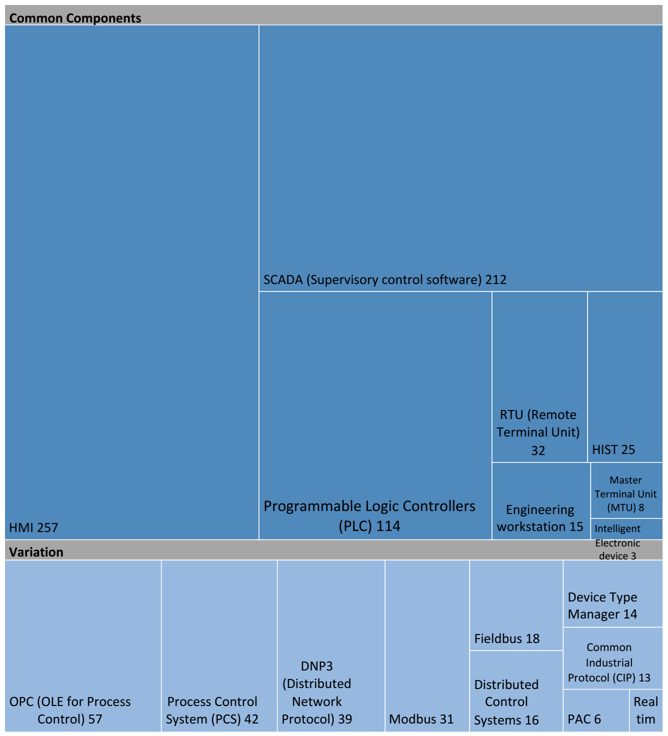

Fig. 3. Heat Map of Component Frequency in Advisories

IV. Results

In this section we present the results of the experiments described in Section III, organized by the research question they aimed to answer. We provide a combination of quantitative frequency data and qualitative studies that aim to look deeper into the most frequently found weaknesses, tactics and components.

A. RQ1: Which ICS Components are Most Vulnerable?

Our analysis detected 17 ICS components in 544 ICS advisory reports (55.06% of the entire corpus). The number of components mentioned in individual reports ranged from 1 to 6, but the majority of reports (331) mentioned only 1. Figure 3 is a heat map showing the most common components in these ICS advisory reports. The size of each block in the heatmap represents the *frequency* of a component, measured in number of reports it was detected in (Section III-D). The three most common components were Human-Machine Interfaces (**HMIs**), Supervisory Control And Data Acquisition (**SCADA**) Software, and Programmable Logic Controllers (**PLCs**). The remaining components were significantly less vulnerable. For instance, the fourth component impacted by vulnerabilities is OPC (reported 57 times) which is a software protocol based on Microsoft's Distributed Component Object Model (DCOM), used by manufacturers for interoperability of industrial processes in real time.

RQ1 Key Finding:

- Human-Machine Interfaces (HMIs), Supervisory Control And Data Acquisition (SCADA) Software, and Programmable Logic Controllers (PLCs) are the most vulnerable ICS components.

B. RQ2: What Percentages of ICS Vulnerabilities had an Architectural Root Cause?

From the 859 reports found to have CWE IDs tagged in them, 62.86% (540) were tagged with 1 or more *architectural weakness* (CAWE). The majority of these reports (378) only tagged 1 architectural CWE, but the remaining 162 reports tagged between 2 and 12 CWEs. This implies the importance of secure-by-design concepts in ICS systems.

RQ2 Key Finding:

- 62.86% of ICS vulnerability disclosures had an architectural root cause, while 37.14% of the vulnerabilities were due to coding defects.

C. RQ3: What are the Most Common Architectural Weaknesses in ICS?

Table III shows the top 20 architectural CWEs affecting ICS based on our analysis as described in Section III-E. These results show that *CWE 20: Improper Input Validation* is the architectural weakness affecting ICS most often, found in 142 reports. This is 26.29% of the 540 reports with architectural CWEs. This CWE was marked as a *tactical* weakness by [21] as it clearly affects the *Validate Inputs* security tactic.

RQ3 Key Findings:

- The most common architectural weakness affecting ICS is *CWE 20: Improper Input Validation* (26.29% of 540 reports with CAWEs). The other architectural weaknesses in the top-5 were *CWE 79: Cross-site Scripting*, *CWE 352: Cross-Site Request Forgery*, *CWE 89: SQL Injection*, and *CWE 287: Improper Authentication*.

D. RQ4: Which Architectural Tactic Implementations are Compromised Most Often in ICS?

To answer the previous question, we calculated the frequency of *architectural CWEs*, which were weaknesses affecting implementations of 12 architectural tactics. Each tactic is mapped to 1 or more CWE, so to measure frequency for each tactic we looked at reports which tagged its mapped CWEs and counted the *unique* number of times the tactic was tagged. This means, if a single report tagged more than one CWE mapped to the same tactic, the tactic was counted only once. As mentioned in the results for RQ3, there were 540 ICS advisory reports containing architectural CWEs. The number of *tactics* associated with each report based on tagged CWEs ranged from 1 to 5, but the majority (422) are associated with only 1. We found that CWEs mapped to 10 of the 12 tactics were found in the dataset. The top 3 **architectural security tactics** associated with ICS advisory reports are **Validate Inputs**,

TABLE III
TOP 20 ARCHITECTURAL WEAKNESSES (CAWES) IN ICS

Tactic	CWE	#Freq.
Validate Inputs	CWE-20 Improper Input Validation	142
Validate Inputs	CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	75
Authenticate Actors	CWE-287 Improper Authentication	70
Authorize Actors	CWE-284 Improper Access Control	63
Validate Inputs	CWE-352 Cross-Site Request Forgery (CSRF)	45
Authenticate Actors	CWE-798 Use of Hard-coded Credentials	44
Validate Inputs	CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	37
Authenticate Actors	CWE-259 Use of Hard-coded Password	20
Encrypt Data	CWE-522 Insufficiently Protected Credentials	18
Validate Inputs	CWE-94 Improper Control of Generation of Code ('Code Injection')	17
Authenticate Actors	CWE-306 Missing Authentication for Critical Function	16
Authorize Actors	CWE-434 Unrestricted Upload of File with Dangerous Type	14
Authorize Actors	CWE-269 Improper Privilege Management	13
Encrypt Data	CWE-326 Inadequate Encryption Strength	13
Encrypt Data	CWE-312 Cleartext Storage of Sensitive Information	12
Validate Inputs	CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	12
Authorize Actors	CWE-285 Improper Authorization	11
Encrypt Data	CWE-319 Cleartext Transmission of Sensitive Information	10
Encrypt Data	CWE-311 Missing Encryption of Sensitive Data	10
Encrypt Data	CWE-256 Unprotected Storage of Credentials	10

TABLE IV
10 COMPROMISED ARCHITECTURAL TACTICS

Tactic	# of Reports
Validate Inputs	351
Authenticate Actors	174
Authorize Actors	133
Encrypt Data	114
Limit Access	18
Identify Actors	13
Manage User Sessions	11
Cross Cutting	8
Verify Message Integrity	4
Audit	2

Encrypt Data, and **Authenticate Actors**. Frequency data for these 10 tactics is reported in Table IV.

RQ4 Key Finding:

- The top 3 architectural security tactics compromised in ICS are Validate Inputs, Authenticate Actors, and Authorize Actors.

E. Detailed Discussion of the ICS Common Weaknesses

Here we discuss the top 10 architectural weaknesses and examples of such vulnerabilities in ICS components.

1) CWE 20: Improper Input Validation: This weakness occurs when an ICS component either *does not* or *incorrectly* implements the *Validate Inputs* tactic. Our analysis indicates that this weakness can be present in every ICS component and the consequences often results a denial-of-service attack. For instance, ICSA-14-079-01 reports two vulnerabilities impacting Siemens SIMATIC S7-1200 Programmable Logic Controllers (PLCs). The crafted packets sent on 2 specific ports could initiate a special "defect mode" on the PLC device. The subsystem receiving the packets fails to validate, thus causing a vulnerability that can be exploited remotely and causes denial-of-service. These vulnerabilities were viable since the devices were accessible through the internet, however, they were mitigated by Siemens in new versions of these PLCs.

2) CWE 79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'): This weakness occurs in an ICS component that has a web accessible UI, but *does not* or *incorrectly* "neutralize" input provided by one user (commonly a web request) that is subsequently used to generate web pages which are served to additional users. With this flaw, attackers can inject client-side scripts into web pages viewed by other users. This weakness is related to the *Validate Inputs* tactic, however, it is limited to ICS components with web-accessible UIs. In contrast, with the previous weakness (CWE-20), attackers could implement a denial-of-service attack using a crafted input that did not necessarily involved web applications. ICSA-15-342-01 describes a cross-site scripting vulnerability affecting XZERES 442SR Wind Turbines. The web-based operating system of the turbine generator did not properly neutralize incoming web request content. This could be exploited remotely, and could cause the entire system to lose power.

3) CWE 287: Improper Authentication: A common design weakness discovered in ICS is that a component *fails to* or *incorrectly* verifies the identity claims of an actor interacting with it. For instance, ICSA-17-264-04 reports that iniNet Solutions GmbH's SCADA Webserver assumed the SCADA is used in a protected network and did not implement "Authenticate Actor" tactic at all. However, the system was connected to the Internet. In another case, ICSA-16-308-01 describes vulnerabilities in multiple versions and models of Moxa's OnCell Security software, which are cellular IP gateways that connect devices to cell networks. Access to a specific URL is not restricted, and any user may download log files when the URL is accessed. This can be exploited and used for an authentication bypass, where the malicious actor accesses the URL without prior authentication.

4) CWE 284: Improper Access Control: This is a weakness in *Authorize Actors* tactic that occurs when an ICS component *fails to* or *incorrectly* restricts an unauthorized actor from accessing resources or equipment. This weakness can result in privilege escalation, disclosure of confidential or sensitive data, or execution of malicious code. ICSA-13-100-01 discusses a case that MiCOM S1 Studio Software, which is used to configure parameters of electronic protective relays, fails to limit access to executables, meaning users without administrative privileges can replace the executables with malicious code or perform unauthorized modification of the relay parameters. The malicious actor could also exploit this to allow other users to escalate their own privileges. Exploiting this vulnerability requires physical access to the device and it can't be exploited remotely. The company provides mitigation strategies for users, but it has not been patched or fixed by adding appropriate security tactics.

5) CWE 352: Cross-Site Request Forgery (CSRF): ICS components may not verify that an expected and/or *valid* request to a web server was sent *intentionally* by the client (a.k.a. "forged" request). A malicious entity could take advantage of this weakness to force or trick a client into sending an unintentional request to an ICS component or equipment, which appears to be valid to the SCADA server or process controllers. ICSA-15-239-02 describes a cross-site request forgery (CSRF) vulnerability in the integrated web server of Siemens SIMATIC S7-1200 CPUs, which are used in Programmable Logic Controllers (PLCs). A user with an active session on these web servers could be tricked into sending a malicious request, which would be accepted by the CPU server without verifying intent. This can be exploited remotely, and would allow the malicious entity to perform actions to the PLC using the tricked user's permissions.

6) CWE 798: Use of Hard-coded Credentials: This occurs when ICS components stored *any* type of credential used for authentication, external communication, or encryption of data in readable formats (e.g. plain text) in discoverable locations. This weakness compromises an *Authenticate Actors* tactic in ICS, and enables malicious actors to implement authentication bypass exploits and make ICS components or equipment perform actions that require authentication or elevated privilege.

This weakness was found across all ICS components. For instance, ICSA-15-309-01 discloses a hard-coded SSH key in Advantech's EKI-122X Modbus gateways, which integrate Modbus/RTU and Modbus/ASCII devices to TCP/IP networked devices. The firmware of these gateways contains unmodifiable, hard-coded SSH keys. Since the keys cannot be changed but are discoverable, they could be remotely exploited to intercept communication by a malicious external actor.

7) CWE 89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'): This is another weakness in ICS that affects implementations of the *Validate Inputs* tactic. It describes an ICS component's *incorrect* or *failure to* "neutralize" user-provided inputs which are inserted as parameters to SQL queries, which can be exploited in SQL Injection attacks. In such cases the un-handled input is executed as *part of* the query itself instead of as a parameter. This can lead to the database unintentionally returning sensitive data or modification of the existing data. This weakness was mostly associated with components such as Human-machine Interface (HMI), Supervisory control software (SCADA) and Process Control System (PCS). ICSA-14-135-01 discusses a SQL injection vulnerability in several versions of the web-based CSWorks framework that is used to build process control software. The framework fails to sanitize or validate ("neutralize") user provided inputs which are intended to be used to read and write paths.

8) CWE 259: Use of Hard-coded Password: Hard-coded passwords are passwords stored in plain text or easily decryptable formats that are located somewhere discoverable by malicious actors, who can exploit them to bypass authentication or authorization checks. This weakness affects implementations of the *Authenticate Actors* tacic in ICS products.

For instance, ICSA-12-243-01 report describes a privilege escalation vulnerability in multiple versions of two varieties of GarrettCom Magnum MNS-6K Management Software, which is used for device management on managed Ethernet switches. An undocumented but discoverable hard-coded password could allow a malicious actor with access to a pre-existing account on a device to elevate their account privileges to admin levels. With these elevated privileges, a denial-of-service attack could be rendered or sensitive equipment settings could be changed.

9) CWE 522: Insufficiently Protected Credentials: This weakness occurred when the ICS components used an insecure method for transmission (e.g. insecure network) or storage (e.g. plain-text) of credentials. This insecurity can be exploited for malicious interception or discovery. This weakness affects implementations of the *Encrypt Data* tactic. ICSA-16-336-05B discusses unprotected credentials which could be exploited in multiple versions of 3GE Proficy Human Machine Interface (HMI), Supervisory Control and Data Aquisition (SCADA), and Data Historian products. In these products, if a malicious actor has access to an authenticated session, they may be able to retrieve user passwords not belonging to the authenticated account they are using. However, it cannot be exploited remotely.

10) CWE 94: Improper Control of Generation of Code ('Code Injection'): This weakness occurs when the ICS components *fails to* or *incorrectly* "neutralize" user-provided input to remove code syntax. This can result in this code being executed when the input is used to generate a code segment in the software. This weakness affects the *Validate Inputs* tactic.

ICSA-14-198-01 discusses a code injection vulnerability affecting some versions of Cogent Real-Time Systems, Inc's DataHub, a middleware used to interface with various control systems. If a malicious actor has access to and creates a 'Gamma' script on the local file system, the can craft a specially formatted user name and password to perform a code injection attack via an ASP page to execute that script file.

V. Comparing CAWEs In ICS Products and Enterprise Web Applications

We also compared the architectural weaknesses (CAWEs) we found to strongly affect ICS components with CAWEs previously identified as high-risk to commercial/enterprise web applications. The goal of this analysis was to determine if ICS components are most affected by unique weaknesses or if there was a similar pattern of common risks given that many

ICS products have web-based interfaces (such as HMIs and SCADA software). To do so, we compared our ranking (by frequency) of vulnerable architectural weaknesses in ICS to architectural weaknesses mapped to the 2017 OWASP Top 10 Web Application Security Risks [6], as shown in Table V. The OWASP Top 1ist is a report curated by the Open Web Application Security Project (OWASP) that enumerates which risks were most prevalent or impactful to web applications each year. We chose this list because it is widely-used when discussing web security and is based on empirical data and input from security experts.

TABLE V
COMPARISON WITH OWASP TOP 10

OWASP Top 10	ICS Architectural Weaknesses
A1-Injection	• **Rank#7: CWE-89 SQL Injection** • Rank#10: CWE-94 Code Injection • **Rank#16: CWE-78 OS Command Injection**
A2-Broken Authentication	• **Rank#3: CWE-287 Improper Authentication** • **Rank#9: CWE-522 Insufficiently Protected Credentials** • Rank#11: CWE-306 Missing Authentication of Critical Function • **Rank#20: CWE-256 Unprotected Storage of Credentials** • Rank#6: CWE-798 Use of Hard-coded Credentials • Rank#8: CWE-259 Use of Hard-coded Password
A3-Sensitive Data Exposure	• **Rank#15: CWE-312 Cleartext Storage of Sensitive Info.** • **Rank#18: CWE-319 Cleartext Transmission of Sensitive Info.** • **Rank#19: CWE-311 Missing Encryption of Sensitive Data**
A4-XML External Entities (XXE)	N/A
A5-Broken Access Control	• **Rank#4: CWE-284 Improper Access Control** • Rank#13: CWE-269 Improper Privilege Management • **Rank#17: CWE-285 Improper Authorization**
A6-Security Misconfiguration	N/A
A7-Cross-Site Scripting (XSS)	• **Rank#2: CWE-79 Cross-site Scripting**
A8-Insecure Deserialization	N/A
A9-Using Vulnerable Components	N/A
A10-Insufficient Logging	N/A

To create Table V, we first cross-referenced the MITRE CWE view that provides a mapping of CWEs to the 2017 Top 10 OWASP risks [7] with our top-20 architectural CWEs affecting ICS products (Table III). Each weakness on the right is marked with its ranking (by frequency) in our ICS dataset. Bolded weaknesses were included in the MITRE CWE-OWASP Map, and the non-bold weaknesses were added by us. To accurately map additional CAWEs to the OWASP risks, we carefully reviewed the OWASP Risk descriptions and the CWE description for each of the CAWEs from our top-20 which were not already included in the MITRE CWE-OWASP Map. Based on this, 4 of our top-20 most common CAWEs in ICS could not be appropriately mapped to an OWASP risk. This included our #1 most-frequent CAWE (*CWE-20: Improper Input Validation*). This weakness, although related to the *Validate Inputs* security tactic, is different than *A1-Injection* risk. Based on the description provided by OWASP, this risk focused on injection attacks such as code or SQL injections which are common in web-application. We randomly sampled 70 ICS vulnerability reports that tagged CWE-20 and found that this weakness in ICS devices targets availability of the devices and if exploited will result in denial-of-service attacks. The other CAWEs we were unable to map to OWASP Risks for similar reasons are *(#5) CWE-352: Cross Site Request Forgery*, *(#12) CWE-434: Unrestricted Upload of File with Dangerous Type*, *(#14) CWE-326: Inadequate Encryption Strength.*

The remaining 16 weaknesses were mapped to OWASP top 10 risks. Analysis of ICS vulnerability reports indicate the there are many web-based components in an ICS (e.g HMI) and these components are vulnerable to OWASP top 10 risk. Furhtermore, some of these components are critical to the operation of an ICS. If attackers can gain access of the HMI, they can often control the entire system.

Research Summary:

- Many modern HMIs are now web-based and there are cloud-based SCADA systems, therefore common web vulnerabilities affect these components of ICS.
- The most common architectural weaknesses in ICS are different than for web applications. ICS are most vulnerable to *CWE-20 Improper Input Validation* that can result in denial of service attack, crashing ICS process and equipment.
- ICS components are not secured-by-design. Many vendors have indicated that their products are designed to be used in a protected environment and therefore do not have mitigation techniques (e.g. authentication and input validation tactics) required to work in an untrusted environment.
- Industrial control systems rely on many vendors for PLC, RTU, IED and other controllers and equipment. This increases their attack surface and makes enforcing *Input Validation* tactic difficult.

VI. IMPLICATIONS FOR PRACTITIONERS

In this section, we provide a set of recommendations for ICS developers based on our findings in Section IV:

Generalizability and applicability of findings: As argued by Wieringa, describing the context and characteristics of the studied cases as in Table I allows us to "generalize" our findings by analogy (i.e., our findings may apply to systems that are similar to the cases of this study or have similar components) [30]. We also discuss generalizability in Section VII when we acknowledge validity threats.

Based on the findings from the case study, as well as the mitigation techniques reported by ICS-CERT in each case, we follow an inductive approach [30] and infer several recommendations:

• **Isolate control system devices and/or systems from untrusted networks** To increase security of ICS, advisories recommended minimizing network exposure for all control system devices and/or systems, and ensure that they are not

[6] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project

[7] https://cwe.mitre.org/data/slices/1026.html

accessible from the Internet or not connected to the business network. This can be also accomplished by putting remote devices behind firewalls, and isolating them from the business network. Other recommendations include: when remote access is required, secure methods must be used, and all the unused ports must be closed.

- **Sanitize the user's inputs to ICS components and devices** To mitigate CWE-20: Improper Input Validation, all the inputs to ICS endpoints [8] must be sanitized. For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side.
- **Encrypt Sensitive Data** Insufficiently protected credentials can enable attackers to gain access to the system. Furthermore, if the communicated data is not encrypted, the system is vulnerable to the *Authentication Bypass by Capture-replay*. An attacker can sniff the wireless transmissions between the SCADA, remote controller and the equipment and then bypass authentication by replaying it to the remote controller or equipment.
- **Secure ICS endpoints** Many equipment can be accessible through their designated port. If attackers have access to the ports they can implement various forms of attacks. Access to the endpoints outside the SCADA/HMI functions must be limited. Otherwise, users' access to the endpoints (e.g. ports) must be authenticated, and the users activity must be authorized. In case of HMI and Engineers Workstation, access to each URL must be authenticated to prevent authentication bypass weaknesses.
- **Follow tactic-centric approach to ICS security** By knowing the type of tactical vulnerability that can impact the security architecture of ICS, we can plan targeted testing and assurance strategies to mitigate security risks in these systems.
- **Usage of unpatched Devices** Since ICS components are often deployed in an industrial field in operation, patching vulnerabilities is an ongoing challenge. A common recommendation for usage of unpatched devices is to ensure the affected ICS components is running in a protected network environment.

VII. Threats to Validity

In this section we discuss threats present in our methodology and findings that could compromise 3 types of validity [31].

Construct Validity is a measure of how well the design of experiments reflects the research questions they aim to answer. In our study, we used vulnerability advisory reports released by the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) to measure the frequency, in number of reports, of common ICS components, architectural weaknesses, and security tactics in order to draw conclusions on the *most vulnerable* components and the architectural weaknesses that caused these vulnerabilities. Therefore, we rely on the accuracy of these reports to draw our conclusions and it is possible some relevant weaknesses or components were not found in these reports.

Internal Validity is the measure of how bias is minimized in the study. In order to identify *components* of ICS systems in these reports, we reviewed online literature about the ICS domain to manually compile a dictionary of components and their alternative names/phrasings. We acknowledge that there is a possibility of a *selection bias*, and components or phrasings could have been missed. To mitigate this, all authors of this paper reviewed the material independently, and only components/terms with full agreement were added to the dictionary.

External Validity is the measure of how generalizable the findings of a study are to the study's overall domain. In this work we aimed to draw conclusions about architectural vulnerabilities in the ICS domain. To do so, we used the vulnerability advisory reports published by ICS-CERT. Our dataset was comprised of 988 reports disclosed by 277 vendors of ICS products. While we acknowledge that this does not represent *all* products or vendors in this domain, we believe the dataset is large enough and covers a wide enough timeline (2010-2018) to be generalizable.

VIII. Related Work

Existing research in software architecture for security has mainly focused on secure design concepts, the analysis and evaluation of the existing security architecture [32], [33] as well as identifying potential vulnerabilities from the architecture [34], [35] . While these works discuss focus on overlap of security and architecture, they do not in particular study the architectural weaknesses in ICS. On the other hand, there have been research efforts that have looked into security aspects of ICS. Empirical studies in this domain has been limited to examining ad-hock security incidents relating to SCADA and critical infrastructure [9]–[12]. Other researchers have developed techniques for mitigating security design issues [3], [13], [14], such as embedding security into industrial IoT systems from early on using model driven approaches and tools [15]. In addition, past research has performed case studies of incidents rooting from a specific security issue, such as the impact of Stuxnet Worm on ICS [5]. There has been other research work focused on maliciously exploiting a particular ICS component without resorting to vulnerabilities or bugs [6]. Unlike previous research efforts, our work performs a comprehensive review of all US-CERT reported incidents, to investigate common security architectural weaknesses in ICS.

IX. Conclusion

This paper has presented a first-of-its-kind empirical study towards understanding architectural vulnerabilities in industrial control systems. We have examined 988 vulnerability advisory reports released by the Industrial Control System Cyber Emergency Response Team (ICS-CERT) regarding ICS products from 277 vendors. Our approach used a manually compiled dictionary of ICS components, CWE data from MITRE, and a mapping of CWEs to architectural tactics [17] to determine what the most-frequently compromised ICS components were and the architectural weaknesses which made

[8] endpoint: is the application's interface (e.g. URL or port) where the services can be accessed by a client application or user

them vulnerable. Our key findings were: (1) Human-Machine Interfaces, SCADA configurations, and PLCs were the most vulnerable components, (2) 62.86% of vulnerability disclosures in ICS had an architectural root cause, (3) most tactic-related vulnerabilities were related to the tactics "Validate Inputs", "Authenticate Actors" and "Authorize Actors", and (4) the most common CAWEs were related to "Improper Input Validation", followed by "Improper Neutralization of Input During Web Page Generation" and "Improper Authentication".

Acknowledgments

This work was partially funded by the US National Science Foundation under grant numbers CNS-1823246, CNS-1816845 and IIP-0968959 under funding from the S2ERC I/UCRC program and US Department of Homeland Security.

References

[1] S. Gold, "The scada challenge: securing critical infrastructure," *Network Security*, vol. 2009, no. 8, pp. 18 – 20, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1353485809700789

[2] D. of Homeland Security, "Department of homeland security (u.s.). critical infrastructure and control systems security curriculum," March 2018.

[3] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ics) security," *NIST special publication*, vol. 800, no. 82, pp. 16–16, 2011.

[4] W. Yang and Q. Zhao, "Cyber security issues of critical components for industrial control system," in *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, Aug 2014.

[5] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2011, pp. 4490–4494.

[6] S. A. Milinković and L. R. Lazić, "Industrial plc security issues," in *Telecommunications Forum (TELFOR), 2012 20th*. IEEE, 2012, pp. 1536–1539.

[7] K. KOBARA, "Cyber physical security for industrial control systems and iot," *IEICE Transactions on Information and Systems*, vol. E99.D, no. 4, pp. 787–795, 2016.

[8] A. Shahzad, S. Musa, A. Aborujilah, and M. Irfan, "Industrial control systems (icss) vulnerabilities analysis and scada security enhancement using testbed encryption," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*. ACM, 2014, p. 7.

[9] B. Miller and D. Rowe, "A survey scada of and critical infrastructure incidents," in *Proceedings of the 1st Annual Conference on Research in Information Technology*, ser. RIIT '12. New York, NY, USA: ACM, 2012, pp. 51–56. [Online]. Available: http://doi.acm.org/10.1145/2380790.2380805

[10] J. Stamp, J. Dillinger, W. Young, and J. DePoy, "Common vulnerabilities in critical infrastructure control systems," *SAND2003-1772C. Sandia National Laboratories*, 2003.

[11] R. Tsang, "Cyberthreats, vulnerabilities and attacks on scada networks," *University of California, Berkeley, Working Paper, http://gspp. berkeley. edu/iths/Tsang_SCADA% 20Attacks. pdf (as of Dec. 28, 2011)*, 2010.

[12] R. J. Turk *et al.*, *Cyber incidents involving control systems*. Citeseer, 2005.

[13] E. D. Knapp and J. T. Langill, *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems*. Syngress, 2014.

[14] L. Obregon, "Secure architecture for industrial control systems," *SANS Institute InfoSec Reading Room*, 2015.

[15] F. Ciccozzi, I. Crnkovic, D. D. Ruscio, I. Malavolta, P. Pelliccione, and R. Spalazzese, "Model-driven engineering for mission-critical iot systems," *IEEE Software*, vol. 34, no. 1, pp. 46–53, Jan 2017.

[16] J. C. Santos, A. Peruma, M. Mirakhorli, M. Galster, J. V. Vidal, and A. Sejfia, "Understanding software vulnerabilities related to architectural security tactics: An empirical investigation of chromium, php and thunderbird," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, pp. 69–78.

[17] J. C. S. Santos, K. Tarrit, and M. Mirakhorli, "A catalog of security architecture weaknesses," in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, April 2017, pp. 220–223.

[18] M. Mirakhorli and J. Cleland-Huang, "Transforming trace information in architectural documents into re-usable and effective traceability links," in *Proceedings of the 6th International Workshop on SHAring and Reusing Architectural Knowledge*, ser. SHARK '11. New York, NY, USA: ACM, 2011, pp. 45–52. [Online]. Available: http://doi.acm.org/10.1145/1988676.1988685

[19] M. Mirakhorli, Y. Shin, J. Cleland-Huang, and M. Cinar, "A tactic-centric approach for automating traceability of quality concerns," in *2012 34th International Conference on Software Engineering (ICSE)*, June 2012, pp. 639–649.

[20] C. Izurieta, D. Rice, K. Kimball, and T. Valentien, "A position study to investigate technical debt associated with security weaknesses," in *Proceedings of the 2018 International Conference on Technical Debt*, ser. TechDebt '18. New York, NY, USA: ACM, 2018, pp. 138–142.

[21] J. C. Santos, K. Tarrit, A. Sejfia, M. Mirakhorli, and M. Galster, "An empirical study of tactical vulnerabilities," *Journal of Systems and Software*, vol. 149, pp. 263 – 284, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121218302322

[22] J. M. Verner, J. Sampson, V. Tosic, N. A. Bakar, and B. A. Kitchenham, "Guidelines for industrially-based multiple case studies in software engineering," in *Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on*. IEEE, 2009, pp. 313–324.

[23] D. Sullivan, E. Luiijf, and E. J. M. Colbert, *Components of Industrial Control Systems*. Cham: Springer International Publishing, 2016, pp. 15–28.

[24] T. Agarwal, "Introduction to scada systems architecture, functionality and applications," Last Accessed Dec. 2018. [Online]. Available: http://www.efxkits.com/blog/scada-systems-architecture-functionality-applications/

[25] J. Foreman, M. Turner, and K. Perusich, "Educational modules in industrial control systems for critical infrastructure cyber security," *ASEE Annual Conference and Exposition, Conference Proceedings*, vol. 122, 01 2015.

[26] D. G. Peterson, "Digital bond, inc.: Quickdraw scada ids," 2014. [Online]. Available: http://www.digitalbond.com/tools/quickdraw/

[27] J. Rubio-Hernan, J. Rodolfo-Mejias, and J. Garcia-Alfaro, "Security of cyber-physical systems," in *Security of Industrial Control Systems and Cyber-Physical Systems*, N. Cuppens-Boulahia, C. Lambrinoudakis, F. Cuppens, and S. Katsikas, Eds. Cham: Springer International Publishing, 2017, pp. 3–18.

[28] P. Van Vliet, M.-T. Kechadi, and N.-A. Le-Khac, "Forensics in industrial control system: A case study," in *Security of Industrial Control Systems and Cyber Physical Systems*, A. Bécue, N. Cuppens-Boulahia, F. Cuppens, S. Katsikas, and C. Lambrinoudakis, Eds. Cham: Springer International Publishing, 2016, pp. 147–156.

[29] E. Foo, M. Branagan, and T. Morris, "A proposed australian industrial control system security curriculum," in *2013 46th Hawaii International Conference on System Sciences*, Jan 2013, pp. 1754–1762.

[30] R. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014.

[31] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[32] S. T. Halkidis, N. Tsantalis, A. Chatzigeorgiou, and G. Stephanides, "Architectural risk analysis of software systems based on security patterns," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 3, pp. 129–142, 2008.

[33] J. Ryoo, R. Kazman, and P. Anand, "Architectural analysis for security," *IEEE Security & Privacy*, no. 6, pp. 52–59, 2015.

[34] B. J. Berger, K. Sohr, and R. Koschke, "Extracting and analyzing the implemented security architecture of business applications," in *17th European Conference on Software Maintenance and Reengineering (CSMR)*. IEEE, March 2013, pp. 285–294.

[35] S. Al-Azzani and R. Bahsoon, "Secarch: Architecture-level evaluation and testing for security," in *2012 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*. IEEE, 2012, pp. 51–60.