DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2021

Can a robot's confidentiality be trusted?

JESPER ÖBERG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Can a robot’s confidentiality be trusted?

JESPER ÖBERG

Bachelor in Computer Science
Date: June 9, 2021
Supervisor: Robert Lagerström
Examiner: Pawel Herman
School of Electrical Engineering and Computer Science
Swedish title: Går det att lita på tystnadspliken hos en robot?

Contents

0.1 Acronyms

IT - Information Technology
IoT - Internet of Things
OS - Operating System
ROS - Robot Operating System
VM - Virtual Machine
STRIDE - Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege
DREAD - Damage, Reproducibility, Exploitability, Affected users, Discoverability
CVE - Common Vulnerability and Exposures

Chapter 1

Introduction

Hacking is a well know concept today where almost weekly one can hear on the news that some company is affected by it. Everything connected to the internet could be targeted by hackers, how vulnerable something is depends on the level of cyber security. A useful way to test the cyber security capacity of a device connected to the internet, also called internet of things (IoT), is by hacking it.

There are three types of hat; white, black and grey[1]. The white hat is when the hacker has permission from the company to hack with the intent to analyse potential vulnerabilities in order to report back to the company. A grey hat attacks a target without the company's knowledge but with no evil intent. When a vulnerability is found the grey hat will report it to the company, sometimes even with a promise of a reward. Black hat hackers also attack a target without knowledge or approval by the company. but what differentiates them from grey hats is their intent, which is often about collecting sensitive data or making the target unavailable for the user. All the different types of hats are called hackers but the white hat is often referred to as ethical hacker or a penetrations tester. Where ethical hacking is a moral aspect and penetration testing is more of a job title.

When a company has a new idea which will result in creation of a new IoT device security is seldom something that is a part of the foundation. It is very common when creating something that security comes in second hand, it is usually the fun or important part of one's idea that comes first. Even when security is implemented the level of security in IoT is not something that gets

[1]https://www.itgovernance.eu/blog/en/ethical-hacking-vs-penetration-testing-whats-the-difference

prioritized[2].

One particular type of IoT that this report will analyse is a robot. Robots have been used for many years and are well known to be used in the automotive industry. The industry has for the last couple of years begun to transform itself into the fourth industrial revolution, which in reality means that it is being connected to the internet . A more common name for this transformation is smart, e.g. smart machine, smart factory, smart home or smart TV. Smart means connected to a network, often it means the Internet [3].

Robots can be used for a various different things today, their usability is not restricted to one industry. But something that is still in common for most of the robots that are used today is that they are all connected to the internet and become an IoT device, which makes them vulnerable to cyber attacks. One study that is done researching different types of robots found many vulnerabilities which could lead to many different types of possible threats depending on what each robot has equipped [1].

1.1 Scope

This thesis is a cyber security evaluation of a robot connected to the internet. The scope of this study will be determined by analysing vulnerabilities from a threat model. Depending on what vulnerabilities that are found, those that are deemed most critical will be the focus of this thesis. Each possible vulnerability found in an architectural overview will be categorized with the use of STRIDE and DREAD, which will result in a scoring system where those with the highest points are the most critical and will be analysed. One part of the system that will not be analysed at all is the connection to the cloud, due to restrictions from the company. The goal of this thesis is to do a cyber security evaluation of an IoT device, which in this report is a robot

1.2 Research question

There are two main question that will be the focus of this thesis; how a robot can be vulnerable to cyber security attacks and does the vulnerability affect the integrity of its user?

[2]https://www.bitdefender.com/box/blog/iot-news/iot-security-still-not-priority-survey-reveals/

[3]https://en.wikipedia.org/wiki/Smart_device

Chapter 2

Background

2.1 Related Work

There has not been any previous cyber security research done on this robot, nor have any known cyber security evaluation been done. A case study [2] on the cyber security of robots was encountered during literature search. It examined the different ways in which the robot can be compromised. Issues such as a lack of robot authentication, sharing a wireless network connection, hacking into the robot's video feed, as well as easy access were discovered. There was also research that during development of an assistant robot for the elderly evaluated effects of different security measures on the performance of the robot [3].

Other research[4] [5] on robots was more of a theoretical and analytical character where aspects such as attack and threat scenarios were looked at closely. One of few research papers[6] that went through hacking of a robot was a digital forensic investigation of a hacked industrial collaborative robot called Universal Robots UR3. There is also research[7] that present common development environment for robotics and cyber security researchers called Alurity in order to eliminate difficulties concerning expensive prices and shortage of the robot hardware. Nonetheless, literature on cyber security of robots involving penetration testing and ethical hacking seems to be severely limited. Research on IoT and ethical hacking, on the other hand, were more frequently encountered.

2.2 Ethical hacking

Penetration testing, also known as ethical hacking or pen testing, is a process where a network application, a web application, or an entire computer system gets scanned for security vulnerabilities that could be exploited. Penetration testing can be done by hand as well as with special software.

There are operating systems made for penetration testers. One of the more popular ones is the Debian-based Linux penetration testing distribution Kali designed specifically for digital forensics and penetration testing. It is open source and is packaged with many software programs needed to automate many kinds of penetration testing[1].

The cyber security evaluation that will be performed on the robot can also be called a penetration test. In order to do a successfully cyber security evaluation the procedure consists of three parts: planning, exploitation and reporting. The first part, planning, is about information gathering and threat and vulnerability analysis. The second part, exploitation, will be the more practical part, which is often seen as the hacking or penetration test part. And the last, reporting, is really the most important one, if one is to perform ethical hacking.

The result of the first part, planning, will result in a threat model. Where the potential threats and vulnerabilities will be analysed and categorized. This will also be a guideline on where the focus for further tests will be done. The second part, exploitation, is the practical part. Here the vulnerability will be tested to see if it is vulnerable or not. It may be tested with software programs or a more hands-on approach. When a vulnerability has been successfully exploited and the system has been breached, the breach itself must be tested. What can this now unauthorized user do on the system? This is often tested by uploading some malicious payload. The result of exploitation can be seen in the result section of this thesis. For the last part of three in the cyber security evaluation, reporting, the one hacking reports everything that has been tested and the results of it. Which in this case will be the thesis as a whole.

2.3 Cyber security assessment

Security assessment work can look different depending on the dimensions of the prior knowledge available to security researchers. There are three different categories these assessments are divided into: black box, white box and grey box.

[1]https://www.kali.org/

- Black box is done with no prior insider knowledge of the system. This type of security assessment can be conducted by security researchers, consultants as well as internal security teams[2]. The black box is often viewed as the realistic approach, meaning that this is similar to what an black hat hacker would have. When a system is successfully breached from a black box perspective, it might not be the most vulnerable attack but it will be the one of the most critical ones. Because if someone with no inside information from the company can hack it means anyone with enough knowledge can hack it.
- White box on the other hand is the opposite of black box meaning security researchers get full access to all insider information about the product, often that means the source code. A white box approach is the most thorough security assessment. It is quite likely that the white approach will find the biggest vulnerabilities but one thing to remember is that the biggest vulnerability does not have to be the most critical one. Here it is the opposite from black box. Were the hacker only able to find it because of the inside information or is it easy for anyone to find.
- Grey box as the name suggests is combination of the two above. Here researchers possess limited insider knowledge. But will still have some type of inside information that is not publicly available. For example, one might not get to analyse any code but can discuss about system's architectural overview with a programmer who has written it. It could even be as simple as given hint on what protocol that is insecure.

2.3.1 STRIDE and DREAD

Threat modelling can be done manually or with tools that automate the process. The threat model can be seen as architectural overview of the system that is tested. It will be displayed by a graph, where a node is either a part of the system or a user interacting with it and the edges are between where the communication occurs. To categorize threat modelling STRIDE and DREAD types are known[3].

STRIDE is a method for identifying threat use cases and at the same time both categorizing and ranking them. The STRIDE model groups threats into

[2]https://learning.oreilly.com/library/view/iot-penetration-testing/9781787280571/ecdf6d97-b5bd-4512-814b-16f56df73d0f.xhtml

[3]https://learning.oreilly.com/library/view/iot-penetration-testing/9781787280571/567e9064-4de1-465f-a2c5-46920d269026.xhtml

six categories in order to formulate questions to discover possible threats. STRIDE stands for Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege.

- First threat category, **spoofing**, is when a person or program makes an attempt to gain access to a system without authorization by pretending to be someone how do.
- **Tampering** is when attacker modifies data, for example, as it flows over a network between two devices.
- **Repudiation** is when an attacker tries to hide their own attacks by, for example, erasing all possible traces that they might have left after themselves that can be detected. Spoofing credentials of another user is also a repudiation measure taken as the choice of attack already from the beginning.
- Next category, **Information disclosure**, is when private data is exposed to an unauthorized user. This could could be when a user views a file that it is not authorized to see or it could be by monitoring the network traffic. This type of information can be really useful for an attacker.
- **Denial of service** is when an attacker makes a system to become unavailable to others. For example, a denial-of-service attack can be done by sending a server with countless requests to use up all system resources.
- Final category, **Elevation of privileges** is when an attacker gains additional privileges. This can be done, for example, by tampering the system and modifying the attacker's own privileges or by spoofing someone with higher privileges.

In order to be able to measure and rate risks of each threat an additional rating system is used. DREAD is a common tool used for this, which is more focused on the damages done one the vulnerability has successfully been exploited. This rating system stands for the following:

- Damage potential - the magnitude of damage when exploited
- Reproducibility - the amount of effort and/or skill needed to accomplish reproducing the attack
- Exploitability - the amount of effort and/or skill needed to accomplish the attack

- Affected users - approximation of the number of victims of the attacks
- Discoverability - the amount of effort and/or skill needed to accomplish finding the vulnerability

Risk rating system of DREAD ranges between 1 and 3. This range is in descending order so 1 being lowest risk and 3 highest.

For automation of these processes' tools such as Coreline, Foreseeti and SecuriCAD can be used for threat modelling as well as threat rating. Automation tools in general suit white and/or grey box approaches more, since it can be hard to run a program testing a systems vulnerabilities without access to the system. This means that one would have to gain access if not given, which would require exploitation of a vulnerability.

When gathering information about the device all potential attack vectors are equally important to consider, but they all require different kinds of accessibility. Depending on the product, limitations and relevancy of hardware and firmware, radio communication, network services, web applications, cloud APIs and mobile applications should be properly scanned as potential attack vectors[4].

2.4 Network services

Network services consists of different ports that are connected to the internet to make communication possible. To gather information about which protocol and services that are used for a specific device a port scanning tool like Nmap can be used[5]. The information collected from Nmap is then analysed for further investigation.

2.5 Web application

The website uses a HTTP service to host a web design, it can be used for a lot of different purposes. A common way to analyse the web application is by using a web page crawler to discover all webpages connected to the application in order to find out more about its development techniques.

[4]https://nse.digital/pages/guides/pentest-process-planning.html
[5]https://nmap.org/

2.6 Delimitation

Due to mostly relevance and accessibility the focus on this cyber security evaluation will be on network services and web application. As previous stated there several more to consider when evaluating the security of a IoT device.

2.6.1 Hardware and firmware attacks

The hardware attack consist of adding hardware back doors, trojans or hardware modifications. It also gives the possibility of reverse engineering components of the robot. Hardware trojans embedded in processors can be enabled with software or leak critical information[8].

A way to attack the OS of a robot would be to through an update of its firmware. The firmware update remotely by an internet connections or physically by e.g. inserting an USB.

One research that has evaluated autonomous system, which are build using ROS, have concentrated their work on the importance of encrypted ROS communication[3]. This is needed because it otherwise pose a great threat and the communication could easily be interfere, where private messages and even supersede node can be read. But unfortunately the result show that encryption impose significant delays, which makes it interesting to see how many prioritize this.

2.6.2 Radio communication

The only communication that is used by the robot is through the network services.

2.6.3 Cloud API

This will not be analysed in this thesis due to restriction by the manufacture company of the robot.

2.6.4 Mobile application

This is not applicable since there is no mobile application connected to the robot.

Chapter 3

Methods

When preforming a cyber security evaluation of an IoT device there are a lot of different ways to do it and depending on how and for what the product is used the evaluation focus can differ. Even if the focus may be different, the steps that need to be performed follow the same structure. The first step is planning which often means information gathering, this part of the project started out from a black box perspective. In this case it intended to find all publicly available information online to begin to create an architectural overview of how the communication between user and robot occurs. When the architectural overview is categorized and ranked with the use of STRIDE and DREAD it becomes a threat model. It is important to not just consider the regular users of the robot. To be able to control the device and where the real harm can be done is by elevating one's privileges from a user to an admin.

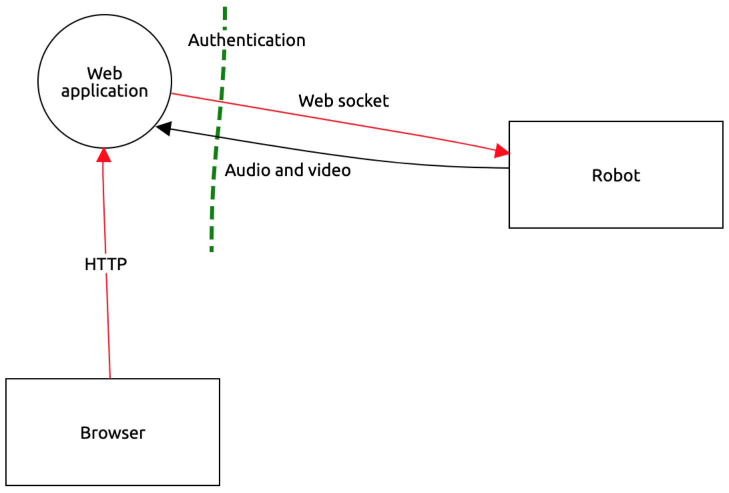
When all the public available information was gathered from the internet and the threat model were made. The next step of information gathering began. That was to see what information the robot leaked without doing any penetration attempts on the system. This was mainly done by two programs: Nmap and Whirshark. To follow up any possible vulnerabilities found by Nmap the software program Metaspolite were also used. So, before the actual explosions phase could begin all possible vulnerabilities needed to be accounted for.

The next part to check a robot's cyber security is exploitation. Now that the potential vulnerabilities have been found its time to test them and see what possible damage can be done. In this part the process of testing the vulnerabilities can differ between attack vectors. But they have the same end goal, to gain access to the system or manipulate the information between the user and server.

3.1 Threat model

Threat modelling is a process where testers analyse attacks and threats before moving to the next step which is conducting pen testing[9]. It provides a structured way of doing so and securing the software in question[10].

The threat model will follow at least two different types of ranking systems, STRIDE[1] and DREAD[2]. Evaluating results from different threat models will be done with the help of [11] and [12]. Depending on the threat model that have been made suspected vulnerabilities will be analysed and penetration tested. As a threat model is created to find vulnerabilities the focus will on gaining access to the system.



3.2 Penetration testing

Kali Linux operating system and its software applications will be used for information gathering, such as scanning, sniffing, and exploitation, such as testing vulnerabilities and trying to make use of malicious payloads.

[1]https://docs.microsoft.com/sv-se/archive/blogs/larryosterman/threat-modeling-again-stride

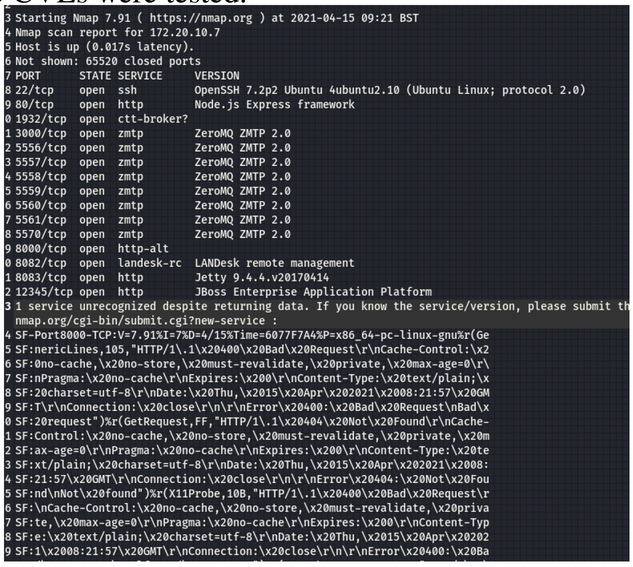
[2]https://docs.microsoft.com/en-us/archive/blogs/david_leblanc/dreadful

The evaluation of this study will be from the results of those penetration tests and how big of an impact they might have to the integrity of the system. Regardless of a successfully penetration or not the result of this study will be an evaluation of the robot's cyber security.

Chapter 4

Results

The first thing that was done when evaluating the robot was to scan the network connected to it and sniffing all its communication. From the scans that were acquired with the use of Nmap, all open ports were further investigated for possible known vulnerabilities, also known as CVE (Common Vulnerabilities and Exposures). The main goal at the beginning was to find if any CVE's where applicable against the robots architecture. With the use of Metasploit two CVEs were tested.



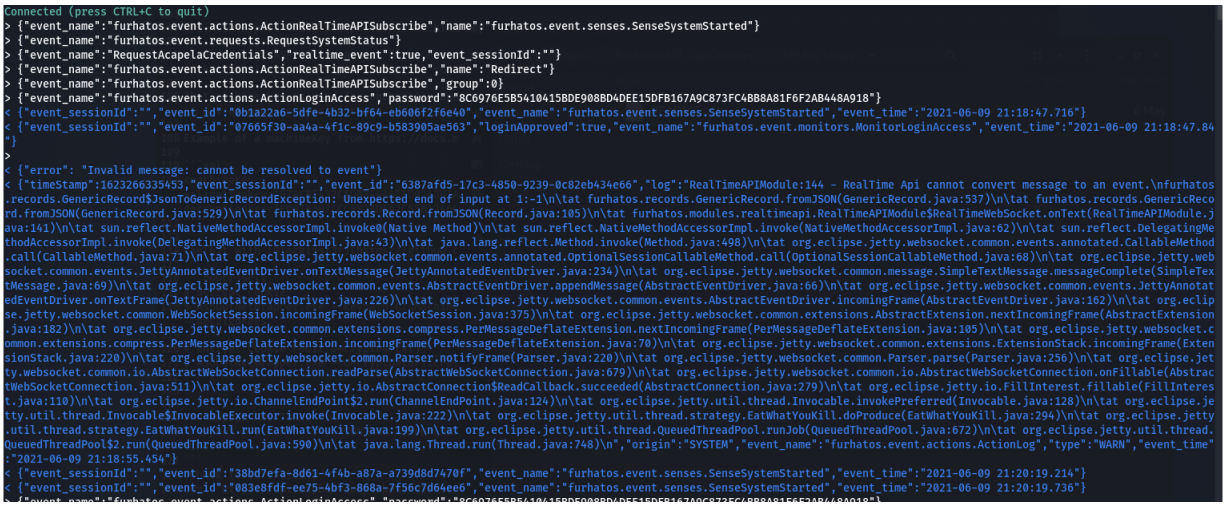
```
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-15 09:21 BST
Nmap scan report for 172.20.10.7
Host is up (0.017s latency).
Not shown: 65520 closed ports
PORT      STATE SERVICE     VERSION
22/tcp    open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http        Node.js Express framework
1932/tcp  open  ctt-broker?
3000/tcp  open  zmtp        ZeroMQ ZMTP 2.0
5556/tcp  open  zmtp        ZeroMQ ZMTP 2.0
5557/tcp  open  zmtp        ZeroMQ ZMTP 2.0
5558/tcp  open  zmtp        ZeroMQ ZMTP 2.0
5559/tcp  open  zmtp        ZeroMQ ZMTP 2.0
5560/tcp  open  zmtp        ZeroMQ ZMTP 2.0
5561/tcp  open  zmtp        ZeroMQ ZMTP 2.0
5570/tcp  open  zmtp        ZeroMQ ZMTP 2.0
8000/tcp  open  http-alt
8082/tcp  open  landesk-rc  LANDesk remote management
8083/tcp  open  http        Jetty 9.4.4.v20170414
12345/tcp open  http        JBoss Enterprise Application Platform
1 service unrecognized despite returning data. If you know the service/version, please submit th
nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port8000-TCP:V=7.91%I=7%D=4/15%Time=6077F7A4%P=x86_64-pc-linux-gnu%r(Ge
SF:nericLines,105,"HTTP/1\.1\x20400\x20Bad\x20Request\r\nCache-Control:\x2
SF:0no-cache,\x20no-store,\x20must-revalidate,\x20private,\x20max-age=0\r\
SF:nPragma:\x20no-cache\r\nExpires:\x200\r\nContent-Type:\x20text/plain;\x
SF:20charset=utf-8\r\nDate:\x20Thu,\x2015\x20Apr\x202021\x2008:21:57\x20GM
SF:T\r\nConnection:\x20close\r\n\r\nError\x20400:\x20Bad\x20Request\nBad\x
SF:20request")%r(GetRequest,FF,"HTTP/1\.1\x20404\x20Not\x20Found\r\nCache-
SF:Control:\x20no-cache,\x20no-store,\x20must-revalidate,\x20private,\x20m
SF:ax-age=0\r\nPragma:\x20no-cache\r\nExpires:\x200\r\nContent-Type:\x20te
SF:xt/plain;\x20charset=utf-8\r\nDate:\x20Thu,\x2015\x20Apr\x202021\x2008:
SF:21:57\x20GMT\r\nConnection:\x20close\r\n\r\nError\x20404:\x20Not\x20Fou
SF:nd\nNot\x20found")%r(X11Probe,10B,"HTTP/1\.1\x20400\x20Bad\x20Request\r
SF:\nCache-Control:\x20no-cache,\x20no-store,\x20must-revalidate,\x20priva
SF:te,\x20max-age=0\r\nPragma:\x20no-cache\r\nExpires:\x200\r\nContent-Typ
SF:e:\x20text/plain;\x20charset=utf-8\r\nDate:\x20Thu,\x2015\x20Apr\x20202
SF:1\x2008:21:57\x20GMT\r\nConnection:\x20close\r\n\r\nError\x20400:\x20Ba
```

First one was against their HTTP server. The HTTP server is Apache which was vulnerable to a denial of service (DOS) attack. The attack were successful in that matter that in order to continue use of the robot we needed to reboot it. But no further information were leaked from the system.

The other CVE that was found was on port 22, which is often used for SSH. In this case they used Open-SSH version 7.2. The vulnerability that was found could with this exploit try out different usernames and be able to find out if that username is connected to a user or not. When cracking user's authentication to the SSH a common method is the use of brute force. If one does not know either the username or the password, completing a brute force attack can be very time-consuming. But if one were to know the username the complexity goes from exponential to polynomial as you would only have to test a list of common passwords. For this there existed a python program which could be used on the SSH, port 22. For the list on known typical usernames that were used on this SSH port the only one that where found was *root*. When the username root was found another tool from Kali was used, this time it was Hydra, which is a common tool when attempting brute force. When installing the Kali operating system several of its software tools comes with lists containing most common usernames and passwords.

The communication between user and robot goes through a web socket. When the user authenticates to the robots they use a web browser and connect to its IP address. The IP address will open a web application where the user needs to authenticate with its password. The web application uses HTTP which is protocol that is not encrypted which means that traffic to it can be sniffed. This leads to another vulnerability that was found during a sniffing session where network traffic is scanned and analysed. It was at an authentication processes where the message that contained the password could be found. The password itself could not be read in plain text but the hashed SHA256 value could be seen. The users password is encrypted with SHA256 but it is not salted, so anyone can use the same hashed code to authenticate. This leads to the situation where someone could connect to the same web socket port just entered the same message got the same credentials as the user who just logged in. When the authentication process is complete it is possible to connect directly to the robot's web application where you as a user have full control over the robots interface and can upload a payload with the file extension skill[1].

[1]https://file.org/extension/skill



```
Connected (press CTRL+C to quit)
> {"event_name":"furhatos.event.actions.ActionRealTimeAPISubscribe","name":"furhatos.event.senses.SenseSystemStarted"}
> {"event_name":"furhatos.event.requests.RequestSystemStatus"}
> {"event_name":"RequestAcapelaCredentials","realtime_event":true,"event_sessionId":""}
> {"event_name":"furhatos.event.actions.ActionRealTimeAPISubscribe","name":"Redirect"}
> {"event_name":"furhatos.event.actions.ActionRealTimeAPISubscribe","group":0}
> {"event_name":"furhatos.event.actions.ActionLoginAccess","password":"8C6976E5B5410415BDE908BD4DEE15DFB167A9C873FC4BB8A81F6F2AB448A918"}
< {"event_sessionId":"","event_id":"0b1a22a6-5dfe-4b32-bf64-eb606f2f6e40","event_name":"furhatos.event.senses.SenseSystemStarted","event_time":"2021-06-09 21:18:47.716"}
< {"event_sessionId":"","event_id":"07665f30-aa4a-4f1c-89c9-b583905ae563","loginApproved":true,"event_name":"furhatos.event.monitors.MonitorLoginAccess","event_time":"2021-06-09 21:18:47.84
"}
>
< {"error": "Invalid message: cannot be resolved to event"}
< {"timeStamp":1623266335453,"event_sessionId":"","event_id":"6387afd5-17c3-4850-9239-0c82eb434e66","log":"RealTimeAPIModule:144 - RealTime Api cannot convert message to an event.\nfurhatos
.records.GenericRecord$JsonToGenericRecordException: Unexpected end of input at 1:-1\n\tat furhatos.records.GenericRecord.fromJSON(GenericRecord.java:537)\n\tat furhatos.records.GenericReco
rd.fromJSON(GenericRecord.java:529)\n\tat furhatos.records.Record.fromJSON(Record.java:105)\n\tat furhatos.modules.realtimeapi.RealTimeAPIModule$RealTimeWebSocket.onText(RealTimeAPIModule.j
ava:141)\n\tat sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)\n\tat sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)\n\tat sun.reflect.DelegatingMe
thodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)\n\tat java.lang.reflect.Method.invoke(Method.java:498)\n\tat org.eclipse.jetty.websocket.common.events.annotated.CallableMethod
.call(CallableMethod.java:71)\n\tat org.eclipse.jetty.websocket.common.events.annotated.OptionalSessionCallableMethod.call(OptionalSessionCallableMethod.java:68)\n\tat org.eclipse.jetty.web
socket.common.events.JettyAnnotatedEventDriver.onTextMessage(JettyAnnotatedEventDriver.java:234)\n\tat org.eclipse.jetty.websocket.common.message.SimpleTextMessage.messageComplete(SimpleTex
tMessage.java:69)\n\tat org.eclipse.jetty.websocket.common.events.AbstractEventDriver.appendMessage(AbstractEventDriver.java:66)\n\tat org.eclipse.jetty.websocket.common.events.JettyAnnotat
edEventDriver.onTextFrame(JettyAnnotatedEventDriver.java:226)\n\tat org.eclipse.jetty.websocket.common.events.AbstractEventDriver.incomingFrame(AbstractEventDriver.java:162)\n\tat org.eclip
se.jetty.websocket.common.WebSocketSession.incomingFrame(WebSocketSession.java:375)\n\tat org.eclipse.jetty.websocket.common.extensions.AbstractExtension.nextIncomingFrame(AbstractExtension
.java:182)\n\tat org.eclipse.jetty.websocket.common.extensions.compress.PerMessageDeflateExtension.nextIncomingFrame(PerMessageDeflateExtension.java:105)\n\tat org.eclipse.jetty.websocket.c
ommon.extensions.compress.PerMessageDeflateExtension.incomingFrame(PerMessageDeflateExtension.java:70)\n\tat org.eclipse.jetty.websocket.common.extensions.ExtensionStack.incomingFrame(Exten
sionStack.java:220)\n\tat org.eclipse.jetty.websocket.common.Parser.notifyFrame(Parser.java:220)\n\tat org.eclipse.jetty.websocket.common.Parser.parse(Parser.java:256)\n\tat org.eclipse.jet
ty.websocket.common.io.AbstractWebSocketConnection.readParse(AbstractWebSocketConnection.java:679)\n\tat org.eclipse.jetty.websocket.common.io.AbstractWebSocketConnection.onFillable(Abstrac
tWebSocketConnection.java:511)\n\tat org.eclipse.jetty.io.AbstractConnection$ReadCallback.succeeded(AbstractConnection.java:279)\n\tat org.eclipse.jetty.io.FillInterest.fillable(FillInteres
t.java:110)\n\tat org.eclipse.jetty.io.ChannelEndPoint$2.run(ChannelEndPoint.java:124)\n\tat org.eclipse.jetty.util.thread.Invocable.invokePreferred(Invocable.java:128)\n\tat org.eclipse.je
tty.util.thread.Invocable$InvocableExecutor.invoke(Invocable.java:222)\n\tat org.eclipse.jetty.util.thread.strategy.EatWhatYouKill.doProduce(EatWhatYouKill.java:294)\n\tat org.eclipse.jetty
.util.thread.strategy.EatWhatYouKill.run(EatWhatYouKill.java:199)\n\tat org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:672)\n\tat org.eclipse.jetty.util.thread.
QueuedThreadPool$2.run(QueuedThreadPool.java:590)\n\tat java.lang.Thread.run(Thread.java:748)\n","origin":"SYSTEM","event_name":"furhatos.event.actions.ActionLog","type":"WARN","event_time"
:"2021-06-09 21:18:55.454"}
< {"event_sessionId":"","event_id":"38bd7efa-8d61-4f4b-a87a-a739d8d7470f","event_name":"furhatos.event.senses.SenseSystemStarted","event_time":"2021-06-09 21:20:19.214"}
< {"event_sessionId":"","event_id":"083e8fdf-ee75-4bf3-868a-7f56c7d64ee6","event_name":"furhatos.event.senses.SenseSystemStarted","event_time":"2021-06-09 21:20:19.736"}
> {"event_name":"furhatos.event.actions.ActionLoginAccess","password":"8C6976E5B5410415BDE908BD4DEE15DFB167A9C873FC4BB8A81F6F2AB448A918"}
```

Chapter 5

Discussion

The vulnerability that was found would gain an attacker the same privilege as a system user of the robot. Since there is a difference from just a regular user that interacts with the robot when a skill is running, e.g., when the robot runs as a fruit salesman then the user can do any systemically harm against the robot. A regular user has a conversation that the robot understands in which you would be able to buy fruits from it. But when one either has authentication or through this hack gain that same level of authentication it is possible to gain the access of a system user. A system user has more control over the robot, not the same security level to the system as an admin but can more be seen as a developer of the robot. How to develop the robot and what type of damage that it might can be to a regular user is still to be determent.

Bibliography

[1] Cesar Cerrudo and Lucas Apa. “Hacking robots before skynet”. In: *IOActive Website* (2017), pp. 1–17.

[2] Justin Miller, Andrew B. Williams, and Debbie Perouli. “A Case Study on the Cybersecurity of Social Robots”. In: *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’18. Chicago, IL, USA: Association for Computing Machinery, 2018, pp. 195–196.

[3] Francisco Javier Rodrıguez Lera et al. “Cybersecurity in autonomous systems: Evaluating the performance of hardening ROS”. In: *XVII Workshop en Agentes Fısicos* 47 (2016).

[4] Federico Maggi et al. “Rogue robots: Testing the limits of an industrial robot’s security”. In: *Trend Micro, Politecnico di Milano, Tech. Rep* (2017).

[5] G. W. Clark, M. V. Doran, and T. R. Andel. “Cybersecurity issues in robotics”. In: *2017 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*. 2017, pp. 1–5.

[6] Yanan Gong et al. “Forensic Investigation of A Hacked Industrial Robot”. In: *Critical Infrastructure Protection XIV*. Ed. by Jason Staggs and Sujeet Shenoi. Cham: Springer International Publishing, 2020, pp. 221–241.

[7] Vıctor Mayoral-Vilches et al. “alurity, a toolbox for robot cybersecurity”. In: *arXiv preprint arXiv:2010.07759* (2020).

[8] X. Wang et al. “Software exploitable hardware Trojans in embedded processor”. In: *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. 2012, pp. 55–58.

[9] Anton V Uzunov and Eduardo B Fernandez. “An extensible pattern-based library and taxonomy of security threats for distributed systems”. In: *Computer Standards & Interfaces* 36.4 (2014), pp. 734–747.

[10] Punam Bedi et al. “Threat-oriented security framework in risk management using multiagent system”. In: *Software: Practice and Experience* 43.9 (2013), pp. 1013–1038.

[11] Wenjun Xiong and Robert Lagerström. “Threat modeling–A systematic literature review”. In: *Computers & security* 84 (2019), pp. 53–69.

[12] Adam Shostack. “Experiences Threat Modeling at Microsoft.” In: *MODSEC@ MoDELS* 2008 (2008).