National College of Ireland

Hybrid Security For Securing A Combination Of Physical And Virtual Information Assets

MSc Research Project
Cyber Security

Shiva Ramasamy
Student ID: X20135530

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shiva Ramasamy
Student ID:	X20135530
Programme:	Cyber Security
Year:	2022
Module:	MSc Research Project
Supervisor:	Ross Spelman
Submission Due Date:	15/08/2022
Project Title:	Hybrid Security For Securing A Combination Of Physical And Virtual Information Assets
Word Count:	5813
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Shiva Ramasamy
Date:	14th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Hybrid Security For Securing A Combination Of Physical And Virtual Information Assets

Shiva Ramasamy
X20135530

Abstract

Hybrid security is a collaborative design in which parameters related to physical and virtual security are collectively analysed to detect multi-domain, multi-departmental, or multi-vertical risks. Hybrid security has been used extensively in smart grid security and can be deployed for small to medium scale manufacturers. This research designed, implemented, and tested a hybrid security system using spring boot framework and Insomnia API gateways. The data storage used was PostgreSQL. The system was tested for four risk scenarios related to collaborative risk analysis of eleven parameters sensed and consolidated in a JSON object at the API gateway. The rules engine was written in Java, which performed with 100% accuracy of risk level identification and 0% false positives. Practical realisation and future research of the designed system is very much feasible given the components chosen are commercially popular. However, the coding and networking will require significant scaling up.

1 Introduction

The modern computing paradigm of industrial control systems has changed significantly as compared to the older days of supervisory and distributed control systems (Boyes et al.; 2018), (Sadeghi et al.; 2015). The older systems of programmable logic controllers are now rapidly getting replaced by the new cyber-physical systems controlled by computers running control engineering software and tools (Sadeghi et al.; 2015). Cyber-physical systems (CPS) provide deep accessibility into the industrial processes thus enhancing the precision of operations ((Sharma et al.; 2018)). Industrial deployment of cyber-physical systems is commonly referred to as Industrial Internet of Things (IIoT) (an industrial design of the Internet of Things architecture under the newly evolving Industry 4.0 framework) (Malik et al.; 2021) (Mugarza et al.; 2020) (Sharma et al.; 2018).

The modern computing paradigm employing CPS/IIoT has transformed the industrial control systems through the innovations of digitalisation, pervasive (or cognitive) computing, ubiquitous computing, and cloud computing ((Malik et al.; 2021; Schumacher et al.; 2019)). This has a multi-layered architecture, as presented in Figure 1 below.

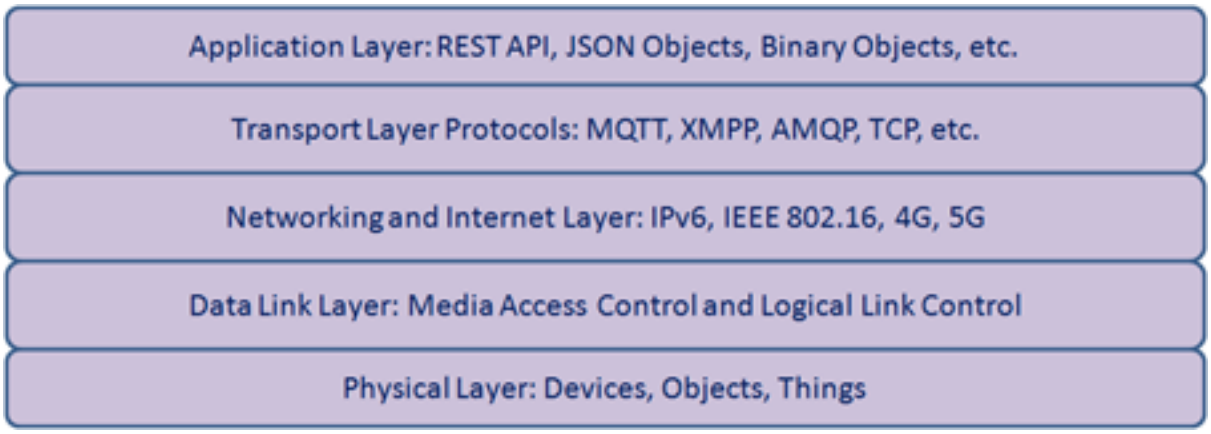

Figure 1: A multi-layered architecture of the modern computing paradigm of Industrial Control Systems.

There are five layers shown in Figure 1: Physical layer that comprises devices and things, data link layer facilitating devices networking in the physical space, networking and Internet layer providing IPv6 addressing and latest technologies for both low power and high power communications, transport layer providing the protocols for establishing sessions, and application layer where the application programming interface (REST), JSON objects, applications, and databases are deployed.

This research was conducted to meet the following aim and objectives:

Aim: To explore, study, design, and test a hybrid security system for industrial control systems using Industrial Internet of Things architecture.

Objective 1: To explore hybrid security designs in industrial control systems using Industrial Internet of Things architecture to capture the design specifications.
Objective 2: To implement an experimentation environment in a laptop following the design specifications and conduct coding and configurations as required to operate it as a fully configured pilot project.
Objective 3:To run test cases of different risk levels and determine whether the experimentation environment detects and logs the correct risk level based on its rules engine specifications.
Objective 4:To describe the practical validity of and future expansion of the hybrid security system.

The following research questions were addressed in this research: How a hybrid security system can be designed to protect industrial controllers and the devices administered by them using the Industrial Internet of Things architecture?

In the next section, the related works have been reviewed to build the theoretical knowledge required for this research.

2 Related Work

There are multiple research studies having studied the challenges of CPS security in IIoT systems. Their works may be broadly divided into three aspects: virtual (logical) security, physical security, and hybrid security. The reviews are presented in three sections as follows.

2.1 Virtual security

The virtual security controls of CPS are researched from the perspectives of authentication and authorisation, device recognition, anomaly detection, privacy-protected data access and transmission, communications security, and data security. This section presents some of the relevant studies presenting virtual security designs of CPS.

(Ning et al.; 2013)discussed a design of secure data access, data sharing, and authority transfer by device control systems configured to match the credentials between cyber entities and cyber targets. The device controllers are designed to refer databases for exchanges, handshakes, authority transfer comprising pre-shared secrets and privacy keys. Multiple handshake and authority transfer scenarios were discussed. Once the authority transfer is ascertained, the cyber entities are authorised to share data with the cyber targets through transport layer encrypted links. This design provided algorithmic interactions between the cyber entities and the device control systems before the formers are allowed to connect and transfer data with cyber targets. However, although the intent of this research was to present a working design of interactions security of CPS, the algorithms appeared as abstract conceptualisation that may require significant expanding before realisation through architecture design and software programming.

Similar studies were conducted by (Yousuf et al.; 2015; Maple; 2017; Rahman et al.; 2016; Roman et al.; 2013; Sadeghi et al.; 2015) . These studies are grouped together because they studied common controls for securing CPS interactions with their controllers. These studies have conducted theoretical reviews of the security of CPS interactions, which are useful in building a contextual design for this research. The key controls studied are CPS in-built (embedded) security architectures (critical code and local data security protocols and trust modules built within the CPS) ((Rahman et al.; 2016; Sadeghi et al.; 2015)), CPS integrity verification to protect against malicious software and hardware manipulations (Sadeghi et al.; 2015; **?**), secure device communications (Rahman et al.; 2016; Sadeghi et al.; 2015), secure localisation (Basri and Elkhadimi; 2020; Rahman et al.; 2016), device identification (Roman et al.; 2013; Vasilomanolakis et al.; 2015), device authentication and authorisation (Yousuf et al.; 2015; Maple; 2017; Rahman et al.; 2016; Roman et al.; 2013; Sadeghi et al.; 2015; Vasilomanolakis et al.; 2015), access control protocol and network security (Maple; 2017; Rahman et al.; 2016; Roman et al.; 2013; Sadeghi et al.; 2015)), trusted sensors with privacy protection ((Yousuf et al.; 2015; Maple; 2017; Roman et al.; 2013; Rahman et al.; 2016; Vasilomanolakis et al.; 2015)), fault tolerance (Roman et al.; 2013; Vasilomanolakis et al.; 2015), informed consent of controllers (Maple; 2017), and gateway for secure mediation (in federated CPS IIoT architectures) .(Yousuf et al.; 2015). These studies covered the context and application of the controls identified

There is a significant advantage in the studies of controls, as reviewed in the previous paragraph. They provide a strong context for designing a security framework with knowledge of rules and policies to be implemented. However, except for the study by (Rahman et al.; 2016; Yousuf et al.; 2015) no other study has presented details on how the controls are organised, positioned, and collaborated. This drawback does not help a security designer beyond establishing the basic design context. For example, the studies by (Roman et al.; 2013; Vasilomanolakis et al.; 2015) described device identification controls but did not explain how it needs to be integrated with device authentication and authorisation controls. Another example is about the device authentication and authorisation controls which are discussed by (Yousuf et al.; 2015; Maple; 2017; Rahman et al.; 2016; Roman et al.; 2013; Sadeghi et al.; 2015; Vasilomanolakis et al.; 2015), but except (Rahman et al.; 2016) no other research discussed about their positioning in the IIoT/CPS network architecture. Hence, advanced controls in collaborative and behavioural security are not covered in these studies.

Beyond establishment of basic context and content, the study by Rahman et al. (2016) is useful in designing the security organisation. This research mapped the security controls with the CPS/IIoT multi-layered architecture (Malik et al.; 2021) introduced in the introduction section. This mapping has been a useful reference in creating the multi-layered architecture designed for experimental study in this research. The architecture is replicated in the Figure 2 below:

Figure 2: A multi-layered security architecture of CPS/IIoT for the modern computing paradigm of Industrial Control Systems.

The above studies did not cover the hardware and operating system (low level) security aspects of CPS/IIoT. Hence, further studies were reviewed to expand the context. The research studies by (Ammar et al.; 2018; Frankó et al.; 2020; Happa et al.; 2019; Mugarza et al.; 2020; Plósz et al.; 2017) were found to be useful for this research. They are reviewed separately because they have covered different aspects of low level security controls in CPS/IIoT architecture and also the concept of collaborative security by analysing multiple data inputs.

The research by (Ammar et al.; 2018) was specifically identified because they reviewed

practical designs of IoT frameworks by several vendors: such as, ARM mbed IoT, AWS IoT, Azure IoT suite, Brillo/Weave, Calvin, and Smart Things. For every vendor they reviewed specifications of smartness, architecture, specifications of hardware, and security. They found common hardware specifications, such as custom microprocessor chips (low cost 32 bit controllers; both complete instruction set and reduced instruction set computing types), on-board random access memory, on-board storage, built in secured socket layer and transport layer security, authentication protocols (different types), SHA 256 hash function (keys exchange), on-board Wi-Fi client as well as access points (where device networking is part of the design), and on-board Bluetooth communication. The storage capacities on the IoT boards are quite low (32 MB to 128 MB). Hence, onboard security features are not sufficient. They need to be complemented by cloud backend controls. The authentication credentials and any subscription keys are normally stored within the device storage. Hence, device manipulation is possible. (Ammar et al.; 2018) provided useful insights into how vendors have implemented their IoT hardware and security. However, their research was too much vendor-specific that did not cover reflections of theoretical concepts.

The next research studied is by (Frankó et al.; 2020). They studied the mechanisms of tracking assets in a cyber-physical infrastructure. Their study comprised of architecture of RFID-based real-time localisation in an indoor positioning system, a communication network at Wi-Fi frequencies, and a front-end visualisation of the location trackers on a map of the facility. They designed the real-time localisation system with four wireless access points deployed at the corners of the facility. This is called a geo-fencing system and the tracking was done using motion-based sensors using the access points as referencing. They measured the overall accuracy of ranging and detection of start and stop events and elements crossing the digital fencing. This is a good research on localisation design although their experimental setup did not get into coding an application or creating a 3D model as a digital twin. They simply focused on accuracy of localisation. The other two studies on localisation conducted in this research are by (Basri and Elkhadimi; 2020; Braggaar; 2018). They described the design of secure triangulation method for secured localisation within a digital fencing.

The next research studied is by (Happa et al.; 2019). They studied collaborative security in a mixed reality business environment. Mixed reality may comprise of a mix of multiple business process verticals serviced by a common network infrastructure. As the business owners are focused on their respective business process verticals, exploiters can take advantage of inter-vertical ignorance causing lack of realisation of anomalies inducted by them. In industrial systems, the sensors may be feeding data to multiple business process verticals managed by different managers. This research showed how the sensor data of different verticals can be collaborated to detect and contain exploits. This research discussed four use cases in which, the use cases of industrial espionage and virtual trying were of special interest to this study. They explained how attack behaviours can be detected by collaborating data from multiple sensors. However, this research did not provide adequate technical details on how the collaboration process was realised. Another research reviewed with collaborative security context was by (Plósz et al.; 2017).

The next research studied was of (Mugarza et al.; 2020). They studied the security issues and solution in deploying security patches in the Industrial Internet of Things net-

working. They described the IEC 62443-2-3 patch management cycle and its application in IIoT networking. They described the framework of current patch states of all devices, patch authorisation statuses, and patching policy for immediate deployment and deployment in 30, 60, and 90 days. Their research covered the issue of patch management in IIoT. This study was shortlisted because patch management is a major security control but not studied adequately in IIoT networks. Their research was, however, theoretical only. They did not study and evaluate any practical implementation.

2.2 Hybrid security

Physical security is a very crucial aspect of CPS/IIoT security. However, physical security is not studied separately by the scholars as the total solution always combines physical and virtual security controls. The research studies reviewed in the previous section do not cover physical security aspects of CPS/IIoT networking (except for localisation covered by (Rahman et al.; 2016). This is because the studies were mostly focused on cloud-hosted controllers. In many industrial settings (especially large ones), controllers are deployed on premises (Kobara; 2016). Hence, physical security is a critical aspect. The physical space is divided into small spaces with varying criticality such that the intensity of controls can be implemented in controlled ways. There is a large body of research on smart grid security and collaborative security (some of them reviewed here). This section presents some hybrid security design reviews from these studies as they are relevant to the context of this research. Smart grid is one of the many industrial applications of CPS/IIoT. However, it is studied separately given that it is one of the largest adopters of CPS/IIoT and has a historical track record in tackling both physical and cyber security threats (Gunduz and Das; 2020; Lei et al.; 2018).

The research by (Neureiter et al.; 2016) explained the fundamental requirements of hybrid security in Smart Grid security models. Smart Grids are formed by thousands of assets forming a massive attack surface for the attackers. Hence, hybrid security model is valid for Smart Grid security. This fact was also explained by (Marksteiner et al.; 2019) in the context of low-voltage distribution architectures. Every asset monitored should be categorised under clearly defined asset classes and addition/removal of assets in each class and their exact locations should be recorded and approved in an assets governance system ((Marksteiner et al.; 2019; Neureiter et al.; 2016)). Pattern of anomalies in each asset should be clearly identified such that their detection and risk logging can be planned. Monitoring every asset may not be possible centrally, especially in low voltage distribution given that millions of them may be outside the core administration domain installed in user or near user premises ((Marksteiner et al.; 2019)). These research studies provided a good insight into the hybrid security in smart grid but covered only the requirement specifications.

Another research on hybrid security reviewed was by (Gunduz and Das; 2020). They described a number of smart grid attack scenarios and explained that the attacks mostly happen through vulnerabilities in human – machine interface, remote terminal units (used for remote administration of the controllers), machine terminal units, and programmable logic controllers. The attacks can happen physically as well as logically. They covered the key components of cyber security in smart grid. However, their research was limited to theoretical description only. An additional theoretical research in the same context

reviewed was by (Lei et al.; 2018). They described the categories of attacks as: interception, modification, interruption, fabrication, active attack, passive attack, insider attack, and outsider attack.

2.3 Research gaps

A clear gap in the research studies reviewed in the previous two sub-sections is about realisation and practical evaluation. The realisation part is avoided by most of the researchers as they focus greatly on theoretical concepts and some evidences taken from running systems. Creating a system requires several components. While the theoretical knowledge of the components is available amply from the existing literature, realising them practically is seldom covered. This gap invokes a need for exploring technical manuals and finding own ways without much and effective empirical guidance. This research has been conducted on actual practical realisation of the hybrid security model. However, except for the knowledge of the five layers of CPS/IIoT infrastructure, and the names of components needed (with some description), the research was largely dependent upon technical books and manuals.

Another research gap is lack of test cases and proper reporting of the test results. Research studies provide test results in the form of running graphs and sensitivity analysis charts. However, the test cases and testing procedure are not described. Further the practical validity of the test results are not covered adequately.

3 Methodology

In this section, the research environment and methodology is described. They are presented in brief as the full details are presented in the configuration manual. This research was conducted to test an experimental pilot of collaborative security using hybrid parameters relevant to physical and logical security in a manufacturing premise. The pilot was designed and configured in Ubuntu 20.0.4 operating system (deployed in a VMware virtual box configured inside Windows 10) and multiple software components installed in it. The components are listed and described in Figure 3.

Component	Description	Role
Open JDK 13	The main Java Development Kit	For providing packages to run the program
Maven and Spring Boot framework	Spring Boot framework on the top of the JDK 13 kit running on the Maven platform;	Actual Java coding done in the Spring Boot framework; Maven provides its runtime by downloading and installing the packages requested in a file called Pom.xml
Insomnia API gateway	Serving as API gateway to the API server developed using Spring Boot framework;	Collects data from sensors and transmits to the API server; in this project real sensors could not be used and hence, data was consolidated in a JSON file manually at the API gateway;
PostgreSQL	Database	For storing the data sent by the API Gateway;
PGAdmin4	Database Management User Interface	For viewing and managing data stored in PostgreSQL
LocationModel.Java	The data modelling code written in Spring Boot	This code defines the columns in the database and the data type and variables; This code serves as the main data construct reference of the pilot;
LocationRepository.Java	The Java Database Connectivity code written in Spring Boot that uses Hibernate (an internal database access layer in Spring Boot)	This code facilitates interactions between the API server in Spring Boot and the PostgreSQL database;
LocationController.Java	This is the main controller code having rules for collaborative security taking data from eleven sensors.	This code takes inputs from the API gateway, saves them to the database through the model and repository codes, and then uses a detailed rules engine to log alerts with risk levels.

Figure 3: Components and their descriptions with roles in the experimental setup.

Other options considered before selecting these components were JavaScript server using Express Node.js framework (; n.d.) and C++ backend in .Net core open source (*.NET Cloud Client Libraries*; 2022). These two frameworks could have been used, as well, for meeting the research objectives. Java in Spring Boot was chosen because of the researcher's comfort level with the language and also its significant popularity in the manufacturing industries for API server applications (Vermeer; n.d.). It may be noted that selecting a framework like Spring Boot does not mean that the core components can be avoided. Just like every framework, Spring Boot is built upon Java Development Kit,

which is a core Java foundation comprising the Java language specifications (Webb et al.; 2020; Gosling et al.; 2021). Currently, Spring Boot is supported on Java Development Kit version 13. Hence, it was the first environment configured. Other software installed with their description and their roles are shown in Figure 3.

The process followed for this research is described in the following steps:

Step 1: A Ubuntu virtual box was configured inside Windows 10.

Step 2: Java JDK version 13 was installed and its environment created.

Step 3: Software essential for this research were installed: Maven, Spring Boot (spring.io), Insomnia API Gateway, PostgreSQL, and PGAdmin4.

Step 4: The spring boot environment was configured using Pom.xml and application properties.

Step 5: Four Java codes were written: Main Class, Location Model, Location Repository, and Location Controller; the Location Controller is the main code comprising the rules engine for hybrid security; these four codes collectively formed the API server; the server interface was created at Localhost: 8081 using embedded Apache in Spring Boot.

Step 6: Four scenarios were created: Low risk level (only one parameter breached the rules), Medium risk level (two parameters breached the rules), High risk level (three parameters breached the rules), and Critical risk level (any number of parameters that breached the rules).

Step 7: The four scenarios were tested by sending four JSON objects to the API server having the breaches conducted in the values.

Step 8: The detection of the breaches by the rules engine was tested in the API gateway interface as well as in the Log file.

Step 9: Verification of the sensor data recording in the PostgreSQL database was also done.

4 Design Specification

The main aim of this research is to explore, study, design, and test a hybrid security system for industrial control systems using Industrial Internet of Things architecture. The literature review helped in exploring the layers of CPS/IIoT architecture. This research followed the knowledge of the CPS/IIoT architecture layers to create a design framework. The design framework was created taking help of the multilayered CPS/IIoT architecture models by (Malik et al.; 2021; Rahman et al.; 2016)(presented in Figures 1 and 2). A multilayered structure was created in the experimental setting of this research based on Figures 1 and 2. The description of the layers is presented as the following:

(a) Physical and data link layers: the physical layer comprises of physical sensors attached to devices and the data link layer comprises of the PHY interfaces of the devices having unique MAC addresses. In this research, setting up the physical and data link layer was not possible given the cost constraints. Hence, the physical layer is simulated by creating a JSON object file in which, the data collected from sensors are consolidated. A sample format of the JSON object used is presented below:

{ "xaxis" : "2",

"yaxis" : "2",
"zaxis" : "8",
"cpuFanSpeed" : "1200",
"cpuTemperature" : "80",
"gpuFanSpeed" : "1000",
"gpuTemperature" : "90",
"threeFailedAuthentication" : true,
"threeFailedCriticalFolderAccess" : false,
"criticalSoftwareInstallation" : false,
"criticalSecurityPatchFailed" : false
}

The variables entered in the JSON file of the API gateways are: X-location, Y-location, Z-location, CPU fan speed, CPU temperature, GPU fan speed, GPU temperature, whether three logins failed, whether critical folder access denied occurred three times, whether critical software installation failed, and whether critical security patch installation failed. In the practical world, this JSON file will be populated by the sensors attached to the physical devices being monitored by this system.

(b) Network layer: the network was configured inside the virtual box. The API server was hosted at http://127.0.0.1:8081 and the API gateway was point to the path http://localhost:8081/location. The local host is mapped by default to the IP address 127.0.0.1 in Ubuntu.

(c) Transport layer: The transport layer was configured using REST API calls to the API server. The calls POST and GET were made in sequence for all the four scenarios tested.

(d) Application layer: The application layer comprised of the Maven and Spring Boot framework, the Java codes written in Spring Boot, and the PostgreSQL database server managed using PGAdmin4.

5 Implementation

The final output of the implementation comprised of a runtime of Java Spring Boot having one main class and three sub-classes serving as the API server, and the runtime of Insomnia that served as the API gateway. This research simulated data collection from sensors using a JSON object compilation in which, values of eleven parameters were entered. The API server was programmed to assess the JSON object compilation sent to it through an API POST call.

The rules configured in the API server were the following. These rules were configured in the main Java program file of the API server named "LocationController.java". Given that this research implemented collaborative security as explained by (Happa et al.; 2019; Plósz et al.; 2017), the rules also comprised of syntax's for collaborating multiple variables (defined by using "&"). All the combinations of the eleven variables with decision-making on rules breaches were programmed in Java. Hence, the coding was completed in about

100 pages.

(a) X-location is changed by greater than 10 metres.
(b) Y-location is changed by greater than 10 metres.
(c) Z-location is changed by greater than 10 metres.
(d) CPU fan speed has increased beyond 1100 RPM.
(e) CPU temperature has increased beyond 80 degrees Celsius.
(f) GPU fan speed has increased beyond 1100 RPM.
(g) GPU temperature has increased beyond 80 degrees Celsius.
(h) Whether three logins have failed (Boolean – Yes or No).
(i) Whether critical folder access denied occurred three times (Boolean – Yes or No).
(j) Whether critical software installation failed (Boolean – Yes or No), and/or.
(k) Whether critical security patch installation failed (Boolean – Yes or No).

The LocationController.java was programmed to take four actions whenever an API POST call is received by it:

(a) Receive and parse the data consisting in the JSON object sent by the Insomna API gateway.

(b) Commit the data parsed in the appropriate columns of the table named "location" in the database named "shivaproject" in PostgreSQL; data committing in PostgreSQL was done with the help of two more Java code files: LocationRepository.Java and LocationModel.Java. These two Java files are explained in the manual document.

(c) Judge the data collectively based on the rules engine and send the judgment result (identification of the risk level and the action to be taken) to the API gateway.

(d) Log all the judgments in a text file named as "output.txt" for future analysis.

In this research, only one API gateway was used because it was a pilot experiment. In real implementation, there may be hundreds of such API gateways installed at the shop floors where the controllers and the IIoT attached CPS devices controlled by the controllers are installed. The local supervisors keep an eye on their respective API gateway monitor to get the risk alerts and take necessary action within the area controlled by them. The next section presents an evaluation of four scenarios tested in the experiment.

6 Evaluation

The final step of the experiment was to run the test cases defined. In this research, four scenarios were tested: risk level low, risk level medium, risk level high, and risk level critical. There were four experiments conducted. Their details are presented in the subsequent sub-sections.

6.1 Experiment 1

In this experiment, the JSON object transmitted to the API server did not have any breaches of rules as defined in the Location Controller Java code (discussed in previous section). As shown in Figure 4, the Java Controller clearly identified this combination of parameters as a Low risk level scenario. This test was repeated with multiple combinations and every time the Java Controller could identify the risk level accurately. This indicates that the controller has been written accurately to avoid false positives. The

decision was made in 32.8 Milliseconds. In the practical scenario, the supervisor of the monitoring area need not take any action.

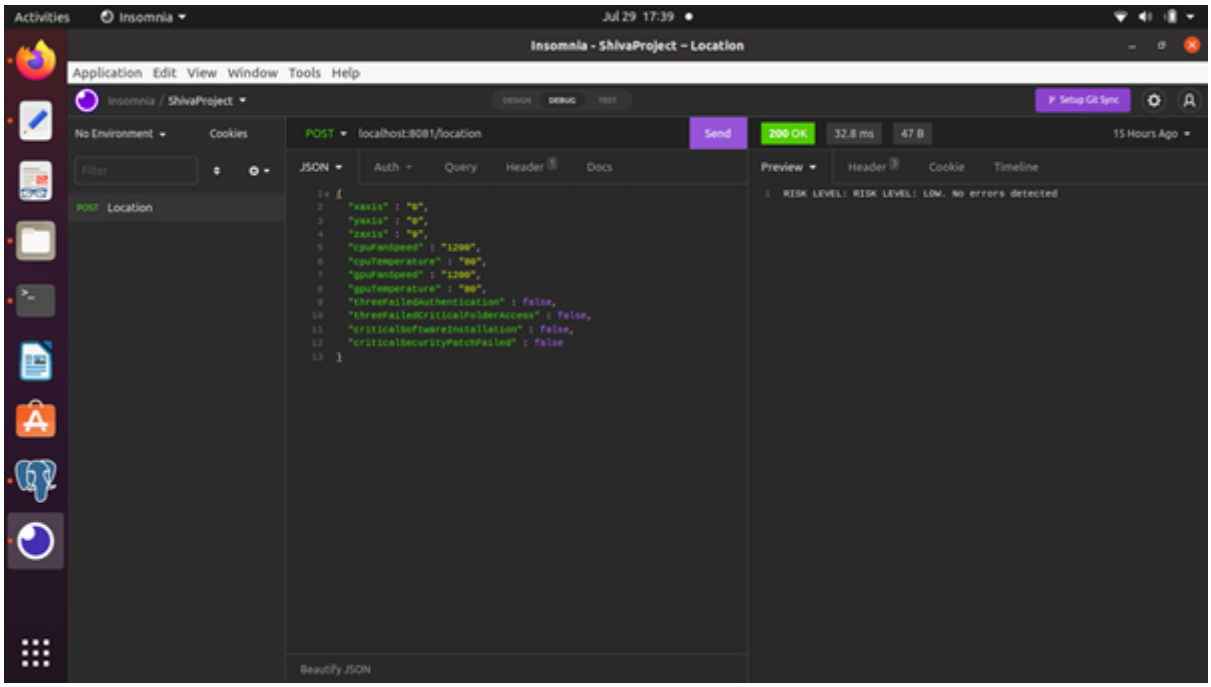

Figure 4: JSON object transmission with no parameter breaching the rules

6.2 Experiment 2

In experiment 2, one of the parameters was allowed to breach the rules. As shown in Figure 5, the breach was caused in the parameter “three failed attempts to access critical folder”. This is a Boolean. In the JSON object, it was marked as “true”. In the practical scenario, the supervisor controlling the CPS assets in the local area needs to check the folder logs and find out who was attempting to access the folder and why the access failed thrice.

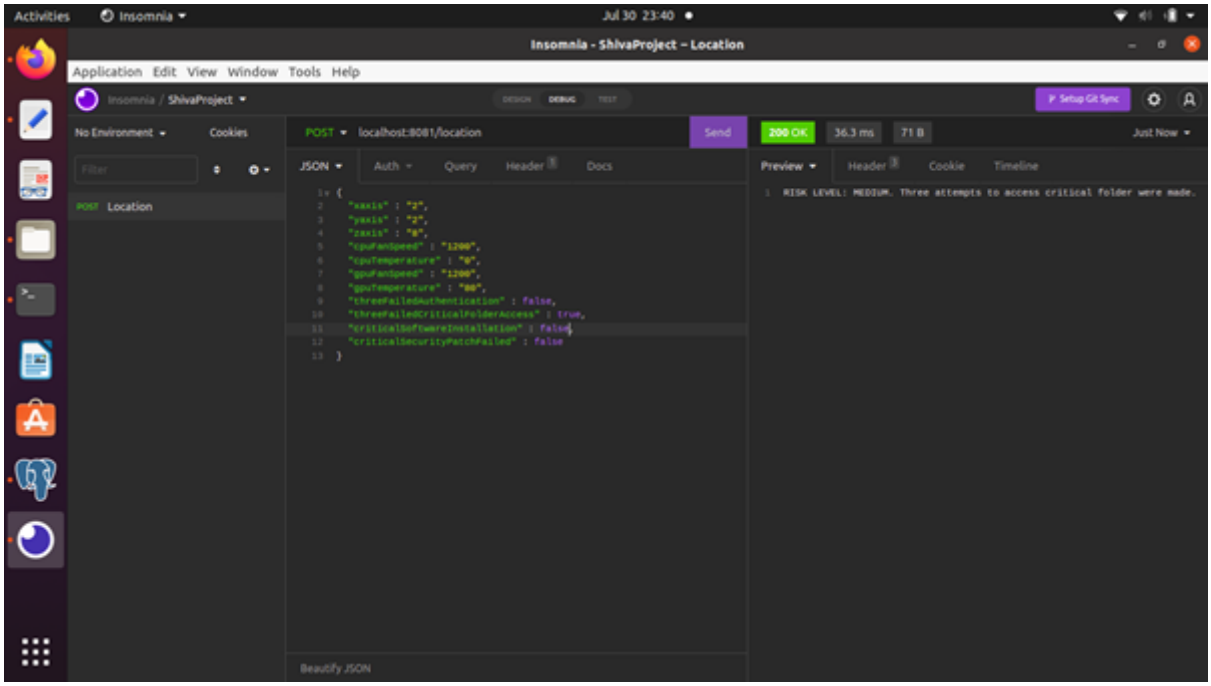

Figure 5: JSON object transmission with one parameter breaching the rules

When the JSON object was transmitted to the API server, the Location Controller rightly identified the risk scenario at medium level. The decision was made in 36.3 Milliseconds. This test was conducted with all the 11 parameters and the controller identified the risk level accurately. This indicates 100% performance of true positives and zero percent false positives.

6.3 Experiment 3

The third experiment was conducted by breaching any two parameters in the JSON object file. As shown in the screenshot in Figure 6, the parameters breached were "three failed attempts to access critical folder" and "critical software installation failed". As shown in the Figure, the Location Controller identified the risk scenario at High level. The decision was made in 43.3 Milliseconds. All possible pairs of breaches were tested in this experiment. The Location Controller identified the risk at "High" level in all the tests. This indicates 100% performance of true positives and zero percent false positives. In the practical scenario, the supervisor should check the logs to find out who has caused the breach and why there were failures. In this scenario, the core security team of the organisation may also get involved because a critical software update or patch installation had failed.

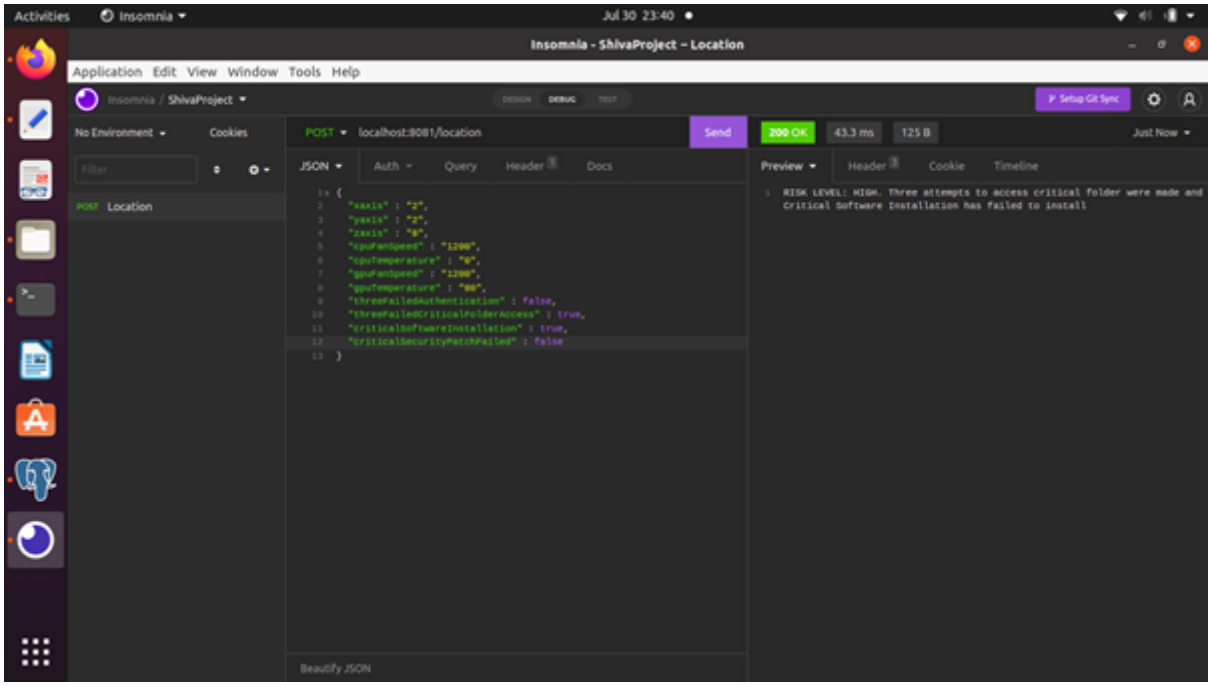

Figure 6: JSON object transmission with two parameters breaching the rules

6.4 Experiment 4

The fourth experiment was conducted by breaching three parameters in the JSON object. As shown in the Figure 7, the parameters "three failed attempts to access critical folder", "critical software installation has failed", and "security patch has failed to install" were marked as Boolean true. The Location Controller rightly identified the risk scenario at "Critical" level. The decision was made in 31.1 Milliseconds.

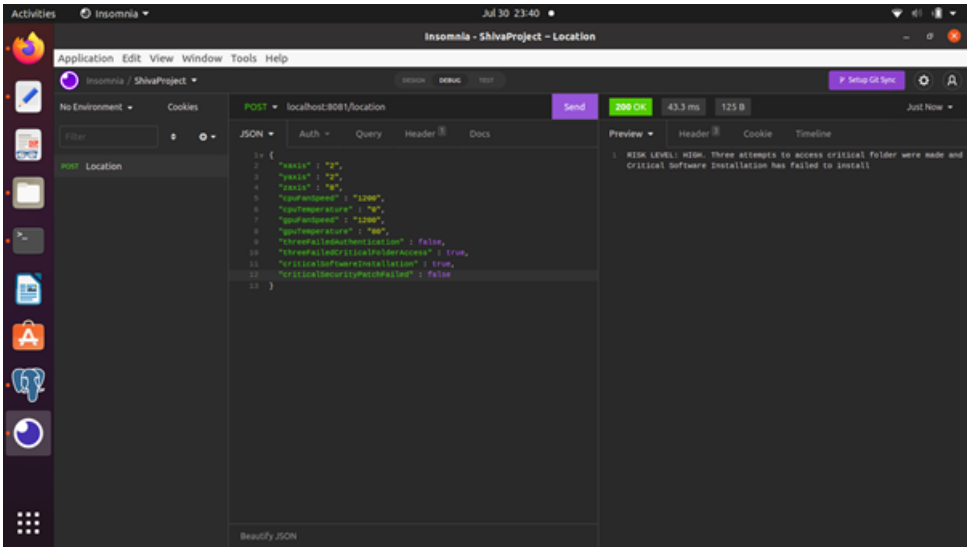

Figure 7: JSON object transmission with three parameters breaching the rules

This experiment was tricky. It is not possible to test all possible scenarios of three

or more breaches among eleven parameters. There will be a significant number of combinations. The rules engine code reached almost 100 pages in covering all the possible scenarios. However, the experiment was conducted with large number of combinations of three, four, and five parameters breaching the rules. In all the attempts, the Location Controller accurately identified the risk level as Critical. This indicates 100% performance of true positives and zero percent false positives. This is a critical situation and hence, the core security team should get directly involved in the investigations.

The text file is used in this research to log Alerts. In real world systems, the logs will be stored in a separate activity logging database, which will have its own analytics layer. By collaborating the information in multiple parameters, the exact threat scenario be judged. For example, if the GPU and CPU temperatures of a server have breached the limits and the server itself has been shifted by about 20 metres, the security administrator can judge that the server has been shifted from its air conditioning environment and is now facing heat. This may have happened because the air conditioning malfunctioned and a technician is on site to repair it. For plant continuity, the supervisor may have decided to shift the server outside the air conditioning cubical while the technician is carrying out the repairs but keeps the server running. There can be numerous such scenarios in a plant campus when this hybrid security system can be very useful.

In the next section, a discussion about the significance and performance of this design has been discussed.

6.5 Discussion

The test results provided useful insight into an automated risk assessment by a control engine, which can be logged for necessary action by the security operations. This capability is already realised in Smart Grid systems, and can be implemented at a lower scale in small to medium sized manufacturing organisation. A projected scenario of this design is presented in Figure 8 as the following.

The design presented in Figure 8 was realised at a small scale within Ubuntu laptop. In real world manufacturing, this design may be merely a module defining digital fencing of a set of assets. There may be hundreds of such modules active. All of them may be communicating with a centralised API server. There were eleven parameters configured to be monitored. In reality, there may be hundreds of them. Hence, detailed risk logs would have to be generated for advanced analytics. The screenshot in Figure 9 shows a snippet of the logging generated in this research. This will be needed at massive scales and a smart log analyser using machine learning may be needed to understand the ongoing and historical attack patterns in the organisation.

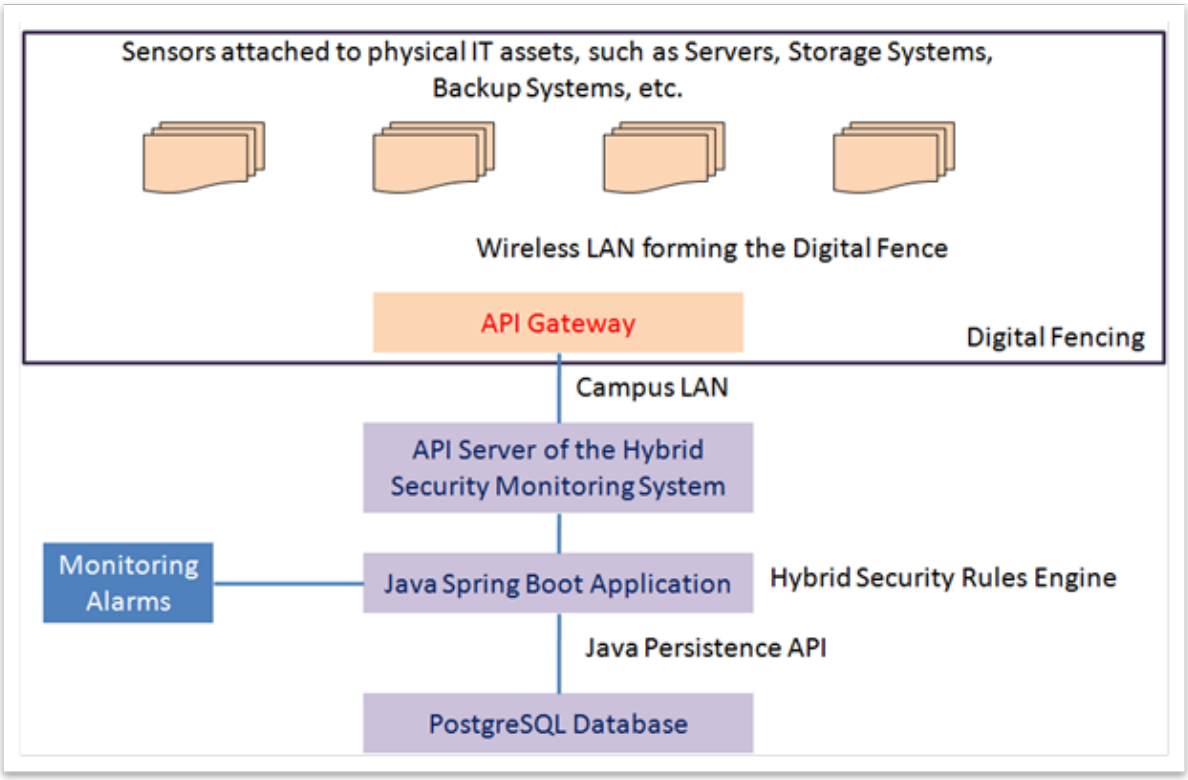

Figure 8: Hybrid Security Monitoring System

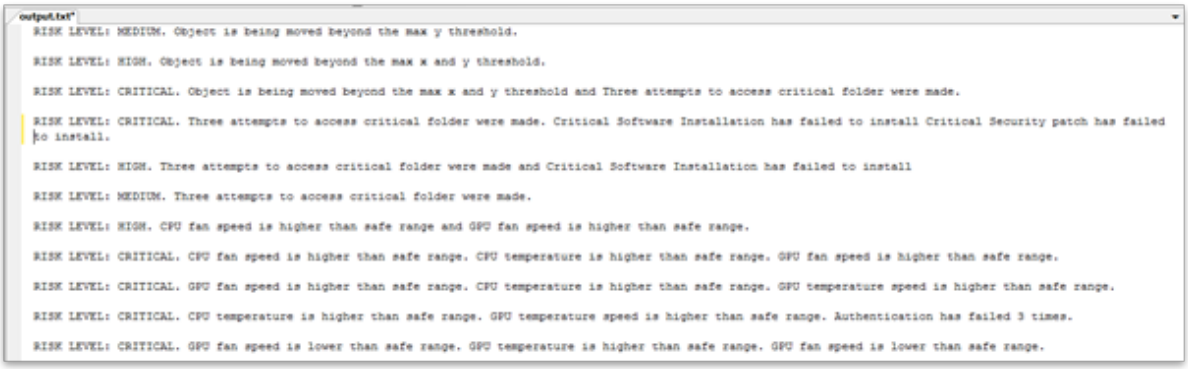

Figure 9: Final output file recording all the risk levels

In this research, security design literature on IIoT/CPS was reviewed. Not all research studies explained integrated designs but majority of them covered the essential components used in the designs. These studies were useful in defining the components those were possible to be programmed and configured within a pilot environment in an Ubuntu laptop. The study by (Rahman et al.; 2016) provided design idea of security controls positioned in a multilayered CPS/IIoT architecture. Further, the study by (Ammar et al.; 2018) provided insight into how these layers are realised in a real world architecture. The research studies by (Basri and Elkhadimi; 2020; Braggaar; 2018; Frankó et al.; 2020) provided insight into secure localisation as a significant control of physical security. (Mugarza et al.; 2020) presented a standard for implementing critical security patch

management. The research studies by (Happa et al.; 2019; Plósz et al.; 2017) provided insights into collaborative security controls when multiple parameters related to different verticals are monitored collectively to identify and report risk levels. Finally, the research studies by (Gunduz and Das; 2020; Lei et al.; 2018; Marksteiner et al.; 2019; Neureiter et al.; 2016) provided insight into hybrid security configuration in Smart Grid where it is implemented and tested the most. These studies provided detailed ideas to be realised as design details for the experimentation. The experimentation was successful as there were no errors in the final setting, and the performance of the Risk analysis was 100% (100% risks identified, 0% false positives identified). Hence, it is hereby declared that this research was successful in meeting its aim and objectives and could get the answer to the research question reasonably.

The components used in the design are already popular in the commercial world. The Spring Boot framework is popular for mission critical applications, although there are multiple alternatives ((Webb et al.; 2020)). However, Java has its original and unique place in designing the layers of a mission critical application. Core Java comprises the foundation of advanced systemic programming, which cannot be avoided while designing mission critical applications ((Gosling et al.; 2021)). It is one of the most organised, easy to learn and widely deployed programming. The experimentation setting of this research can be very much realised in the real world. However, several enhancements are needed. For example, the networking will be complex and risk decision making in a few tens of milliseconds (as achieved in the experimentation) will be hard to achieve. The API server will be contacted by hundreds of API gateways. Hence, it requires very high capacities of hardware and networking. Preferably, it should be cloud hosted even if the main controller application is hosted on premises. The decision engine (controller) code may have to be divided into multiple controller codes as this research had to write about a hundred pages of code for only eleven parameters. In actual practice, there may be hundreds of parameters to be monitored and controlled.

7 Conclusion and Future Work

This research was conducted following directions of one aim, four objectives, and a research question. This section presents detailed discussion on how the objectives were met, and what answer to the research question was obtained.

Objectives:
(a) To explore hybrid security designs in industrial control systems using Industrial Internet of Things architecture to capture the design specifications:

This objective was realised by studying a number of literature on IIoT/CPS security modelling components, localisation security, hybrid security, and collaborative security. Inputs from all these literature were used to develop the design specifications of the experimentation setting.

(b) To implement an experimentation environment in a laptop following the design specifications and conduct coding and configurations as required to operate it as a fully configured pilot project;

This objective was realised by designing and implementing an experimentation setup in Maven and Java Spring Boot connected with an API gateway server. This entire design was realised based on inputs from the literature view.

(c) To run test cases of different risk levels and determine whether the experimentation environment detects and logs the correct risk level based on its rules engine specifications;

This objective was realised by defining four test case scenarios and conducting comprehensive testing in each of the scenarios.

(d) To describe the practical validity of and future expansion of the hybrid security system;

This objective was realised in the previous section where the practical validity and future expansion of the experimentation design and its setting was explained.

The research question of this research was the following:

How a hybrid security system can be designed to protect industrial controllers and the devices administered by them using the Industrial Internet of Things architecture?

This research could answer the research question reasonably although not comprehensively. It was based on a basic design tested within a laptop environment. Hence, the findings were limited to risk identification success (or failure, if was evident) in the four risk scenarios and the decision making time taken by the location controller. In larger experimentation settings, more findings may be reported providing wider answers to this question.

This project can be commercialised in future given that the experiment involved technically tested and commercially used products. For example, the Spring Boot framework is very popular commercially and the codes used in this experiment can be expanded for future commercialisation. However, the expansion needs to be significant. This research used only one module of products in a digital fence talking to one API server through one API gateway. In practical applications, there may be several API servers and perhaps hundreds of API gateways each serving a digital fenced module. One controller Java code file cannot handle such volumes. Designers may create several such controllers attached with one or more API servers. For academic study, a real world digital fence may be pursued in the future. The devices may be real having their sensors communicating with a data consolidation layer (or a data transmission object layer) that can consolidate all sensor data into JSON object files. The JSON object files shall be larger than the one used in this research. All these need to be conducted in a real network scenario. Triangulation for localisation may be implemented using four Wi-Fi routers. The overall setting will be much more expensive than this research but will generate many more useful results.

References

(n.d.).

Ammar, M., Russello, G. and Crispo, B. (2018). Internet of things: A survey on the security of iot frameworks, *Journal of Information Security and Applications* **38**: 8–27.

Basri, C. and Elkhadimi, A. (2020). A review on indoor localization with internet of things, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **XLIV-4/W3-2020**: 121–128.

Boyes, H., Hallaq, B., Cunningham, J. and Watson, T. (2018). The industrial internet of things (iiot): An analysis framework, *Computers in industry* **101**: 1–12.

Braggaar, R. (2018). Wi-fi network-based indoor localisation: The case of the tu delft campus, pp. 1–92.

Frankó, A., Vida, G. and Varga, P. (2020). Reliable identification schemes for asset and production tracking in industry 4.0, *Sensors* **20**(13): 3709.

Gosling, J., Joy, B., Steele, G., Bracha, G., Buckley, A. and Smith, D. (2021). The javar language specification–java se 17 edition, *Oracle Corporation* .

Gunduz, M. Z. and Das, R. (2020). Cyber-security on smart grid: Threats and potential solutions, *Computer networks* **169**: 107094.

Happa, J., Glencross, M. and Steed, A. (2019). Cyber security threats and challenges in collaborative mixed-reality, *Frontiers in ICT* **6**.

Kobara, K. (2016). Cyber physical security for industrial control systems and iot, *IEICE TRANSACTIONS on Information and Systems* **99**(4): 787–795.

Lei, H., Chen, B., Butler-Purry, K. L. and Singh, C. (2018). Security and reliability perspectives in cyber-physical smart grids, *2018 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia)*, IEEE, pp. 42–47.

Malik, P. K., Sharma, R., Singh, R., Gehlot, A., Satapathy, S. C., Alnumay, W. S., Pelusi, D., Ghosh, U. and Nayak, J. (2021). Industrial internet of things and its applications in industry 4.0: State of the art, *Computer Communications* **166**: 125–139.

Maple, C. (2017). Security and privacy in the internet of things, *Journal of cyber policy* **2**(2): 155–184.

Marksteiner, S., Vallant, H. and Nahrgang, K. (2019). Cyber security requirements engineering for low-voltage distribution smart grid architectures using threat modeling, *Journal of Information Security and Applications* **49**: 102389.

Mugarza, I., Flores, J. L. and Montero, J. L. (2020). Security issues and software updates management in the industrial internet of things (iiot) era, *Sensors* **20**(24): 7160.

.NET Cloud Client Libraries (2022).
URL: *https://cloud.google.com/dotnet/docs/reference*

Neureiter, C., Eibl, G., Engel, D., Schlegel, S. and Uslar, M. (2016). A concept for engineering smart grid security requirements based on sgam models, *Computer Science-Research and Development* **31**(1): 65–71.

Ning, H., Liu, H. and Yang, L. T. (2013). Cyberentity security in the internet of things, *Computer* **46**(4): 46–53.

Plósz, S., Schmittner, C. and Varga, P. (2017). Combining safety and security analysis for industrial collaborative automation systems, *International Conference on Computer Safety, Reliability, and Security*, Springer, pp. 187–198.

Rahman, A. F. A., Daud, M. and Mohamad, M. Z. (2016). Securing sensor to cloud ecosystem using internet of things (iot) security framework, *Proceedings of the International Conference on Internet of things and Cloud Computing*, pp. 1–5.

Roman, R., Zhou, J. and Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things, *Computer Networks* **57**(10): 2266–2279.

Sadeghi, A.-R., Wachsmann, C. and Waidner, M. (2015). Security and privacy challenges in industrial internet of things, *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, IEEE, pp. 1–6.

Schumacher, A., Schumacher, C. and Sihn, W. (2019). Industry 4.0 operationalization based on an integrated framework of industrial digitalization and automation, *Proceedings of the International Symposium for Production Research 2019*, Springer, pp. 301–310.

Sharma, V., Choudhary, G., Ko, Y. and You, I. (2018). Behavior and vulnerability assessment of drones-enabled industrial internet of things (iiot), *IEEE Access* **6**: 43368–43383.

Vasilomanolakis, E., Daubert, J., Luthra, M., Gazis, V., Wiesmaier, A. and Kikiras, P. (2015). On the security and privacy of internet of things architectures and systems, *2015 International workshop on secure internet of things (SIoT)*, IEEE, pp. 49–57.

Vermeer, B. (n.d.). Spring dominates the java ecosystem with 60% using it for their main applications.

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C., Deleuze, S., Simons, M. et al. (2020). Spring boot reference documentation, *Retrieved June* **22**.

Yousuf, T., Mahmoud, R., Aloul, F. and Zualkernan, I. (2015). Internet of things (iot) security: current status, challenges and countermeasures, *International Journal for Information Security Research (IJISR)* **5**(4): 608–616.