Bachelor's project

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

Magnus Baugerud
Henrik Mathias Berg
Lars Olsnes Østmo-Sæter

Penetration testing from a software engineering perspective

A case study of two penetration tests.

Bachelor's project in Computer Engineering
Supervisor: Donn Morrison
May 2021

Magnus Baugerud
Henrik Mathias Berg
Lars Olsnes Østmo-Sæter

Penetration testing from a software engineering perspective

A case study of two penetration tests.

Bachelor's project in Computer Engineering
Supervisor: Donn Morrison
May 2021

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

NTNU
Norwegian University of
Science and Technology

Preface

This thesis concludes a three-year bachelor's degree in Computer Engineering at the Norwegian University of Science and Technology (NTNU).

The assignment was requested by Donn Morrison, and performed in cooperation with NTNU IT, who provided two systems for penetration testing. The task of finding systems to test began already in December of 2020, and the project was started in January 2021 and was finished in May 2021.

We would like to thank Christoffer Vargtass Hallstensen, Tore W. Hermansen, and Per Sverre Wold at NTNU IT for working with us during this project.

Lastly, we want to give a special thanks to our supervisor, Donn Morrison. He has been available throughout the project and has given us excellent guidance.

Magnus Baugerud	19.05.2021 / Trondheim
Magnus Baugerud	Date/Place
Henrik M. Berg	19.05.2021 / Trondheim
Henrik Mathias Berg	Date/Place
[illegible]	19.05.2021 / Trondheim
Lars Olsnes Østmo-Sæter	Date/Place

Assignment

The initial assignment was to perform a penetration test on a system which would be chosen by us, the students, with guidance from our supervisor. We were free to choose any type of system to test, with the only requirements being that our supervisor had to deem it a suitable target, and the owner of the system had to agree to it being tested. We made two significant changes to the assignment.

Firstly, we chose to test two systems instead of one. This is because NTNU IT, the product owner we ended up working with, had two smaller systems they wanted tested. These systems are called Fotoboks and ePay.

Secondly, we decided that the focus of the bachelors' thesis would be shifted from performing penetration tests, to discussing research questions related to the experience of performing penetration tests. The project still included testing both systems and reporting the findings to the client in the form of technical reports.

The scope of the penetration tests, which will be performed through the course of this project, is described in the document called Statement of Work. This document, Attachment A: Statement of Work, is signed by the students, the supervisor, and NTNU IT.

Abstract

In this project we have performed penetration tests on two of NTNU's systems, Fotoboks and ePay. Fotoboks is a web application and ePay consist of both a web application and a web API, but the tests performed in this project focused primarily on the web application. In this report we will try to answer two research questions related to the experience of performing penetration tests for the first time. The questions are as follows:

1) How does prior experience affect the quality of a penetration test?
2) How would a software developer benefit from performing a penetration test?

The penetration testing was performed as described by the Penetration Testing Execution Standard, or PTES for short. To guide us through the testing we have utilized the OWASP Web Security Testing Guide, which is a detailed guide to the key phases of PTES. To conduct the penetration tests, we have utilized tools such as Burp Suite's proxy and repeater, as well as automated tools like Nikto, Nmap and ZAP. We have also used a variety of other similar tools and created our own tools using Python.

The two tests were performed one after the other and both systems were also revisited once. In the first testing round, which was the testing of Fotoboks, there were several tests that we did not understand, or that were done incorrectly. But for each round of testing our understanding and skill seemed to improve, and we performed the tests better and found more vulnerabilities. We also performed some testing on the ePay API, but the methods used to test the web applications could not easily be applied when testing the API, so we chose to focus on the web applications.

From our experience, performing a penetration test without prior experience leads to a result of significantly poorer quality than a penetration test performed with experience. The experience gives a better understanding of the tests and how tests are connected, which results in a better coverage. A penetration test performed with no prior experience is more likely to miss vulnerabilities. Understanding the individual tests in the OWASP Web Security Testing Guide is not enough to ensure coverage, as an understanding of the connection between tests is needed. Although our supervisor gave us valuable guidance and the testing guide provided us with detailed descriptions, the deeper understanding of penetration testing had to be gained through experience.

We believe there are multiple benefits in performing penetration tests for software developers. The experience they gain from a penetration test can be used to simulate real attacks while their system is still in development. They will also have a better understanding of how attackers approach systems, which may help them when securing their future systems. Software developers may also gain increased awareness of security from performing a penetration test, which can lead them to prioritize security throughout the development cycle.

Contents

Preface i

Assignment ii

Abstract iii

1 Introduction 1

1.1 Acronyms and Abbreviations 2

2 Theory 3

2.1 Penetration Testing Introduction 3

2.2 PTES 3

2.2.1 Pre-engagement Interactions 3

2.2.2 Intelligence Gathering 3

2.2.3 Threat Modeling 4

2.2.4 Vulnerability Analysis 4

2.2.5 Exploitation 4

2.2.6 Post Exploitation 4

2.2.7 Reporting 4

2.3 Statement of Work 5

2.4 Technical Report 5

2.4.1 Executive Summary 5

2.4.2 Threat Modelling 5

2.4.3 Observations and Recommendations 5

2.4.4 Risk Assessment 6

2.5 OWASP 6

2.5.1 Web Security Testing Guide 6

2.5.2 Risk Rating Methodology 6

3 Methodology 9

3.1 Penetration Testing Methodology 9

3.1.1 Statement of Work 9

3.1.2 OWASP Web Security Testing Guide 9

3.1.3 Technical Report 9

3.2 Distribution of Workload 11

3.3 Testing Tools 11

3.3.1 Automated Tests 11
3.3.2 Manual Testing 11
3.3.3 Python Scripts 12
4 Results 13
4.1 Fotoboks Tests 13
4.2 ePay 13
4.3 ePay Technical Report 14
4.4 Fotoboks Technical Report 14
4.5 Issues 14
4.6 Progress Plan and Time Usage 15
5 Discussion 18
5.1 Testing Evaluation 18
5.1.1 Issues 19
5.1.2 Progress Plan 19
5.2 Professional ethics 20
5.3 Penetration Test Quality 21
5.4 Software Developer Appliance 21
5.5 Teamwork Evaluation 22
6 Conclusion 23
7. References 24
8. Attachments 27
Attachment A: Statement of Work 27

1 Introduction

The team was tasked by Donn Morrison in cooperation with NTNU IT to perform penetration tests on two of NTNU's systems. The project resulted in a technical report for each system that detailed the findings. The systems in question are called Fotoboks and ePay. Fotoboks is a web application for ordering access cards for NTNU without having to take the photo physically on campus. It was created in 2020 to help infection prevention related to the COVID-19 pandemic. ePay is a service for reporting sales in NTNU's web shops. The service was created 13 years ago and was phased out during this project, which was in spring of 2021. ePay consists of an API and a web application. In addition to performing the penetration tests, the team has produced this thesis which discusses the experiences made performing penetration tests with no prior experience in the field.

The team followed the OWASP methodology when performing the penetration tests. The systems are web applications, and therefore the OWASP Web Security Testing Guide [1] was central. This book outlines how a penetration test should be performed on a web application when using the OWASP methodology, and it provides detailed information on tests that should be carried out. ePay also has a web API, but the team focused on testing the web applications in this project.

The project was centered around performing the penetration tests, and then reviewing the team's performance and obtained experience. The performance and the experiences made have been used to discuss two research questions, which are as follows:

1) How does prior experience affect the quality of a penetration test?
2) How would a software developer benefit from performing a penetration test?

The goal of this thesis is to explore and answer these questions. Answering the first question will provide insight into the process of performing a penetration test on a web application as software developers with no prior experience. Answering the second question will provide an overview of benefits software developers might gain from performing penetration tests. To properly answer the first question, the quality of the penetration testing must be evaluated in relation to the team's experience. To answer the second question, it will be discussed how the experience gained throughout this project could benefit the students in potential software developer careers.

This thesis will provide theory on penetration testing in chapter two, followed by a chapter describing the methodology used by the team when performing the penetration tests. The fourth chapter will present the results, which are the observations and experiences made by the team when performing penetration tests. The results will be discussed in the fifth chapter and a conclusion will be drawn in the sixth and final chapter.

1.1 Acronyms and Abbreviations

Following are some acronyms and abbreviations used in this document.

OWASP	Open Web Application Security Project	A nonprofit foundation that works to improve the security of software and web applications. **[2]**
API	Application Programming Interface	"An API is defined as a specification of possible interactions with a software component." [3]
PTES	Penetration Testing Execution Standard	A standard consisting of seven main sections designed to provide a common language and scope for performing penetration testing. [4]
Pentest	Penetration Test	A penetration test is "the simulation of an attack on a system, network, piece of equipment or other facility, with the objective of proving how vulnerable that system or "target" would be to a real attack." [5, p. 14]
SoW	Statement of work	An agreement between a client and a service provider to describe a project's work requirement and to define project-specific activities. [6]
NDA	Non-disclosure Agreement	"A Non-Disclosure Agreement (NDA) is a legally enforceable contract that establishes confidentiality between two parties—the owner of protected information and the recipient of that information. By signing an NDA, participants agree to protect confidential information shared with them by the other party. In addition to not divulging or releasing the information without consent, the recipient also agrees not to copy, modify, or make use of the information in any way that is not authorized by the owner." [7]
ACM	Association of Computer Machinery	ACM is the world's largest computing society, aiming to strengthen the profession's collective voice. This is done through strong leadership, promotion of the highest standards, and recognition of technical excellence. [8]
SSO	Single Sign-On	Single sign-on allows a user to sign in to one application and be automatically signed in to other applications. It allows users to have one set of credentials for multiple applications. [9]

2 Theory

2.1 Penetration Testing Introduction

Penetration testing, also called ethical hacking, is the process of testing a system to find vulnerabilities in its security, the same way a hacker would. The difference is that a penetration test is authorized by the owner of the system and it is not performed for the testers personal gain, but to evaluate the security of the system being tested. A penetration test can be performed on networks, applications, and operating systems, as well as other types of systems. [10]

A penetration test can be black box, white box, or grey box. Black box means the penetration testers are given little or no information about the system. This type of penetration test most closely simulates an attack from an outside source, as the testers must acquire information needed to attack the system themselves. White box means that the testers are given information about the system. This can be information on how the system is built and what technologies it uses, or even access to source code. In this case the testers have additional information compared to what a normal attacker would have, which makes it easier to find potential vulnerabilities. If testers are given some information on the system, but not all, the testing can be defined as grey box. [10]

Penetration testing is an important technique for ensuring the security of an application. However, it should not replace other types of testing that are usually used throughout a development cycle. In software development testing should be a part of the process from the beginning, but penetration testing is a technique best suited for later stages of development. [1, p. 20]

2.2 PTES

The Penetration Testing Execution Standard, or PTES, is a standard consisting of seven main sections designed to provide a common language and scope for performing penetration tests. The seven sections cover everything from pre-engagement interactions to reporting. [4]

2.2.1 Pre-engagement Interactions

The aim of this section is to present available tools and techniques, get an overview of the system, and to work out an agreement between the penetration tester and the client. An important part of this agreement is the scope. The scope gives permission to test and defines what will be tested, including IP ranges to test and rules of engagement. The agreement must include details on deliveries and time estimation, describing what will be delivered and when. It must also include payment terms, details on how to deal with third parties, define lines of communication and set goals for the project. [11]

2.2.2 Intelligence Gathering

This section includes the intelligence gathering activities. The goal is to gather information about the target system and map the system architecture. This is done through identifying and fingerprinting the target systems different components. The more intelligence gathered during this phase; the larger the attack surface will be in the following testing phases. The intelligence gathering might also reveal attack vectors that does not fall under the scope that was agreed upon during the pre-engagement interactions. If any such out of scope attack vectors seem relevant, the testers should contact the client about a potential revision of the scope. [12]

2.2.3 Threat Modeling

PTES defines a threat modeling standard that should be used in a penetration test. The standard has two key elements, which are assets and attackers. Assets are broken down into business assets and business process, and attackers are broken into threat agents and threat capability. These four elements need to be identified and analyzed. The goal of analyzing business assets is to determine which assets are the most likely to be attacked and analyzing business processes is done to determine which processes are critical and non-critical based on the value they generate. All potential threat agents must be identified, and they should also be classified as either internal or external threats. Finally, analyzing the threat capability of threat agents will give an overview of which agents are most capable of performing a successful attack. [13]

2.2.4 Vulnerability Analysis

The vulnerability analysis stage is when the testers start testing the system for issues that might compromise the target system or its users. The nature of the vulnerabilities can vary from insufficient input validation to misconfigurations on the server. Any kind of security vulnerabilities can be tested for here, but it is usually limited by the scope of the penetration test. It is important that the testers test as thoroughly as agreed upon, but also that they do not pass the boundaries defined by the scope. Test fall under two categories. The first category is active tests, which have direct interaction with the component being tested. The second category is passive tests, which mainly consist of metadata analysis and traffic monitoring. It is also relevant to do research on identified components to find publicly available exploits. [14]

2.2.5 Exploitation

This phase focuses on gaining access to systems or information by bypassing restrictions or exploiting other vulnerabilities. The vulnerability analysis directly shapes the exploitation phase as it is the exploitation of findings from the analysis. From the found vulnerabilities the testers find the paths with highest gain and of least resistance into the system. After this the tester will monitor to see if there are any countermeasures in place and see whether any of these countermeasures might be circumvented. Countermeasures may mean anti-virus software, intrusion detection systems or any other method to limit the impact of a security breach. The best way to test for exploitation is generally to perform actual attacks on the system, how elaborately this phase is to be performed is therefore highly dependent upon the scope of the penetration test. [15]

2.2.6 Post Exploitation

The purpose of the section is to get an overview of compromised systems and compromised information. The goal is to determine the value of the findings, which depends on the sensitivity of exposed data and if the exposed information can be used for further exploitation. Risk assessment of findings is an important part of the final report. [16]

2.2.7 Reporting

The reporting is broken down into two major sections: a technical report and an executive summary. The technical report will include details on the different steps taken through the penetration test, while the executive summary communicates the high-level findings. The technical report may contain the executive summary. The reporting should be outlined in the agreement from the pre-engagement interactions section, where it should be described what is to be reported and when the report should be delivered. Findings are often released to the public after a time period specified in the agreement from the pre-engagement interactions section. It is important to handle disclosure as agreed with the client. [17]

2.3 Statement of Work

A statement of work, often called an SoW, is not exclusively used in penetration tests. In general, it is an agreement between a client and a service provider to describe a project's work requirement and to define project-specific activities. [6]

In the context of a penetration test the SoW will describe the scope and deliveries as well as timeframe, location of work, price, and payment schedule. The scope is where the client defines which system is to be tested and set boundaries on the testing team; typically defining which related systems should not be touched and which kind of tests should not be attempted. Normally the client presents a list of IP addresses they give consent to have tested. The SoW will specify what deliveries the service provider will present to the client. In penetration testing the service provider usually reports to the client in the form of a technical report. How to handle disclosure of the findings should be addressed as well, and the client may ask the service provider to sign an NDA before testing. The SoW will be signed, and it gives the service provider formal permission to test the system in accordance with the agreed upon scope and rules of engagement. [18] [19]

2.4 Technical Report

The technical report is the main product of a penetration test. This is the document where the penetration testers describe the findings, as well as supply documentation on how tests were performed.

A technical report will typically have the following form: executive summary, threat modelling, observations and recommendations, and risk assessment. [20]

2.4.1 Executive Summary

The executive summary will communicate at a high level the goals of the penetration test and all the findings, as well as an evaluation of the general security of the system. This section is not necessarily meant for the developers of the system, but rather for everyone who is impacted by the identified issues. This usually means company executives or employees with responsibility for the company's security. [21]

2.4.2 Threat Modelling

Threat modelling is described in more detail in chapter 2.2.3. Threat modelling is necessary to view the report in proper context. Any critical issues in a system with very few threat actors, should not necessarily be prioritized over less critical issues in a system that is more exposed to malicious actors. Issues should also be prioritized this way based on the amount of critical assets they compromise. [13]

2.4.3 Observations and Recommendations

Observations and recommendations are mostly intended for the maintainers of the system. The testers will give detailed descriptions and explanations for all the security issues that were found during the penetration test, along with recommendations for how to fix these issues.

2.4.4 Risk Assessment

Risk assessment will often be incorporated in observations and findings as each risk assessment would correspond to a specific issue. The goal of risk assessment is to prioritize vulnerabilities based on the threat they pose to the system. The issues with the highest risk rating should be addressed first. [20, pp. 37-48]

2.5 OWASP

The Open Web Application Security Project, OWASP, is a non-profit foundation focused on improving the security of software and web applications. To achieve this the foundation offers resources to develop secure applications, but more importantly for this project they offer resources for penetration testing, including methodologies and guides. [2]

2.5.1 Web Security Testing Guide

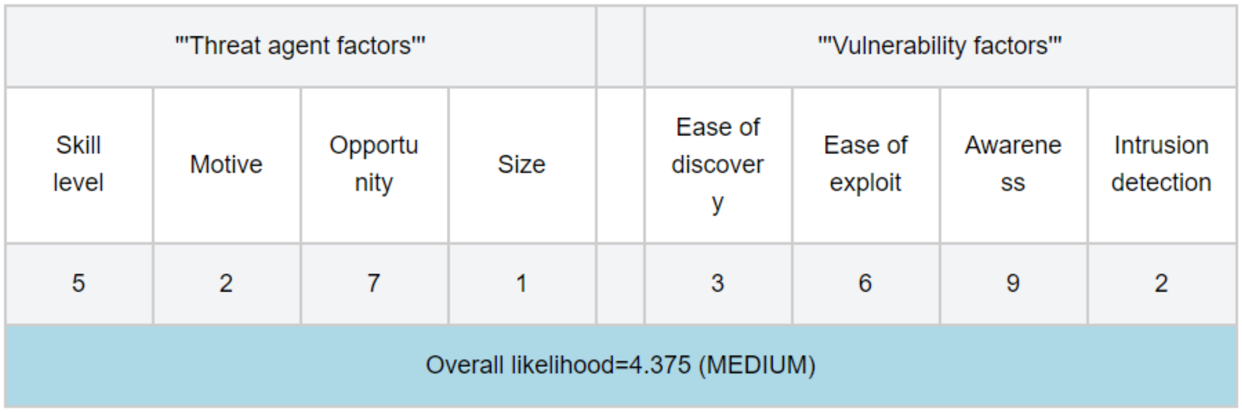
The Web Security Testing Guide is created by security professionals and volunteers and is published by OWASP. It provides detailed information on tests that should be performed when testing a web application. The tests described in the guide are divided into 12 chapters, each representing a category of tests. The chapters are listed below. [1]

1) Information Gathering: Tests for identifying the attack surface.
2) Configuration and deployment management testing: Tests checking for misconfigurations.
3) Identity management testing: Tests checking for proper implementation of role definitions and permissions.
4) Authentication testing: Tests checking for proper implementation of authentication.
5) Authorization testing: Tests checking if authorization can be bypassed or if privileges can be escalated.
6) Session management testing: Tests that check if sessions are properly implemented and secured.
7) Input validation testing: Tests checking if user input fields are properly sanitized.
8) Error Handling: Test checking if errors are properly handled and if error pages give away any information.
9) Cryptography: Tests checking if sensitive information is encrypted and if the encryption is sufficiently strong.
10) Business Logic testing: Tests that check if the logic or workflow can be circumvented.
11) Client-side Testing: Tests checking for vulnerabilities that can be exploited on the front-end.
12) API Testing: Tests for GraphQL APIs.

2.5.2 Risk Rating Methodology

The OWASP Risk Rating Methodology is a basic framework meant to be customized for the organization. The goal is to find the overall risk severity of a vulnerability from two scores: the likelihood and the impact. These scores are calculated using specific factors, which are scored on a scale from 1 to 9 where

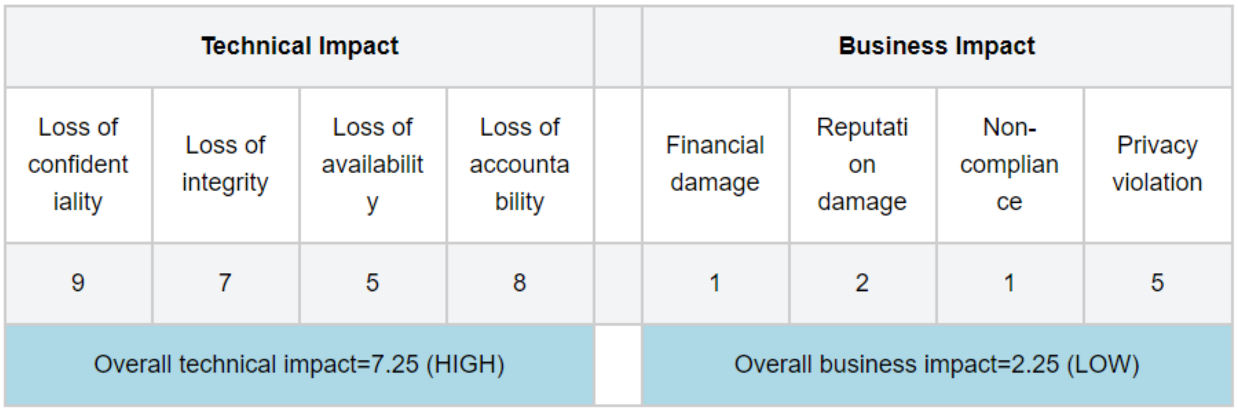
1 is least severe and 9 is most severe. The likelihood score is based on the average of threat agent factors and vulnerability factors. See Figure 1. [22]



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

Figure 1: An example of how to calculate a likelihood score with the OWASP Risk Rating Methodology. [22]

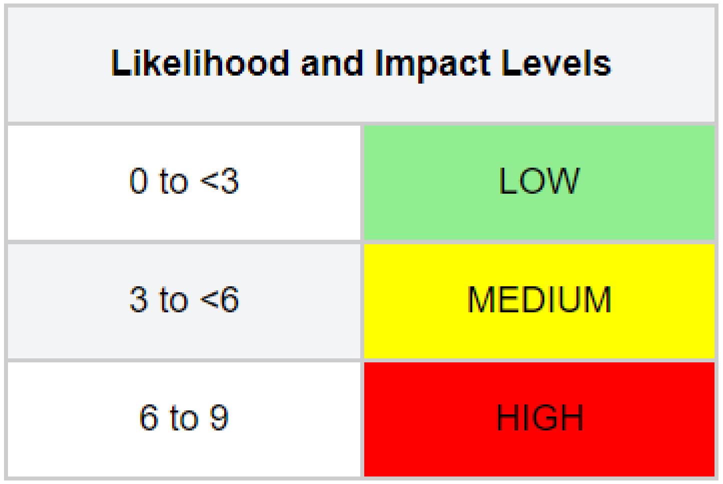
The impact score is based on either technical impact or business impact. If the information about business impact is sufficient, then the impact should be based on the business impact score. If not, the impact score should be based on the technical impact score. Both scores are calculated by taking the average of their factors, see Figure 2. [22]



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

Figure 2: An example of how to calculate technical and business impact scores with the OWASP Risk Rating Methodology. [22]

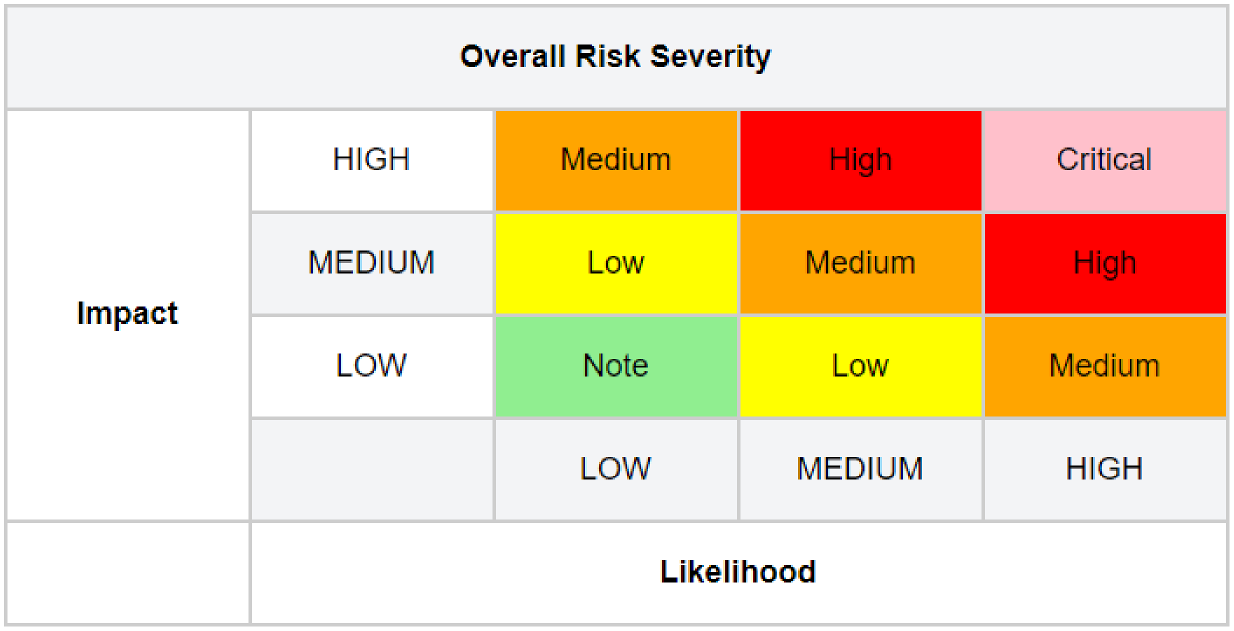
The likelihood score and the impact score are then converted to a severity of either LOW, MEDIUM, or HIGH using the chart shown in Figure 3. [22]



Likelihood and Impact Levels	
0 to <3	LOW
3 to <6	MEDIUM
6 to 9	HIGH

Figure 3: Chart used to convert from numbered score to LOW, MEDIUM, or HIGH. [22]

The overall risk severity is found using the likelihood and impact scores in the chart shown in Figure 4: Chart used to calculate the overall risk severity score with the OWASP Risk Rating Methodology. Figure 4. [22]



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

Figure 4: Chart used to calculate the overall risk severity score with the OWASP Risk Rating Methodology. [22]

3 Methodology

3.1 Penetration Testing Methodology

The team decided to follow PTES [4] when performing the penetration tests. This standard was chosen because it provides a clear structure as to how a penetration test should be performed. It divided the process into seven main sections, which are detailed in chapter 2.2. PTES was chosen because of the structure it provides. The team did not have any prior experience with penetration testing; therefore, a clear structure made the task easier.

3.1.1 Statement of Work

As part of the pre-engagement phase of the penetration test the team decided to write a SoW that was signed by NTNU IT, the students and the supervisor. The reason for making a SoW was that this kind of document allows for defining a precise scope of work and will cover many requirements for the pre-engagement phase. The scope is important to clearly define what the testers will and will not do to test the systems in question. The SoW also states what deliveries will be produced from the project.

3.1.2 OWASP Web Security Testing Guide

As none of the team members had any prior experience with penetration testing, the team chose to use the OWASP Web Security Testing Guide [1] to perform the intelligence gathering, vulnerability analysis and exploitation sections of PTES. This guide provides detailed information on all tests that should be performed on a web application.

The team used a checklist [23] that was based on the OWASP Web Security Testing Guide. The checklist helped the team keep track of which tests had been done, which tests had passed, which had failed, how they were tested, and what was found. Each test in the checklist also had a code which corresponded to the code of a test in the guide. Some updates were made to the checklist because it was based on version 4.0 of the testing guide, while the team used version 4.2.

3.1.3 Technical Report

To report findings the team wrote technical reports including executive summaries as described in chapter 2.4. These technical reports also included risk assessment and covered both the post exploitation phase and reporting phase of the PTES. The team decided to also include detailed test results, which would supply documentation for each test that was performed. The documentation would describe each test as detailed as is necessary to understand the test results. Detailed test results are valuable because they clearly communicate the test coverage and allow the client to see how tests were performed.

The team decided to base the risk assessment on the OWASP Risk Rating Methodology [22]. Each finding was given a likelihood score and an impact score, and an overall risk severity score was calculated using the model pictured in Figure 4 from chapter 2.5.2. The team's risk assessment differs from the OWASP Risk Rating Methodology in what the different levels, low to high, are defined as. While the OWASP Risk Rating Methodology calculates these values using threat agent-, vulnerability-, technical impact-, and business impact factors, the team made more general definitions to apply the risk assessment more easily to the varying types of vulnerabilities found. These definitions are based on the technical impact of the vulnerabilities. There are two reasons for this. Firstly, the team did not have sufficient information on the business impact of Fotoboks. Secondly, ePay is no longer in use, and therefore has no business impact. The general definitions can be viewed in Table 1, Table 2 and Table 3.

Likelihood Score

High	It is very likely that the vulnerability will be used by an attacker, due to it being easy to find and easy to use.
Moderate	It probable that this vulnerability will be used, but it is harder to find and/or exploit.
Low	Due to being very difficult to either find or exploit it is not likely that this vulnerability will be used by an attacker.

Table 1: Overview of likelihood scores and color codes.

Impact Score

High	May totally or partially compromise system.
Moderate	May compromise some users or smaller parts of the system.
Low	May result in compromising less important parts of the system, cannot compromise other users on its own.

Table 2: Overview of impact scores and color codes.

Overall Risk Severity

Critical	Must be fixed as soon as possible. It is very likely that this vulnerability will be exploited, and it will partially or totally compromise the system.
High	Should be fixed as soon as possible. This vulnerability will significantly compromise the system. It is harder to exploit, or it causes less damage than a critical issue.
Moderate	Should be addressed. The impact may be high, but then the likelihood will be low, and vice versa.
Low	Has the possibility to cause problems. Due to it having either a low likelihood or impact means it is not always a real issue.
Note	Should be looked at, but there is no immediate danger.

Table 3: Overview of risk severity scores and color codes.

3.2 Distribution of Workload

The team members were responsible for completing and submitting the agreed upon deliveries, a technical report for each system including executive summaries, within the timeframe of the project. Therefore, the team developed a progress plan to test and document the findings from both Fotoboks and ePay.

All team members did the same nature of work and distributed the work evenly. All documentation was performed as a group, and the penetration testing was distributed by individually performing the next test in the OWASP Web Security Testing Guide until all tests were performed. During meetings Magnus Baugerud was tasked with writing minutes of the meetings, and the other members, Henrik Mathias Berg and Lars Olsnes Østmo-Sæter, were tasked with leading the meetings.

3.3 Testing Tools

The team decided to perform the tests mostly using Kali Linux virtual machines. Kali Linux is an operating system utilizing the Linux kernel. The team utilized Kali by using VirtualBox images [24], and the Kali Linux app [25] for Windows 10 using Windows Subsystem for Linux. The reason for using Kali Linux is that it is specifically built for security testing, and it also has a lot of pre-installed tools that are essential for penetration testing.

3.3.1 Automated Tests

Several tools were used to conduct automated tests. Automated tools are useful for performing tests such as scans, which are slow and tedious to perform manually. However, any vulnerabilities an automated tool reports should be checked manually, as the tool may give false positives. There is also a limit to what type of vulnerabilities automated scans can find, so they should only be used in addition to manual testing, not as a replacement. During the project, the team used Nmap, Wappalyser, Gobuster, DirBuster, Nikto and ZAP.

Nmap [26] is a well-known tool for scanning networks. It can be used to scan open ports on a target machine, which is useful for discovering attack surfaces.

Wappalyser [27] is a browser extension used to identify which frameworks and technologies a web application uses. This gives the testers extra intel about the application during the information gathering phase.

Gobuster [28] and DirBuster [29] are tools used for fuzzing endpoints. Both work by using a wordlist to send requests and analyzing the response code. They can be used for the same purpose, but the team chose to mainly use DirBuster because of its ease of use.

Nikto [30] and ZAP [31] are web app scanners and check for outdated server versions, missing security headers and some version specific problems, in addition to much more. They are non-stealthy tools that generate reports with the results after fuzzing the server.

3.3.2 Manual Testing

Most of the testing was done manually, and several tools were used for this testing.

Burp Suite [32] is a penetration testing tool with many functionalities. One of the most useful functionalities Burp Suite offers is to set up a proxy server that all browser traffic is routed through. This allows the user to see all requests from the browser and responses from the server in plain text, even if the connection is encrypted. One can also change requests intercepted with Burp Suite. This tool is useful for seeing how the client and server communicates, and to change requests to find potential vulnerabilities. Burp Suite also offers automated tools, but most of them are only available in the paid versions. In this project the Community version was used, which is the only free version. ZAP offers similar functionality, but Burp Suite was chosen for manual testing because the team had some prior experience with the application. Zap was still used for automated testing.

Wireshark [33] is used for analyzing network traffic at all network layers. It is the most widely used program for this purpose and is prominently used in education contexts. The students are therefore familiar with using this program. An alternative to Wireshark that was used to a lesser extent is Fiddler [34]. Unlike Wireshark, Fiddler is a web proxy that analyze network activity mainly on the application level, making it easier to evaluate the HTTP requests and responses.

Telnet [35] is a command line tool that communicates directly over TCP. This can be useful for testing open ports, and it allows you to craft your own HTTP requests. SoapUI [36] and Postman [37] are mainly used for sending requests to APIs. SoapUI was used to test the ePay API, while Postman was used to send POST requests to Fotoboks.

3.3.3 Python Scripts

Python is a general programming language that the team used to create several scripts for testing during the project. Using the Requests library [38], Python can be used to automate communication with web servers through HTTP. Using the Python library Beautiful Soup [39], Python may extract data from HTML and XML files. Using them together is particularly useful for fuzzing input fields or simulating normal usage in a web application such as login and logout.

4 Results

Fotoboks was tested before ePay, and after finishing the tests for both systems, the team wrote the technical report for ePay and iterated once more through the tests to confirm and explore test results. Lastly, they wrote the technical report for Fotoboks and iterated through Fotoboks' tests as well.

4.1 Fotoboks Tests

The categories defined by the OWASP web security testing guide [1] was done in the order they were listed, which means information gathering was the first group of tests the team performed. The team had difficulties performing these tests, because they mainly consist of gathering information, but the team did not know what they were looking for. The first time the team tested Fotoboks, some of the tests were not performed properly. WSTG-INFO-10, for example, was not performed at all. Since the team did not know exactly what to look for during information gathering, some information had to be filled in during later tests.

The team relied heavily on the testing guide [1] during the testing of Fotoboks. They had little knowledge of how the tests should be performed, so a lot of time was spent reading the guide and researching the tests using other sources.

As the testing of Fotoboks progressed some tests had to be revisited, since the team gained more experience and therefore had a better understanding of the application. For example, it was initially believed that the session cookie missing the secure attribute was not an issue, because the application has the HSTS header set. However, the team later understood that there exist edge cases where a lack of the secure attribute can be a vulnerability.

After finishing the initial testing for Fotoboks, the team compiled all the details they had on the vulnerabilities they had found. They also risk assessed the vulnerabilities before moving on to ePay.

4.2 ePay

The ePay web application tests worked as a repeat of the Fotoboks tests on another target. Each test that was performed on Fotoboks was also performed on ePay, and the team still reviewed the chapter in the OWASP Web security Testing Guide [1] for each test. Due to having performed the tests once before both reading the guide and performing the tests took less time.

Although ePay consisted of both an API and a web interface the students decided to focus on the web interface as the testing guide was not very suited to test the API. The tests described in the guide was not performed on the API, but the automated scans used on the ePay web interfaces also covered the API.

The automated scans revealed some vulnerabilities that the team would likely miss by only performing manual tests. For instance, Gobuster located the console interface endpoint for ePay's application server, revealing a severe server misconfiguration. The scans also revealed new attack vectors for the team to explore, and the team wrote scripts to attempt brute force login on the console interface. Since the automated tests revealed issues that had not been found through manual testing, the team decided the scans would be run on Fotoboks when it was revisited.

4.3 ePay Technical Report

The technical report for ePay was started after all the tests had been performed. The technical report includes details on how every test was performed and what was found. When writing descriptions for these tests, the team realized some tests had to be revisited. This led to vulnerabilities being found that were not found the first time tests were performed. For example, it was not discovered that one could get unauthorized access to the admin interface or perform CSRF attacks until tests were revisited for the technical report. During the second round of testing, it was also found that the cache control was not implemented correctly.

When revisiting tests, the team also realized some findings were inaccurate. It was first believed that the ePay session timed out after 5 hours, but after further testing it was found that the session can be kept alive seemingly indefinitely if it is not left idle.

The risk assessment had to be reworked several times during the development of the technical report. The team realized that they had misunderstood the impact of the exposed Oracle console vulnerability. Initially, they rated the impact of this vulnerability based on the information revealed about the server. However, the team realized that the exposed console login, which enabled brute force attacks, had a more severe impact. The iframe clickjacking vulnerability was also changed. Initially, it was rated as moderate but was increased to high, because the team deemed Feide credentials to be more sensitive when revisiting the rating.

4.4 Fotoboks Technical Report

The technical report for Fotoboks was written after the technical report for ePay. When the team went through the details on tests performed, they revisited a lot of tests that were either skipped or they found lacking. The experience gained from testing ePay gave the team a better understanding of the tests, which led to the team finding new vulnerabilities when revisiting them. Initially the team concluded that both the logout functionality and the cache control had no issues, but the second time around the team found faults with both. The logout functionality must invalidate session cookies, but it only invalidated the SSO session, not the session for Fotoboks. The cache control was incomplete, and the session cookie was stored in some caches, which it should not be. The team also found that the session cookie was not valid for 8 hours as initially though, but for 24 hours. All these findings were overlooked or not tested thoroughly enough the first time around.

The team also ran more automated tests on Fotoboks while writing the technical report, including ZAP [31], Nmap [26], and DirBuster [29], which exposed additional vulnerabilities.

4.5 Issues

The team had some issues understanding the test described in the OWASP Web Security Guide [1] under the code WSTG-ATHN-06. The chapter describes how to test for incomplete cache-control. The test was deemed a pass, as the students found cache-control on the logout functionality, but as later revealed, both the session cookie and other potentially sensitive information was saved in some caches. Further understanding of cache-control obtained while revisiting ePay tests revealed new vulnerabilities for both ePay and Fotoboks.

There were multiple other tests that the team was unsure of, and therefore they are less confident in the results. These tests include many injection tests under data validation, and other tests for padding Oracle, DOM based cross site scripting, and JavaScript execution.

Some tests were set to non-applicable on the first iteration of testing but were tested on the second iteration after the team got a new understanding of the tests. This understanding also led to some tests that were passed on the first iteration being deemed non-applicable when revisiting them. This happened mainly on Fotoboks, but also occurred while testing ePay.

The team ended up deciding that the ePay API would not be tested thoroughly. Performing some basic tests on the API did not yield any results, as a vulnerability would require a bypass of authentication. The team found that a more rigorous test on the admin interface would be more productive and yield a better report for the client. The OWASP Web Security Testing Guide [1] does not give information on how to test APIs, except for GraphQL APIs.

4.6 Progress Plan and Time Usage

A progress plan was first developed in week 2. However, in the original plan Fotoboks and ePay would be tested at the same time, but the team encountered issues with ePay and focused on testing Fotoboks. The team realized it would be better to test one system at a time, so the plan was reworked in week 7, and the team continued to only test Fotoboks for the first stage of the project. The reworked plan was followed more closely than the original, but there were some deviations. The technical reports took 11 days longer to make than what was estimated, because time was spent revisiting tests. The initial testing of Fotoboks and ePay took 8 days less. It was also planned a week for attempting verification bypass on the ePay API, but as described in chapter 4.5 the team decided to focus on more thorough testing of the web interface. Because the team started earlier with testing the web interface, they could start the technical reports earlier. This means the technical reports were finished as scheduled even though writing the reports took much more time than expected.

It was estimated that each student would spend approximately 500 hours on this course, which means 1500 hours had to be distributed. In the SoW the team estimated that 750 hours would be used for testing the systems, including both researching and performing the tests. The students ended up using 447 hours, which is 303 hours less than estimated. The unused time estimated for testing was used for documenting the project progress in form of the project handbook, writing meeting agendas, writing technical reports, and finishing the thesis. It was estimated 150 hours for lectures, but only 36 hours was used. These hours were also redistributed to documentation, meetings, technical reports, and the thesis. The team could track how the hours were redistributed by reviewing the weekly reports in the project handbook, a document containing all project documentation.

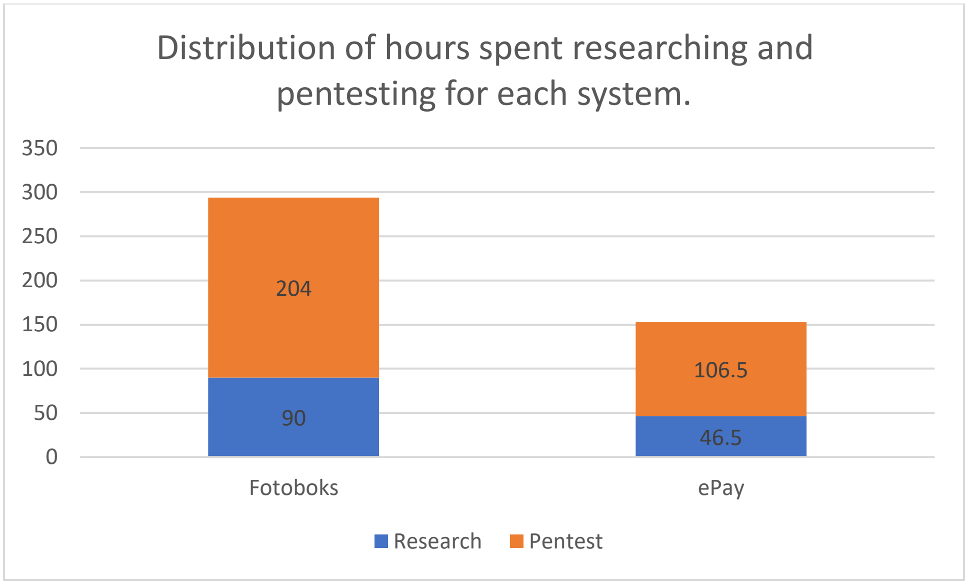


Figure 5: Total hours spent on testing each system.

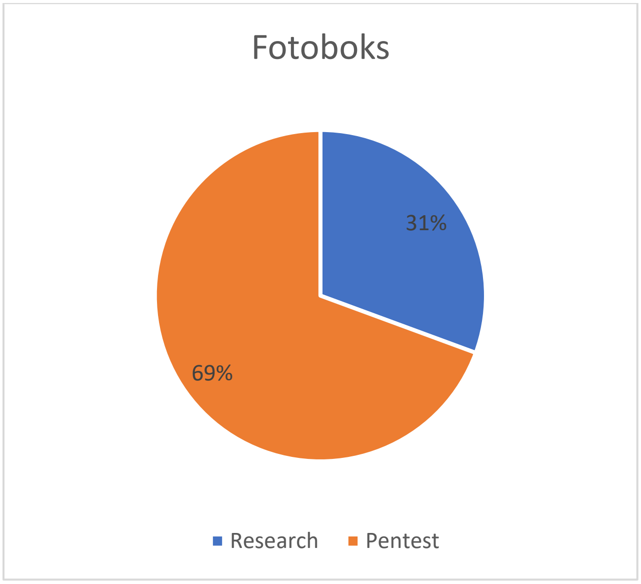


Figure 6: The ratio between work hours spent on research and testing for Fotoboks.

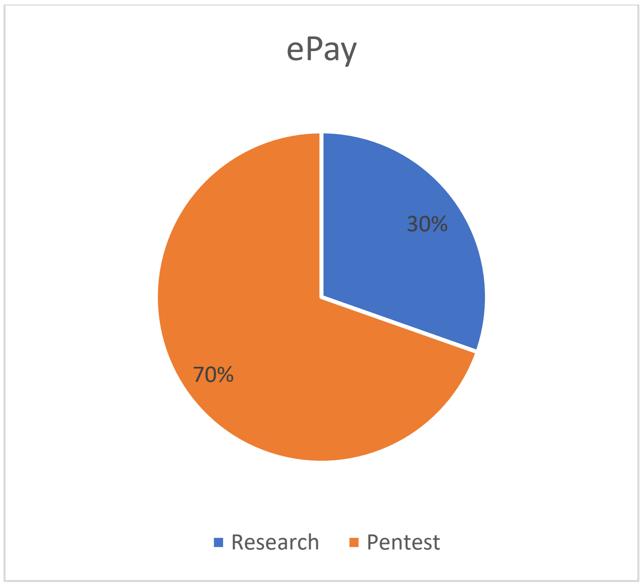


Figure 7: The ratio between work hours spent on research and testing for ePay.

Figure 5 shows that testing for ePay took approximately half as long compared to Fotoboks. Figure 6 and Figure 7 shows that the ratio between Penetration testing and research were almost the same for both systems.

5 Discussion

5.1 Testing Evaluation

During the initial testing of Fotoboks the team had problems understanding the objective of some tests. This came from the fact that the team had never received any training in penetration testing, nor did they have any experience with such tests. Relying heavily on the OWASP Web Security Testing Guide [1], the team gained knowledge on one type of test, and one phase of testing, at the time. The students still lacked a deeper understanding of the penetration testing process. Some tests, especially in the information gathering phase, were performed to make future tests easier. With no knowledge of these future tests the team had problems understanding what to look for during the information gathering phase. This resulted in some tests being delayed or being performed multiple times as the teams' understanding grew.

When the team had been through all the tests on Fotoboks they started testing ePay. The testing of the ePay web interface was finished almost twice as fast as the testing of Fotoboks. This is likely because the team were familiar with the tests and knew what do to and what to look for when performing them.

Initially, ePay and Fotoboks were tested at the same time, but the team had difficulties testing the ePay API and chose to delay the testing of the entire ePay system. The OWASP Web Security Testing Guide [1] focuses primarily on web applications, which probably is why the team faced difficulties. If they were to test the API, it would have been necessary to spend time finding a different methodology or guide, which is why they chose to focus on Fotoboks first. The team later did some basic tests to try and bypass the authentication of the ePay API but stopped when they did not get any results. If the team had continued to test the API, they might have found something, but it could have taken up so much time that it would negatively impact the coverage of the ePay web interface and Fotoboks. Even if they were able to find vulnerabilities in the API, the team could not have been sure if everything had been covered, unlike Fotoboks and the ePay web interface where the testing guide [1] lists all the tests that should be performed on them. Finding vulnerabilities is not the only important part of a penetration test. Good test coverage is also important, because it gives the client confidence that their system is secure.

ePay was delayed because of difficulties, but the choice to delay it may have increased the quality of the testing. As mentioned previously, the students lacked a comprehensive understanding of penetration testing as a whole when testing Fotoboks. When the team started the testing of the ePay web interface, they had gained a better understanding of penetration testing, which likely contributed to increased efficiency of the ePay web interface tests.

Initially the team planned a few days to write the technical reports. The first part of the reports, where the findings were described and risk assessed, did not take more time than expected. However, the work on the technical report was delayed several times when the more detailed test results were written. When the team were documenting all the tests, they found many tests that were not performed correctly. Some of these tests were set as non-applicable, but some also needed to be redone because the team had misunderstood the tests. The number of tests that had to be repeated for ePay was lower than that of Fotoboks, likely due to an increased understanding of the penetration testing process. This also shows that the initial testing of ePay was not only more efficient but also of higher quality, compared to the initial testing of Fotoboks.

If the project were to be repeated the team would plan to use more time for writing the technical reports. If no tests had to be revisited while writing them, they could possibly have been finished in time, but the few days designated to these reports were not enough time to write a good report. Luckily for this project, the team was well ahead of schedule after performing the penetration tests, which allowed them to take the time to quality assure each test. It is not certain that the new issues found while revisiting tests would have been found if the team had simply spent more time on each test. The team believes that by revisiting the tests, while writing the reports, they revealed more issues than they would have if the tests were conducted only once. This is because performing the tests in iterations consistently gave the team more understanding of both the tests and the systems for each iteration of testing. Spending time to revisit tests while writing the reports would therefore be repeated if the team were to do the project again. However, this is not an approach that every penetration testing team necessarily would benefit from. A skilled and experienced penetration tester may only need one iteration of testing.

If the team were to do a similar project, they would continue to use the OWASP web Security Testing Guide [1]. It is a helpful tool that was used throughout the entirety of the project. The different categories provide a clear structure that can be followed when testing the systems. The high variety of tests also give a high test coverage, which gives the client more confidence that potential vulnerabilities have been found and increases the value of the technical reports.

When the team risk assessed the issues for Fotoboks, as described in the OWASP Risk Rating Methodology [22], they found that the risk ratings did not accurately prioritize the most serious issues. The definitions of likelihood and impact scores were changed so that the risk assessment would be more suited for the systems being tested. The team believes the issues are accurately rated with the custom score definitions, and that the overall risk severity prioritizes the most pressing vulnerabilities. However, some issues were hard to risk assess. In some cases, the likelihood or impact was difficult to determine, because the issue would affect another system, or that the issue depended on another system. The students had no prior experience with similar issues that they could base the rating on, which made the task even more difficult.

5.1.1 Issues

There were some test descriptions in the OWASP Web Security Testing Guide that the team had more difficulties understanding than others. One reason for this is that some tests simply are more difficult to understand and perform. For example, checking if the HSTS header is set is simpler than looking for a HTTP request smuggling vulnerability. Another explanation could be that some descriptions were written in a way that is difficult to understand for an inexperienced penetration tester. The guide is written by security professionals, not pedagogues, so it is reasonable to believe that they were not always able to write the descriptions in a way that is easy to understand for beginners.

5.1.2 Progress Plan

The team believes that the reworked progress plan was more suitable to perform the planned work. By splitting up the project to perform two penetration tests on different systems instead of one test for two systems, it became easier to track the progress of the work itself as well as the progress of the team’s ability to perform high quality tests. As discussed, performing more iterations of tests improved the quality of the final report. The team believes it was a good choice to change the approach early, and if the project were to be repeated the team would structure it like the revised progress plan.

When the team made the revised progress plan, they had not yet finished the testing of Fotoboks. However, they tried to estimate how much more time they needed for Fotoboks and how long ePay would take. It turned out the tests were performed much quicker the second time, which meant the team overestimated the time needed for ePay. The team thinks that the inability to accurately estimate how much time the tests would take is mostly due to inexperience, but even with experience the time usage can be unpredictable. The size and intricacy of each system was around the same in this project, with ePay being slightly larger. Other systems with more input fields, or more assets to protect might pose a challenge when trying to estimate time usage even if the testers have experience.

The ratio between testing and researching is almost identical for ePay and Fotoboks, even though almost twice as much time was spent on Fotoboks. There could be several explanations for this. One explanation is that the team's experience with the tests let them both read the guidance and perform the tests faster, with no considerable difference in ratio. However, the hours were logged at the end of the day, so the team did not always remember exactly how much time was spent on what. Therefore, another possible explanation is that the team fell into a pattern of logging approximately 70% of the hours on testing and 30% on researching, even if that was not accurate. Finally, the ratios being almost identical could be a coincidence.

5.2 Professional ethics

ACM, the Association of Computer Machinery, has made a Code of Ethics and Professional Conduct [40], which includes a list of seven general ethical principles. The principals relevant to the project are to avoid harm, be honest and trustworthy, respect privacy, and honor confidentiality. In the following paragraphs some ethical questions relevant to the principles listed above will be explored.

Did the team avoid causing harm while performing the penetration test? The two systems the team tested were Fotoboks and ePay. The team tested development instances of both systems, running in developer environments, as opposed to testing the production instance. Therefore, the systems had no users that could be harmed during the testing, and breaking the system would not lead to any considerable damage.

Was the team honest and trustworthy when describing the testing and findings in the technical reports? The team described every test and finding in the technical reports. The technical reports are honest, as they describe the project to the best of the team's ability. When evaluating whether the reports are trustworthy there is one more factor to consider. Since the students have no prior experience with penetration testing, do their reports cover as much as they believe? The client, NTNU IT, has continuously been in contact with the team throughout the project, and were informed of the students' lack of experience from the beginning. The reports, on the other hand, does not communicate this lack of competence clearly, and therefore a 3rd party reading the reports may put more trust in the tests than what is advisable.

Did the team respect the privacy of the systems' users? The systems that were tested ran in developer environments, which meant there were no one using the systems. Therefore, the team did not have to worry about user privacy. Information that could be used to compromise users' privacy was found while testing ePay, but the information was on dummy users.

How does the team honor confidentiality regarding the penetration testing? Before the team started testing the client's systems, every student signed an individual non-disclosure agreement. These contracts are legally binding and assures that the student does not breach confidentiality. The agreement ensures confidentiality regarding technical specifications and methods, as well as maintenance and business conditions with impact on privacy, security, or the business' competitiveness. The contracts also specifies that all findings may be published, but that the client may delay the publication for up to 5 years.

5.3 Penetration Test Quality

The penetration testing performed in this project was done in iterations, and the first iteration was the initial testing of Fotoboks. At the beginning of this iteration the team had no prior experience with penetration testing, which seems to be clearly reflected in their performance. The testing took much longer than later iterations, and a lot of the time was spent reading the OWASP Web Security Testing Guide and conducting research. There were tests that the team did not understand, which meant they had to be revisited later. There were also vulnerabilities that were missed during the initial testing of Fotoboks.

When the team tested ePay, they already had experience from Fotoboks. This experience is likely the reason why the testing of ePay took approximately half as long. There were also fewer tests that was misunderstood or had to be revisited. Overall, the initial testing of ePay was of higher quality than that of Fotoboks.

When the initial testing of ePay was finished, the team started to work on its technical report. While the team worked on the technical report, they decided to revisit some tests and new vulnerabilities were found. In addition to the team being more experienced, the vulnerabilities may have been found because the team was more familiar with ePay. The team may have lacked familiarity with the system during the initial testing, because the information gathering phase was not performed sufficiently. Had the phase been performed sufficiently the vulnerabilities may have been found earlier.

Work on the Fotoboks technical report began after the technical report for ePay was finished. The extra experience from ePay clearly helped. The team redid tests that they had not understood or that had been done incorrectly the first time, which led to the discovery of new vulnerabilities. For example, they found that the cache-control test had been done incorrectly, and when they redid the test, they found issues.

It seems clear that the quality of the penetration tests increased as the team gained experience. However, it is difficult to know how good these penetration tests are compared to a professional one. Given that both vulnerabilities and incorrect tests were found during the last iterations of both systems, the team believes that further iteration would reveal even more issues. However, the tests were reviewed multiple times, which means it is reasonable to believe that most of them were done correctly. Therefore, the technical reports do have some value to the client, even if the test quality is not on a professional level.

5.4 Software Developer Appliance

As the students' understanding of system security grew, so did their understanding of a software developer's responsibility for security. A software developer must ensure a high level of security in their systems to avoid harm to the clients or their users. Sensitive information leakage or other security breaches may cause harm to both companies and individuals. The students have expanded their understanding of the difficulties in providing said security, and they may be better suited to evaluate and prioritize security when developing their own systems.

By performing penetration tests the team has seen web security from the attacker's viewpoint. This has given the students insight into what attackers might want to exploit, and the most common hacking tools. Some of the things an attacker will exploit are outdated components, incorrect handling of user input, missing security headers, and missing cookie attributes. Attackers have access to many hacking tools, where some are used manually to look for weakness, while others perform automated tests or attacks. Burp Suite [32], Fiddler [34], and Wireshark [33] are tools used to manually analyze or modify http requests, while Telnet [35], SoapUI [41], and Postman [37] are manual tools for crafting request to communicate with servers. Software developers must expect usage of such tools and not rely on frontend security, as an attacker can easily bypass such security. Automated tools like Nikto [30], DirBuster [29], GoBuster [28], Nmap [26], Wappalyser [27], and ZAP [31] may fuzz user input, or scan for security headers, endpoints, files, open ports or frameworks. Software developers can use knowledge of these tools to decrease the tools' usefulness by detecting and blocking them. Software developers should assume that attackers can still access the information that the automated tools would provide, and secure the systems accordingly.

The students may also use their penetration testing experience to test their own systems. As a system is developed, the developers usually run tests to make sure it works as intended. These tests often focus on functionality, but they are also used to verify that the authentication and authorization of users work. With penetration testing experience the students may simulate real attacks on these functionalities while the system is still in development. If vulnerabilities are found at this early stage, fixing the issues will be easier, take less time, and cost less.

5.5 Teamwork Evaluation

The team has worked digitally throughout the entire project because of COVID-19. Despite this, the team have worked well together. In weekdays the team has worked from 8 to 16, which made for a good structure that led to steady progression. When testing, the team members worked on individual tests, but would help each other when someone got stuck, which they believed to be an efficient way to work.

6 Conclusion

In this project the team has performed penetration tests on two different systems, ePay and Fotoboks, explored how experience affects such tests, and how penetration test experience can be beneficial for software developers. The testing was primarily focused on web applications, even though a part of ePay was an API. The team did not have time to properly test the API, because it required a different testing approach from that of the web applications.

From the team's experience, it seems clear that performing a penetration test without prior experience will lead to a result of poor quality. However, the quality seems to increase significantly for each penetration test performed, due to the experience gained. Going through tests several times for the same system seems to help as well, because of experience and familiarity with that system.

When performing a penetration test, blindly following the OWASP Web Security Testing Guide is not enough to ensure that the penetration test is of high quality. It is also important to have a deeper understanding of the penetration testing process, which allows a tester to know what to look for, to catch small details, and to see how findings can be combined to find other vulnerabilities. The team believes this kind of understanding can only be obtained through experience.

The team believes there are multiple benefits for a software developer in performing a penetration test. In addition to software developers being able to simulate real attacks while the system is still in development, they will also gain knowledge of how an attacker thinks and will be more aware of common security issues. As software developers, the students expect the experience of penetration testing will aid them in developing more secure systems in the future, by prioritizing security throughout the development.

If the team were to perform another penetration test, they would have done it differently. Because of the difference between testing APIs and web applications, the team would choose to only test one or the other. It is not possible for this team to repeat this exact project, as the basis for it was that the team had no prior experience. For anyone else who would repeat this project, or perform a similar one, the team recommends working more closely with a professional penetration tester. This would allow the team to learn more and deliver a higher quality penetration test.

The team has gotten a lot of guidance and help from their supervisor, who has a lot of experience in the cyber security field, but they believe it would be interesting to get input on their work process from a professional penetration tester. They also believe it would be interesting to compare the team's technical reports with a professional penetration test of the same systems.

7. References

[1] E. Saad and R. Mitchell, "OWASP Web Security Testing Guide v4.2," 3 December 2020. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/. [Accessed 17 January 2021].

[2] OWASP, "OWASP About," [Online]. Available: https://owasp.org/about/. [Accessed 29 April 2021].

[3] HubSpire, "What is an API," [Online]. Available: https://www.hubspire.com/resources/general/application-programming-interface/. [Accessed 10 May 2021].

[4] PTES, "The Penetration Testing Execution Standard," [Online]. Available: http://www.pentest-standard.org/index.php?title=Main_Page&oldid=950. [Accessed 29 April 2021].

[5] K. M. Henry, Penetration Testing: Protecting Networks and Systems, IT Governance Ltd., 2012.

[6] Oregon State Government, "Statement of Work (SOW) Writing Guide," 2 April 2018. [Online]. Available: https://www.oregon.gov/das/Procurement/Guiddoc/SOWWritingGuide.pdf. [Accessed 29 April 2021].

[7] Rocket Lawyer, "Make your Free Non-Disclosure Agreement," [Online]. Available: https://www.rocketlawyer.com/business-and-contracts/intellectual-property/confidentiality-agreements/document/non-disclosure-agreement#:~:text=Disclosure%20Agreement)%3F-,A%20Non%2DDisclosure%20Agreement%20(NDA)%20is%20a%20legally%20enforceable,them%20by%. [Accessed 6 May 2021].

[8] AMC, "About the ACM Organization," [Online]. Available: https://www.acm.org/about-acm/about-the-acm-organization. [Accessed 6 May 2021].

[9] Auth0, "Single Sign-On," Auth0, [Online]. Available: https://auth0.com/docs/sso. [Accessed 19 May 2021].

[10] L. Rosencrance, "Pen Test (penetration testing)," [Online]. Available: https://searchsecurity.techtarget.com/definition/penetration-testing. [Accessed 29 April 2021].

[11] PTES, "Pre-engagement Interactions," [Online]. Available: http://www.pentest-standard.org/index.php?title=Pre-engagement&oldid=940. [Accessed 30 April 2021].

[12] PTES, "Intelligence Gathering," [Online]. Available: http://www.pentest-standard.org/index.php?title=Intelligence_Gathering&oldid=953. [Accessed 30 April 2021].

[13] PTES, "Threat Modeling," [Online]. Available: http://www.pentest-standard.org/index.php?title=Threat_Modeling&oldid=956. [Accessed 30 April 2021].

[14] PTES, "Vulnerability Analysis," [Online]. Available: http://www.pentest-standard.org/index.php?title=Vulnerability_Analysis&oldid=945. [Accessed 30 April 2021].

[15] PTES, "Exploitation," [Online]. Available: http://www.pentest-standard.org/index.php?title=Exploitation&oldid=946. [Accessed 30 April 2021].

[16] PTES, "Post Exploitation," [Online]. Available: http://www.pentest-standard.org/index.php?title=Post_Exploitation&oldid=947. [Accessed 30 April 2021].

[17] PTES, "Reporting," [Online]. Available: http://www.pentest-standard.org/index.php?title=Reporting&oldid=948. [Accessed 30 April 2021].

[18] B. Aston, "How To Write A Statement Of Work In Project Management," 15 01 2021. [Online]. Available: https://thedigitalprojectmanager.com/how-write-statement-of-work-complete-guide/. [Accessed 21 04 29].

[19] K. Bork, "What To Look For In A Penetration Testing Statement Of Work?," [Online]. Available: https://www.triaxiomsecurity.com/what-to-look-for-in-a-penetration-testing-statement-of-work/. [Accessed 29 04 2021].

[20] "PENETRATION TEST (Sample Report)," [Online]. Available: https://pulsarweb.de/Pentest_Report.pdf. [Accessed 10 05 2021].

[21] B. Caudill, "Four Things Every Penetration Test Report Should Have," [Online]. Available: https://rhinosecuritylabs.com/penetration-testing/four-things-every-penetration-test-report/. [Accessed 18 05 2021].

[22] OWASP, "OWASP Risk Rating Methodology," [Online]. Available: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology. [Accessed 1 May 2021].

[23] P. Phongthiproek, "OWASP-Testing-Checklist," 10 08 2019. [Online]. Available: https://github.com/tanprathan/OWASP-Testing-Checklist. [Accessed 01 05 2021].

[24] Offensive Security, "Kali Linux Virtual Images," [Online]. Available: https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/#1572305786534-030ce714-cc3b. [Accessed 12 May 2021].

[25] Microsoft, "Microsoft Store: Kali," [Online]. Available: https://www.microsoft.com/en-us/p/kali-linux/9pkr34tncv07?activetab=pivot:overviewtab. [Accessed 12 May 2021].

[26] "Nmap," [Online]. Available: https://nmap.org/. [Accessed 3 May 2021].

[27] "Wappalyzer," [Online]. Available: https://www.wappalyzer.com/. [Accessed 3 May 2021].

[28] "GoBuster," [Online]. Available: https://github.com/OJ/gobuster. [Accessed 3 May 2021].

[29] "DirBuster," [Online]. Available: https://tools.kali.org/web-applications/dirbuster#:~:text=DirBuster%20is%20a%20multi%20threaded,pages%20and%20applications%20hidden%20within. [Accessed 3 May 2021].

[30] S. Chris and L. David, "CIRT.net Nikto2," [Online]. Available: https://cirt.net/Nikto2. [Accessed 3 May 2021].

[31] "ZAP," [Online]. Available: https://www.zaproxy.org/. [Accessed 3 May 2021].

[32] PortSwigger, "Accelerate application security testing with Burp Suite," [Online]. Available: https://portswigger.net/burp. [Accessed 03 05 2021].

[33] Wireshark, "Wireshark," [Online]. Available: https://www.wireshark.org/. [Accessed 7 May 2021].

[34] Telerik, "Telerik Fiddler," [Online]. Available: https://www.telerik.com/fiddler. [Accessed 7 May 2021].

[35] "Telnet," [Online]. Available: http://www.pcmicro.com/netfoss/telnet.html. [Accessed 3 May 2021].

[36] "SoapUI," [Online]. Available: https://www.soapui.org/. [Accessed 12 05 2021].

[37] Postman, "Postman," [Online]. Available: https://www.postman.com/. [Accessed 7 May 2021].

[38] "Requests Library," [Online]. Available: https://docs.python-requests.org/en/master/. [Accessed 3 May 2021].

[39] "BeautifulSoup Library," [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. [Accessed 3 May 2021].

[40] Association of Computer Machinery, "ACM Code of Ethics and Professional Conduct," [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed 6 May 2021].

[41] SmartBear Software, "SoapUI," [Online]. Available: https://www.soapui.org/. [Accessed 7 May 2021].

[42] ZDI, "The Zero Day Initiative," ZDI, [Online]. Available: https://www.zerodayinitiative.com/advisories/disclosure_policy/. [Accessed 29 April 2021].

[43] Kali Tools, "Kali Linux Tools Listing," [Online]. Available: https://tools.kali.org/tools-listing. [Accessed 03 05 2021].

8. Attachments

Attachment A: Statement of Work

NTNU

Statement of Work

Version 0.1

Magnus Baugerud
Henrik Mathias Berg
Lars Olsnes Østmo-Sæter

February 2021

Version History

Date	Change	Version
01/02/2021	First Draft	0.1

Contents

Version History 1
1 Introduction 1
2 Stakeholders and Distribution of Work 2
2.1 Stakeholders 2
2.2 Distribution of work 2
2.3 Responsibilities 2
3 Scope 3
3.1 Out of scope 3
4 Deliveries 3
4.1 Technical Report 3
4.2 Executive Summary 3
5 Methodology and Threat Model 4
5.1 Methodology 4
5.2 Threat Model 4
6 Disclosure 4
7 Permission to Test 5
References 5

1 Introduction

This document is an overview of the project details for a penetration test of NTNU IT systems, performed by third year bachelor students at the Bachelor of Computer Engineering Course. The systems that will be tested during this project is: Fotoboks, a web application for requesting NTNU access cards; and Epay, a payment API utilized by various NTNU web commerce solutions. The testing will be supervised by Donn Morrison as part of the student's bachelor's thesis. The project will take place in Trondheim and last from January to May 2021.

2 Stakeholders and Distribution of Work

2.1 Stakeholders

Who	*Role*	*Interest*
NTNU IT	Client	Have their systems tested for weaknesses and security vulnerabilities.
Magnus Baugerud , Henrik Mathias Berg, Lars Olsnes Østmo-Sæter	Students	Performing a penetration test with OWASP methodology as basis for a bachelor thesis.
Donn Morrison	Student Supervisor	Assist and advice the students through their project.
NTNU	University	The institution where the students are taking their bachelor thesis. Will also have their system tested.
Various	Threat Actors	Damaging and exploiting the system for personal gain.

2.2 Distribution of work

The students have a total of 1500 hours and estimate that:

- 750 hours will be dedicated to testing the systems.
- 300 hours will be spent on meetings and documentation. Documentation includes documenting findings and writing both executive summary and technical report, documenting progress and status during the project, as well as preparing for meetings and writing meeting minutes.
- 300 hours will be spent on writing the bachelor's thesis.
- 150 hours will be spent on the bachelor's course's included lectures.

2.3 Responsibilities

The students are responsible for completing and submitting the agreed upon deliveries, described in section 4, within the timeframe of the project. They are also responsible for upholding the signed contracts relating to the project, including the NDA and the employment agreement, as well as this statement of work. Finally, the students are responsible for arranging necessary meetings. This involves preparing and distributing meeting notices, conducting and leading the meetings, and writing meeting minutes.

3 Scope

The students will test the following systems on behalf of NTNU IT:

- Fotoboks
- ePay API
- ePay Administration Page

The systems will run in a test environment, and the attacks will be web based and executed remotely through NTNUs VPN. All attacks are permitted except the ones specified in section 3.1.

3.1 Out of scope

The target systems are hosted on a server among other systems. These other systems are off-limit, and there shall not be performed any DoS attacks on the server itself.

There should not be performed any physical attacks, nor any form of social engineering.

4 Deliveries

The results from the penetration test will be put together in a technical report. In addition to the technical report the students will write an executive summary of the findings.

4.1 Technical Report

The Technical report will give a detailed description for the results of each test performed on the systems. All findings no matter what the threat assessment yields will be listed in the technical report, as well as suggestions on how to fix the vulnerabilities. The report will also give a description of the systems and contain the detailed results of all tests performed during the project.

4.2 Executive Summary

The executive summary gives an abstract of the most important findings. It will give a simple non-technical explanation of the issues found during the project. The purpose of this document is not to describe the issues in detail, but to give a description of the penetration test results to people without technical knowledge of web- and software-security.

5 Methodology and Threat Model

The students will use open-source methodologies to perform a thorough and controlled test. The students do not have extensive knowledge of security testing, so a well-known and established open-sourced methodology was chosen for this project.

5.1 Methodology

The students will perform a penetration test of the systems described in section 3 using the *OWASP Web Security Testing Guide* (OWASP, 2021) by systematically going through the checklist made by Prathan Phongthiproek (Phongthiproek, 2021).

5.2 Threat Model

The students will assess the risk of any findings using the OWASP risk rating methodology (OWASP, 2021). All threat assessments will be given in the technical report.

6 Disclosure

If any findings are significantly severe, they will be reported immediately along with suggestions for fixes through established communication channels between the testers and the client. Other findings will be disclosed through the technical report and the executive summary. All findings will be disclosed to the public when the client has approved publication, which can be delayed for a maximum of 5 years from the end of the project as stated in the NDA.

7 Permission to Test

[illegible]	26.3.2021
NTNU IT	Date
[illegible]	29.03.2021
Student Supervisor	Date
Magnus Baugerud	02.03.2021
Student	Date
[illegible]	02.03.2021
Student	Date
Henrik M Berg	02.03.2021
Student	Date

References

OWASP. (2021, February 1). *OWASP Risk Rating Methodology*. Retrieved from OWASP: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology

OWASP. (2021, February 1). *OWASP Web Security Testing Guide*. Retrieved from OWASP: https://owasp.org/www-project-web-security-testing-guide/

Phongthiproek, P. (2021, February 1). *tanprathan / OWASP-Testing-Checklist*. Retrieved from Github: https://github.com/tanprathan/OWASP-Testing-Checklist