NTNU
Norwegian University of
Science and Technology

Mitigation of Cheating Threats in Digital BYOD exams

Thea Marie Søgaard

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Guttorm Sindre, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

Digital exams are not the most common practice in Norwegian higher education, but it is being tested and used at several institutions. Exams are normally conducted on pen and paper with limited access to resources, and it has been a challenge to make an affordable, digital adaption for this type of exams.

One approach to create this digital exam environment, is with the use of lock-down browsers on student owned computers. At NTNU they have chosen to use a lock-down browser called Safe Exam Browser, which is continuously being improved and is widely used for digital exams.

This thesis will look into whether Safe Exam Browser sufficiently protects against cheating during a digital Bring-Your-Own-Device exam. It will be a technical evaluation of Safe Exam Browser as a software, that ensures a secured exam environment and helps mitigate cheating behaviors. The vulnerabilities of Safe Exam Browser is discovered, evaluated and tested with penetration testing.

Safe Exam Browser does have vulnerabilities, and some of them are presented in this thesis. It is open source and for exams conducted at NTNU, it is installed on the examinees' personal computers, and it will always be possible that someone choose to use a corrupted version to be able to cheat. There are different mechanisms in this software, to detect and deny access to exams, if the lock-down browser has been modified. These mechanisms make it harder, but not impossible, to make and use a corrupted version.

Sammendrag

Digital eksamen i Norge er ikke like utbredt som eksamen med pen og papir, for høyere utdanning, men det brukes og testes ulike løsninger rundt omkring. Eksamen foregår normalt med pen og papir uten tilgang til andre hjelpemidler. Det er utfordrende å finne rimelige, digitale løsninger som kan begrense ressurstilgangen på samme måte som ved en slik papir eksamen.

En av de eksisterende løsningene er å bruke en nettleser som låser kandidatens datamaskin (lock-down browsers). Ved NTNU har de valgt å bruke en slik nettleser, kallt 'Safe Exam Browser', som fortsatt blir forbedret og brukes i mange forskjellige digitale eksamensløsninger.

Denne masteren vil se på om Safe Exam Browser kan beskytte mot fusk på en tilfredsstillende måte, under en eksamen. Den vil inneholde en teknisk evaluering av Safe Exam Browser, i forhold til hvordan den sikrer en digital eksamensløsning og begrenser fusk. Sårbarheter hos Safe Exam Browser ble oppdaget, evaluert og testet gjennom penetrasjonstesting.

Safe Exam Browser har sårbarheter, der noen av dem blir diskutert her. Programmet har åpen kildekode og er, ved eksamen hos NTNU, installert på eksamenskandidatenes personlige datamaskiner. Det vil alltid være mulig for noen å bruke en korrupt versjon av programmet for å jukse under en eksamen, og det er flere mekanismer for å hindre utnyttelse av dette. Disse mekanismene gjør det vanskeligere, men ikke umulig, å bruke en korrupt versjon.

Acknowledgements

I would like to thank my supervisor Guttorm Sindre for his guidance and advice throughout the thesis work. He suggested this topic to me already in the specialization project in the fall, and he enabled me to choose the approach that I found most interesting. While he, at the same time, challenged me by suggesting additional views and approaches.

I would like to thank PhD candidate Aparna Vegendla, who have paid a special interest in my project, ever since I started working with the specialization project. She helped me set up a working test environment, and was always available to give me suggestion, advice or feedback. She pushed me to work harder, and helped me get through the periods when I found it hard to get motivated.

Contents

List of Figures

List of Tables

Glossary

browser exam key This is a feature provided by SEB to ensure secure use of the correct software and configuration settings.

clipboard is an option that is available in almost any Windows computer. You can copy data to the clipboard and paste it somewhere else from the clipboard, either within or between applications.

Gecko Mozilla open source rendering engine, for rendering web pages/HTML.

Inspera A Norwegian company that makes, improves and operates Inspera Assessment, for digital exams with SEB.

Moodle An open source learning management system with tools to create quizzes and exams, and a plugin option for the use of safe exam browser with the browser exam key.

registry The registry editor is a Windows(OS) feature that enables an advanced Windows user to change the configurations for different system features, in one location. This is also referred to as the registry.

SEB Client refers to the SebWindowsClient application. Which is SEB's lock-down browser, this is the main application of SEB.

SEB Config refers to the part of the SEB source code, that provides the SEB ConfigTool, which is used to create configuration settings for a running instance of the SEB lock-down browser.

SEB ConfigTool is a graphical user interface used to create SEB configuration files. It is included in the general SEB application installment, and the source code is located in SebWindowsConfig folder in the SEB source code.

SEB Resetter Refers to the SEB registry resetter, which is a small application that enable users to reset the registry settings that have been modified by the Safe Exam Browser Service.

SEB Service refers to the SebWindowsServiceWCF and is the service application which creates and control the running SEB Windows service for the SEB installation on your BYOD computer. This service does registry editing and is accessed by SEB client and SEB Resetter.

UNINETT A company, owned by the Norwegian department of education, that provides network and network services to Universities and colleges.

XULRunner is a Mozilla runtime package. The framework is, amongst other things used to embed Gecko, to form a complete browser engine.

Acronyms

ADT Attack Defence Tree.

API Application Programming Interface.

ASD Attack Sequence Description.

BYOD Bring Your Own Device.

ETH Zurich Swiss Federal Institute of Technology.

GNU GNU is Not Unix.

GPL GNU General Public License.

GUI Graphical User Interface.

HARM Hacker Attack Representation Method.

IA Inspera Assessment.

LDB Lock-Down Browser.

LMS Learning Management System.

MPL Mozilla Public License.

MUCM Misuse Case Map.

MUSD Misuse Sequence Diagram.

NTNU Norwegian University of Science and Technology.

OS Operating System.

OSS Open Source Software.

SEB Safe Exam Browser.

UML Unified Modeling Language.

VDI Virtual Desktop Infrastructure.

VM Virtual Machine.

WCF Windows Communication Foundation.

XML EXtensible Markup Language.

Chapter 1

Introduction

This chapter will provide a basis knowledge on the motivation, scope and approaches for this thesis. The first section will discuss the motivation behind the topic, based in scientific research and in personal experiences. The second section will define the scope of the thesis. Then I will discuss the research questions for the thesis. The research methods will be presented, before the related ethical concerns are discussed. Finally, I will present the outline of this report.

1.1 Motivation

For a long time, pen and paper exams have been used for assessment in higher education. Students are to use their knowledge of a particular subject to answer questions on a written exam. Gradually, technology have become an important part of education on all levels, from primary school, to high-school, college and universities. Whether it is used by the teacher to assess students, distribute and present learning material, or by the students to practice their knowledge, find literature or answer and deliver assignments. Students use computers, Learning Management Systems (LMSs), and other digital tools regularly, to perform study tasks and assignments [12]. Higher education are using end-of-term, pen on paper exams for assessing students in a large percentage of courses, this includes the Norwegian University of Science and Technology (NTNU) [49]. This practice should be, and is being, modernized, to comply with the increased use of technology in other areas of education. Students have become more comfortable with typing on keyboards than writing on paper [18, 20], due to writing being replaces with typing in several aspects of everyday life. Studies imply that people have decreasing level of motor skills today, due to these developments [50]. The only time a student will be writing, with pen on paper, for any significant amount of time, is during an examination. The readability, along with the amount of text one is able to produce during this time, is greatly affected by whether it is conducted digitally or manually. The variations in handwriting skills can affect a teacher's evaluation of the content of for example an exam answer [23]. The text's readability affects the objectivity of the examiner, it has also been suggested that typed text in general receives a better result than written text[26, 34]. If all students type their exam answers we remove the evaluation bias of readability, because all the responses will have equal readability. Also, the answers will be more anonymous. A teacher/professor with a relatively small number of students, might recognize their handwriting, when grading their exam answers. The grading might then be affected by some bias toward the students, even if they attempt to be unbiased.

Digital examinations may also participate in reducing students' discomfort and stress level [18, 20]. As we have already argued, they are more comfortable with typing, than with handwriting. They can easily move text around, copy-paste and navigate back and forth in a digital document. Examinees do not have to manually organize paper sheets with written copies, and they do not have to rewrite their draft into a final version, before the exam is over.

Important quality criteria for digital exams could be validity, reliability, costs, acceptability and feasibility [37, 48]. The validity criteria depends on whether the assessment gives a representative measurements of the students knowledge, skills, competences and other related aspects which represent how a student will handle related work situations in their future. For example if the student has the basis to become a productive employee in his future carrier. The reliability criteria requires that the exam assessment is consistent. The same answers should receive the same score. Acceptability depends on whether students and faculty are confident with the assessment, and the availability of necessary equipment is described as feasibility. The criteria that have been described, could also be used to assess the quality of pen and paper exams.

Reliability can be achieved by anonymity and objective evaluation, which have already been discussed in this section. By digitalizing exams, the risk of the evaluation being affected by handwriting skills and recognition is decreased. Reliability could also be improved by implementing automatic scoring of certain question types, such as for multiple choice questions, to eliminate human bias and error when evaluating.

Reliability could also be reduced with digital exams, for example if the system that is being used is unreliable. The automatic scoring can be implemented wrong, which probably will have greater consequences for the reliability than a few human errors. It is important to ensure that the software being used is reliable and sufficiently tested.

Validity can be increased by making the digital exam environment similar to real-life situations. The exams have to assess the students, based on the course's learning goals, and make grades reflect their knowledge. Assessment and result should reflect how the candidate will cope in his or her future carrier. Digital exam assignments have the potential to use more tools and creativity during assessment, by for instance using interaction and multimedia. For example to play sounds or show images that the students need to identify, or in programming courses, examinees could be able to both look up and access certain code libraries and run their own code during an exam. The possibility to integrate allowed tools into the exam environments, could give a more realistic evaluation of a student's skills.

The validity of an exam can also create a bad reflection of student's skills and knowledge. If the exam software has bad usability, the results can be negatively affected, and make the student look less desirable, for future employers, than they deserve. The reliability could also be negatively affected by cheating behaviors. Students who cheat, will get unreliable results, which make them more attractive to future employers, even if they lack the necessary skill set. In this scenario the students who do not cheat will end up as victims. They will have to compete with students, that have gained an unfair advantage, for the same jobs, or the same Master and PhD programs. The future employers of the cheating examinees will also be negatively affected, by increased probability to hire unqualified personnel. This could also damage a University's reputation, as the graduated, cheating students, will be unqualified for their jobs.

Research about the acceptability towards digitization of exams is limited, and the results varies depending on for example which university, faculty, and solution that is being used. Hillier [19] has conducted a digital exam test run and surveyed the students on their experiences. The participation was limited and the perceptions varied. Acceptance should increase when students and faculty members have gained more experience with the digital exam solution. Students, as discussed previously, are more experienced with typing than with writing. This should increase their acceptability

There have been conducted a study into students' views on online exams at a University in Florida [25], where they poll business students. The students are asked about different behaviors and issues regarding online exams, as well as their perception on cheating on traditional versus online coursework. Almost 74 % of the students that where polled, believe that it it easier to cheat on online course work, which does not reflect good acceptability amongst students.

The two remaining criteria are feasibility and cost. Feasibility represent the necessity of having enough equipment, of sufficient quality. Costs can, for example, be related to the resources necessary to acquire and maintain this equipment. These criteria could maybe be the drivers that made NTNU choose to conduct digital exams on student owned equipment, which, according to UNINETT [36], is the cheapest solution.

Cheating is a major concern for all types of assessment, and there are several existing methods to mitigate cheating. During an exam there should be invigilators present, to assist the examinees and to detect cheating. For papers written at home, there exist several different tools to conduct plagiarism control with, to ensure the paper has not been copied. There are, however, fewer mechanisms to catch students who get assistance on their home exams. The punishment, if you are caught cheating, is severe, and it may include being kicked out from the school. At NTNU, there are few episodes were cheaters have been caught and prosecuted, probably because students who cheat don't get caught and it is expenses to prosecute cheating students in Norway. According to McCabe [30], cheating behaviors are used and perceived and cheating mitigation is an ongoing struggle.

Cheating is a major threat toward the quality criteria we have discussed. In particular towards acceptability and validity. Cheating is important to avoid, in any kind of high-stakes assessment, it has been a threat for paper based exams and it will be a threat for digital exams. Digitalizing exams will introduce new vulnerabilities and behaviors that can be used to cheat. It also introduces new ways to mitigate cheating.

1.2 Scope of the Thesis

This thesis will focus mainly on the validity criteria, by discussing different aspects of cheating in digital exams In particular, cheating behaviors during a digital examination. The main research will be conducted on an open source exam environment that should imitate the environment that is used at NTNU for digital exams. I will assess this open source environment, to gain a perspective on the vulnerabilities that could potentially be used to cheat during an exam. The vulnerabilities will be used to plan and conduct attacks, in a lab environment, and I will assess the validity of these attacks by comparing them with real-life exam situations. I will discuss possible mitigation strategies, mainly regarding the technological and social aspects. I will also assess the method that I use to conduct these security tests, to see if this is an approach that can be recommended. This thesis is not meant to aid students with cheating behaviors, but rather to inform the stakeholders of the particular risks involved with digital assessment. The technical details should be vague enough to make them harder to copy, while at the same time give value to the stakeholders. It is important to emphasize that the vulnerabilities that are presented are only concerned with the software that is being assessed, in a lab environment, and there are no guarantees that the attacks will work in a real-life digital assessment situation. The vulnerabilities and mitigations that are discussed are only concerned with cheating behaviors during an exam, that have the possibility to increase the academic result for the cheating examinee for that exam.

1.3 Research Questions

The objectives of this master thesis is to look further into the security of the technical solution, which NTNU has chosen, to conduct digital exams. The main focus will be on cheating related security during an exam, which means for an examinee to use any form of communication or use prohibited tools while conducting an exam. Cheating has been defined as a behavior that may compromise academic integrity [30], and is a known issue with regard to student assessment. Cheating behavior is not unique for digital assessment, but have been researched and discussed in regards to written assessment as well. For written end-of-term assessment, communication and the use of prohibited tools and notes during an exam is considered to be cheating behaviors. For digital end-of-term assessment, the amount of different cheating behaviors are increased. In addition to traditional ways of cheating, for example to bring crib notes or use code signals to communicate, the examinees can take advantage of the technological opportunities. What are the technological opportunities, and how can cheating be prevented so that future academic integrity will not be compromised due to digitization. I will look at cheating behaviors that are realized through the BYOD exam computer, and not discuss the possibilities regarding the use of external tools, for example a mobile phone.

RQ1 What are the cheating vulnerabilities of SEB and how can they be mitigated?

RQ2 What are the cheating vulnerabilities of SEB used with Moodle during an exam?

RQ3 What are the strengths and weaknesses of HARM [24, 52] used for penetration test planning?

RQ1 is related to the prestudy [46], which is presented in section 2.7. The vulnerabilities that was confirmed by penetration testing in the prestudy, will be assessed and mitigation strategies will be suggested. The prestudy tested the open source lock down browser called Safe Exam Browser, without a learning management system, which it is designed for. There are learning management systems(LMSs) that have a special plug-in for SEB exams, that helps increase the security features that are designed to prevent cheating behaviors. To answer **RQ2**, I will plan and conduct penetration tests, to assess these security features, and I will discuss how they can be mitigates. To plan the penetration tests I will use a method, which was suggested by my advisor and one of his Ph.D candidates. It is an extension of the Hacker Attack Representation Method (HARM), designed to plan penetration testing, which was partially based on my prestudy. I will use this method and assess the strengths and weaknesses discovered during the process, to answer the third research question (**RQ3**).

1.4 Research Method

This thesis consists of an empirical study of the digital solution that are being used for conducting exams at NTNU, and the existing cheating mitigation mechanisms. The research questions **RQ1** and **RQ2** are mostly of a technical nature. Firstly research of the topics involved will be conducted based in literature, to create a basis of knowledge and explain the motivation. I have personal experience and motivation for this study as I have participated in a number of written exams at NTNU, and digital exams during my high-school years. The main inspiration for my prestudy [46] came from Dawson's "Five ways to hack with BYOD e-exams" [13], where he rejects the idea of digital assessment in the form of Bring Your Own Device (BYOD) e-exams due to the many cheating vulnerabilities he discovered. For this master thesis I will use my prestudy as conceptual framework, to expand the findings of that project, and get more valid results.

The research questions will be covered by exploratory case studies [38, p. 143], to explain the issues of digital examinations by using penetration testing and planning. It will be an in depth, short-term, contemporary study. **RQ1** will be answered by code analysis. Static and dynamic code analysis will be applied to gain an understanding of how the security features, that where tested in the prestudy, function. **RQ3** will be based on penetration test planning, by using the method that are to be assessed. To answer **RQ2**, I will conduct the penetration tests that was planed using HARM, by the same approach used in the prestudy, but the testing will be extended further. The Hacker Attack Representation Method (HARM) method has been extended to give a systematic approach to penetration test development [52], and should help create a good overview, through different models, on relevant vulnerabilities, threats and mitigations for the system.

1.5 Ethical Issues

Ethics is an important subject when it comes to both cheating behaviors and software hacking, it should be emphasized that this research is not meant to aid students in any way to turn to cheating behaviors, but rather to decrease the occurrence of cheating. SEB is an Open Source Software (OSS), which could be a disadvantage with regard to security, as any examinee can access and modify the source code, and modify their SEB installation. The advantage is that any researcher can distribute to, and test, the source code as they please.

SEB is an OSS, licensed by Mozilla Public License (MPL) version 1.1 (see appendix D.1) [35]. MPL is an open source license, which allows free reuse of the original software, as long as developers keeps the license. There are many different open source licenses with different demands toward distribution of software that are modified versions of, or has been derived through, the original licensed software [17].

The LMS used for the penetration tests is called Moodle and it is licensed under the GNU General Public License (GPL) (shown in appendix D.2). The GNU license is a common open source license, and software under this license is copylefted. Copylefted software is free for access and redistribution, which mean that all derived software has to also be open source [40]. Software that are not copylefted, can be used to develop commercial software that are not open source. This decreases contributions from independent developers in improving the original software. The core files of software under MPL is copylefted, to motivate independent contributions. SEB uses a browser engine called Gecko, which is developed by Mozilla and licensed under MPL This is why SEB is also licensed under MPL, and it is free for testers, developers and others to use as they like and contribute, as long as they continue using the MPL on redistributions and derived software.

During a presentation held by the lead developer of SEB during a fall seminar with Inspera, he explains that because Gecko is no longer being supported by Mozilla, SEB will have to start using a different browser engine [44]. This will probably be WebKit, which is the browser engine that is already being used in SEB for Mac users. Even though this is likely to happen, he ensures the audience that SEB will still continue to be open source, maybe be under a different license.

To conduct ethical hacking, it is important to not damage the target system, but assess the security with the aim to increase it [42]. I informed the lead developer of SEB at ETH Zurich of my findings in the prestudy, and I will continue to report any findings, so they can be used to mitigate the discovered threats. I also disclosed my findings to the exam office at NTNU and Inspera. I will make sure that my tests will not harm any systems permanently and I will report my findings to the stakeholders to ensure that my research remains ethical.

As a conclusion, I would say that I am free to use, test and document the SEB software and the Moodle web application, as long as I keep the existing licensing for potential redistributions of the software. However, based in ethical guidelines, I still hold an obligation to inform the stakeholders of any vulnerabilities that I discover, that could have impact on the application and it's intended use.

1.6 Report Outline

Chapter 2: Background Research and Related Work: An introduction of the technologies, prior work and existing studies which will be used in the thesis. This chapter presents the current status of digital exams at different institutions, including NTNU.

Chapter 3: SEB: Code and Architectural Overview: This chapter presents the technological details of SEB and creates an overview of the applications architecture, through descriptions, diagrams and figures.

Chapter 4: HARM, Vulnerabilities and Risks: This chapter shows how the HARM method was used to discover vulnerabilities and plan penetration tests.

Chapter 5: Penetration Testing: Describes the penetration tests, and how the systems and configurations where set up. It presents the tests, their outcomes and discuss the results.

Chapter 6: Discussion: This chapter further discuss the test results, with regards to cheating and mitigation strategies. It will contain an assessment of the affect of HARM for the test planning, and finalize the basis knowledge for which to make the conclusion.

Chapter 7: Conclusion and Future Work: This final chapter will conclude by answering the research questions, and present suggestion to future work based on this conclusion.

Chapter 2

Background Research and Related Work

In this chapter, I will discuss research and theories that can be related to this thesis. The first section presents the situation at NTNU regarding digital exams. The next two sections discuss cheating behaviors in both paper based and digital exams and how cheating can be an important topic with regard to digitizing of exams and exam processes. The fourth and fifth sections should create a background of the tools that are being tested in this thesis, SEB and Moodle. Sections 6 describes briefly a selection of different existing solutions that are being used to create a secured environment for digital exams. The next section presents and discuss the specialization project, from the fall of 2015. This project will be referred to as the prestudy throughout this thesis. The last two sections will present and explain different approaches used to create and maintain secure systems and how to plan and conduct penetration testing.

2.1 Digital Exams at NTNU

In this master thesis, I will attempt to assist in the improving of NTNU's digital exam solution with regard to mitigating cheating during digital exams. I will do research, testing and evaluation of the technology being used. NTNU is currently running small scale exams for testing purposes, using student owned equipment. There have been conducted thorough investigation, to determine which solution fulfills the requirements of digital exams at NTNU, partially covered by a preliminary report by NTNU [21] and a report written for UNINETT [36]. The NTNU preliminary report describes why digital exams is the appropriate apprach for the future of assessment. NTNU hosts about 60 000 written exams per semester, in the span of 18 days(3 weeks) with two subsequent sessions each day. In order to keep this schedule and size, the University would need to invest in at least 1667 exam workstations, and the costs where estimated to be between 6.7 and 10 MNOK [21]. The maintenance, operations and other administrative costs are excluded. If the exams are run on University owned equipment, it will be easier to control the digital environment throughout the exam, and it makes sure of equal conditions for the examinees. The preliminary report also discuss BYOD as a possible solution. They argue that most students today have their own PCs, to use for school work, that they should be more familiar with, despite the argument that this might give an advantage to students with more money to spend on equipment. They suggest to use the laptops as thin clients running a Virtual Desktop Infrastructure (VDI), which will be hosted by one or more servers. One of the main challenges of BYOD exams is all the different technologies, equipment, operating systems and devices that the exam system need to support. There are a need to test the compatibility of student equipment prior to an exam, to make sure all examinees manage to run the system.

In the UNINETT report [36], they further investigate different relevant BYOD solutions focusing mainly on requirements regarding cheating, loss of data, scalability, anonymity and storing of data. They recommend two methods for conducting BYOD exams, one VDI solution and one solution using only LMS, both with SEB. These solutions fulfill the main requirements, is scalable and can include different tools and software in the controlled environment. The first solution that was mentioned, is a relatively expensive solution, with the need to invest in physical servers, necessary licenses, and operations. The second solution they recommend is to install SEB on the student's Operating System (OS), to run a cloud based learning management system that contains the exam, and tools to answer and deliver it. This service is provided by Inspera, and they do the technological setup for exams by using Amazon cloud services. The Evry report was made before Inspera was able to give an exact cost estimate, but it seems to be the cheapest solution. To be able to conform with the restrictions in Norwegian laws about privacy and archiving personal documents, the cloud service that are to be used have to fulfill certain demands, stated in a Norwegian report on the legal aspect of digital assessment in [15]. All documents that contain personal information, such as delivered exams and results, can not be stored for any longer than necessary. This information can only be stored by a service that is pre approved, and stores the data safely. There is no need to invest in servers or equipment for this solution, if an approved cloud storage service can be used. The institution is responsible for any consequences if the data gets compromised, and they have to strictly evaluate the cloud storage service. The Amazon Web Service used by Inspera is located within the EEA (European Economic Area), and they have to follow the EU data protection laws, This is one of the properties, recommended by the legal report[15], that ensures a sufficiently secured cloud storage service

NTNU is currently cooperating with Inspera to test and scale a digital exam solution. Lecturers of small courses have the possibility to run their exam digitally, using SEB to run Inspera Assessment (IA). Students use a predefined configuration file to run IA in Safe Exam Browser (SEB), which provides authentication, distribution of exams, a text editing tool, a multiple choice environment, and more exam assignment features in a locked environment. It administers delivery options to the user, or IA delivers automatically, when the exam time is up. For the purpose of this thesis, I will assess an exam environment that should be similar to the environment used at NTNU. Moodle will replace Inspera Assessment as the learning management system, while SEB will be used as intended.

2.2 Cheating Behaviors

There have been attempts to research academic cheating, but this is difficult due to the ethical and moral aspects of the problem [8]. The attempts that have been made by surveying students and faculty with regard to cheating, have the risk of receiving few and dishonest responses, as students can be hesitant to admit to cheating. There are several ways to define cheating, and the perception of what is considered to be cheating behavior may be subjective. Cheating can be described as all behaviors that are not strictly correct, it can involve academic achievements through immoral behavior, or as defined in [25], behaviors that compromise academic integrity. They define cheating as ”a transgression against academic integrity which entails taking an unfair advantage that results in a misrepresentation of a student’s ability and grasp of knowledge”[25]. For this thesis we will use this definition of cheating behavior.

The problem of cheating is familiar to any university, but has rarely been researched and explored. According to [8], cheating is relatively normal, while detection rates can be as low as 1,3%. This implies that cheating behavior for the most part gives students an advantage, with little risk of consequences.

McCabe [30] describes six cheating behaviors which he uses as examples of cheating for his student and faculty surveys. Some of these behaviors include, learning what is on a test beforehand, copying off of another student, help someone cheat and using unauthorized crib notes. He surveys students about their perceptions of the different behaviors, and discovers that the degree of average acceptance toward certain behaviors is reflected by the number of students who have actually participated in these behaviors. This tendency was also recognized in a study on password policies in organizations[10]. The perception and understanding of different policies were reflected in how employees followed their strict, and sometimes impossible, password policies. The more a behavior is accepted and understood, the more it is being used. McCabe’s surveys have a response rate average of 10-15%, and the answers are subjective, which makes it important to understand that the results are not a perfect general picture of the cheating climate. It still contributes to give a relative indication of cheating behavior tendencies.

A similar study conducted with first year IT students [47], shows a tendency towards toleration and use of cheating behaviors amongst students. The study shows that more than half the participants admitted to having cheated, and also would not report it if they see someone else cheat. These studies indicate that cheating is not a particularly new phenomenon, and it is an important threat against assessment quality.

2.3 Cheating Behaviors and Digital Exams

The need to conduct digital assessment in higher education is increasing. Digital assessment of different study assignments are already being used in practice, but for exams with controlled environment, the majority is still done with pen and paper. King et al. conducted a study with 121 business students' perceptions on online exams and cheating [25]. The majority of these students believe that cheating is easier with online assessment, and cheating in online exams were less disdained than cheating in traditional exams.

The results presented above creates a motivation to further investigate the cheating possibilities in digital exams, particularly in comparison with the existing cheating threats of pen and paper exams. Sindre and Vegendla presents such a study [49], where they discuss the pros and cons for digital and paper based exams. They focus on a digital Bring Your Own Device (BYOD) solution, which is presented as the most affordable, digital solution in a study performed by the Norwegian company and ordered by UNINETT [36]. Sindre and Vegendla claim that, even though a digital BYOD exam presents more possibilities for cheating, it also creates more methods to mitigate cheating. Examples of such mitigation strategies could be desktop and network surveillance, biometric identification of examinees, mixed seating and non-uniform questions.

Sessink et al. suggests a secured online exam solution by using a lock-down browser and a learning management system [45]. They suggest to use this, together with supervision of the client computer and network activities, which can only be sufficiently implemented on institution owned equipment. Hillier and Fluck argue, in their paper, on the benefits of digital assessment [20], and how it is about time to develop a stable solution for this purpose. They recommend BYOD exams, because of the costs, maintenance requirements and that most students already own a computer that they are familiar with.

Dawson [13] presents his research into digital BYOD exams by presenting 5 different ways that he has managed to hack different systems and was able to cheat. He states that this form of digital assessment is not sufficiently secure, in order to ensure equal or less cheating than with traditional pen and paper exams.

There are several approaches that can be taken to mitigate cheating in high-stakes exams. One way is to use risk avoidance, and avoid the risk completely, by not having exams. This is not an option as it removes any indication of the individual achievements by students. It could be possible to allow behaviors that are considered cheating. This could be a valid option for some forms of assessment, but if communication is allowed, it is hard to ensure that the grades ill sufficiently reflect a student's skills. To allow certain tools, not related to communication, could not only limit cheating behaviors, but also make the exam a better representation of a real-life situation. Cheating detection is a valid approach but, as previously stated, it is expensive to prosecute student. The final approach is to eliminate all cheating behavior by, using preventive actions. At a certain point, cheating behaviors become sufficiently rare, and the costs of further mitigation will outweigh the benefits. SEB is software that attempts to prevent cheating behaviors, but it only mitigates threats that are related to the digital BYOD environment. Other behaviors, for example impersonation, are not mitigated.

2.4 Safe Exam Browser

There are to versions of Safe Exam Browser, one for Mac and one for Microsoft Windows computers. This master thesis will focus on the version of SEB made for Microsoft Windows machines because this is the operating system that has registered the highest number of users, and one of these users is the author. The thesis will not assess the SEB version designed for Mac users, due to time limitation and limited access to equipment. The newest version available is Safe Exam Browser 2.1.2 for Windows, which was released on March 24th 2016 [6]. The version that was used for the prestudy was Safe Exam Browser 2.1, which is the main reason why this master thesis will continue to work on this version. The differences between 2.1 and 2.1.2 will be discussed in chapter 3. SEB 2.1 is optimized for Microsoft Windows 7 and 8/8.1, including a touch optimized mode for Windows tablet computers [1].

As previously mentioned, SEB creates a locked-down kiosk environment, containing a browser with restrictions on web navigation. It is designed to open the LMS which contains the exam, in a full screen browser window. SEB provides features such as process surveillance, it detects if it is running on a VM, access to allowed third party applications and a built in browser that can communicate with any LMS available [1]. When SEB is running, it prevents the user from using certain shortcut keys that can help access text, other programs or exit SEB. It prevents the user from right clicking, switching applications, accessing menus, accessing Windows Security Screen and using the print screen shortcuts. SEB monitors processes and stops new ones from being initiated while the exam is running. It prevents the computer from going idle or into sleep mode. SEB has a graphical user interface to edit and save configuration settings, where security features can be activated and disabled, such as allowing a third party application to run with SEB. This graphical configuration tool is meant to be used by the exam and course administrations, to configure and secure settings as files to be distributed to the students. To make sure that all students use the same configuration file, a browser exam key is generated by using the software version and configuration settings. If SEB is supported by the LMS that is being used, the LMS will use this key to check whether the configuration and software version are correct and that the source code has not been modified[43].

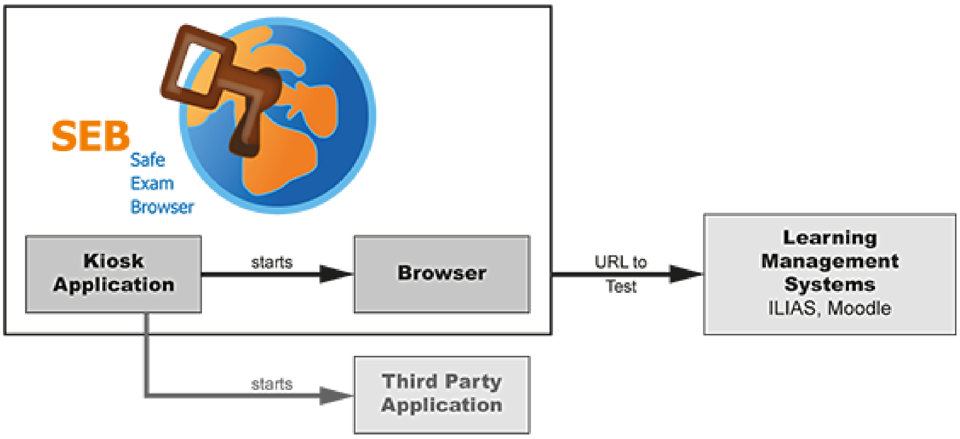


Figure 2.1: Illustration of an online exam with SEB. [1]

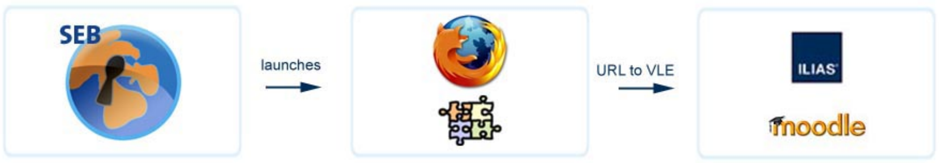


Figure 2.2: Illustration of SEB's architecture. [2].

SEB uses a browser engine called Gecko made by the Mozilla Foundation [1], licensed under MPL. They use the programming languages C#, with the .NET framework, and JavaScript. Safe Exam Browser for Windows consists of a kiosk application and a browser, (see figure 2.1). The kiosk application opens a controlled Windows desktop environment, which runs third part applications, that have to be defined in the configuration settings, and Mozilla's Gecko. SEB for Windows uses Gecko, which provides the framework for running SEB, the browser does not contain any navigation options and status bar, to make sure the user will remain in the preconfigured LMS environment (see figure 2.2).

The application that provides a user interface for editing the settings, is called SEB ConfigTool, which is described in section 3.4. This configuration tool will be used for creating a default, exam configuration file for all the tests, and for creating modified, 'malicious' settings, represented in XML in appendix B.2. The SEB developers recommend to only use encrypted configuration files, when giving the settings to the examinees, or the settings can be directly manipulated in a text editor.

SEB 2.1.2 has a few new features and fixes that does not exist in the version that is being researched in this thesis. The major functionalities that will be assessed in this thesis, by reviewing and testing SEB v. 2.1, should be the same in the different versions. Some of these features are hardware acceleration, Windows 10 compatibility and touch mode is enabled [5].

2.5 Moodle

Moodle is an open source learning management system, and can be used directly as a learning management system or as a framework for a LMS. Moodle is a web application written in php and HTML, and can be downloaded, modified and/or installed on any server of your choice. The Moodle source code can be downloaded from the moodle web page [3]. On the default Moodle version, it is possible to create tests and quizzes, using different question types. One can create multiple choice questions, questions were the student can pair different alternatives, and short and long text answer questions. Moodle includes logic that can automatically evaluate the appropriate question types, such as multiple choice questions.

One of the modifications that can be done on a customized Moodle installations, is to install the Safe Exam Browser plug-in specifically designed for Moodle. This enables you to use SEB to create a secured exam, that only allows access a quiz if the correct SEB browser exam key is used. This is made to ensure that, if an exam is made as a quiz in Moodle (with different question types) it can only be accessed from the specified, secured environment.

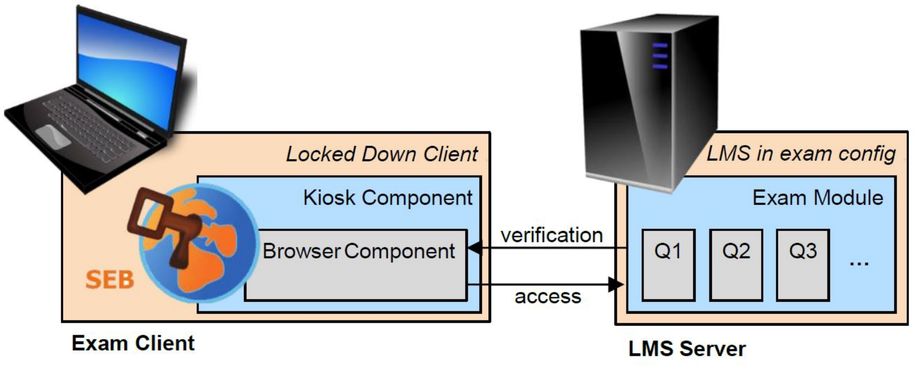


Figure 2.3: Schematic of the technical setup for browser-based online exams, using SEB [18].

2.6 Digital Exam Solutions in Practice

There are several different ways to conduct digital exams. For Norwegian high-schools, students use their own computers connected to a restricted network where they can only access and deliver the exam. They are allowed to access and use all resources and tools that are stored on their personal computers, as long as they do not communicate, which includes Internet access [51]. For higher-education there is a need to conduct digital examinations in controlled environments, similar to that of pen and paper exams in a controlled location, with invigilators. This makes the digitization process more complicated. Examples of proposed digital solutions are to use thin clients in fixed computer labs [18], to boot the exam system from a USB or DVD[13, 16, 20] or to use a Lock-Down Browser (LDB) [12, 13, 18], or various combinations of these.

One of the many institutions to introduce digital exams is the Swiss Federal Institute of Technology (ETH Zurich). In 2007, they launched the "Online Exams at ETH Zurich" project [18]. The goal was to improve learning outcome by making examinations more meaningful and motivating for students. Their objective is to improve the overall assessment-, scoring- and pedagogical processes by implementing a secured digital exam environment. They use Safe Exam Browser (SEB) along with the Learning Management System (LMS) Moodle in fixed computer labs on virtual desktop computers, where the examiners can control network access. SEB is developed as an Open Source Software (OSS) project at Swiss Federal Institute of Technology (ETH Zurich). and is continuously being improved and maintained by the developers of this project. Figure 2.3 shows how SEB can be used in a computer lab to conduct digital exams in a controlled environment. By using invigilators, controlled computer labs, good service design and operations, ETH Zurich manages to conduct secure and reliable digital exams. Some important properties of a digital exam solution is scalability as well as reliability.[20, 49]

It is normal for institutions of higher education to conduct exams at the end of each term for a large proportion of courses. This makes the necessity to conduct a big number of exams concurrently. At ETH Zurich in 2014, they had enough computers to conduct exams for about 548 examinees daily [18]. At a seminar, hosted by Inspera, in September of 2015, the SEB lead developer Schneider stated that they are able to conduct digital exams for more than 50 courses and 5000 students and that they will continue to expand the capacity[44]. The system they developed have the possibility to be extended to work for a much bigger amount of examinees, the problem is the number of computers and computer labs. Their solution is not scalable enough, and to increase the capacity, without adding more computers and computer labs. They plan to increase the capacity at a regular pace, while testing the system, until they meet the needed number of concurrent exams, but this is a costly and time consuming process.

One possible way to decrease the costs of making a digital exam solution scalable, could be to use BYOD. Most students today have their own personal laptop, which they use regularly for, amongst other things, school work. If they can answer an exam, using their own computers, the investment costs for the institutions will decrease and the scalability will increase. BYOD stands for Bring Your Own Device and is a concept used mostly to describe how employees, of an enterprise, are allowed to access internal corporate resources with their own personal devices [39]. BYOD can also be applied in education, for example if students downloads and runs a digital exam system on their personal computers. The exams no longer have to be held in computer labs, with loads of expensive University owned equipment. This Bring Your Own Device (BYOD) solution is harder to invigilate and control, than ETH Zurich's solution.

In Norway, the number of digital exams are increasing. The government owned company UNINETT has developed guidelines to aid the process of digitizing examinations in higher education. The University of Oslo use at least 3 different approaches to conduct digital exams, where law students and medical students use other, specialized digital exam solutions. One of the solutions is provided by Inspera. Inspera organizes digital BYOD exams using Safe Exam Browser, and their own learning management system called Inspera Assessment. Inspera is working with several institutions all around Norway, for example the University of Agder, the University of Bergen, 'Høgskolen i Sør-Trøndelag(HIST)' and NTNU[9, 11, 29, 37]. HIST and NTNU was merged January 1th, but their cooperation with Inspera continues.

According to Schneider, who created SEB, the lock-down browser is only secure to a certain point for BYOD exams [44]. He explains that there are known vulnerabilities, for example if you run a third party application within the browser. The third party application may grant the examinee access to local computer files. The software was mainly created for virtual clients, but features to secure a local BYOD environment have been added along the way. SEB can be, and is being, used for different digital exam setups. It is used to secure a lo environment on a BYOD computer, either through a bootable OS device or by creating a new desktop on the local computer OS. It can also be used in different virtual desktop infrastructures, either on student or University owned equipment. NTNU uses a solution provided by Inspera, which uses SEB, installed on the computers existing OS, to access the learning management system, Inspera Assessment. Monitoring of activities on a personal computer during a BYOD exam is a legal question in Norway. There are no current law to restrict or allow this, but in general it is only recommended to use surveillance for technical support purposes. Examinees in Norway would at least have to sign a written agreement to allow such monitoring [15].

2.7 Prestudy

I have previously conducted a preliminary penetration test on NTNU's chosen exam software in a lab setting. I focused on cheating during an exam, not prior or post exam. The focus was to test possibilities to access resources or to communicate using the BYOD computer, while running SEB. The results are summarized in table A.1. There were some successful attacks that have been reported to the stakeholders. Some attacks were unsuccessful, implying that the solution protects against those security threats. Attacks related to modifying source code, and configuration settings where difficult to confirm without the use of a LMS. A LMS which has the functionality to work with SEB should check the software version and configuration settings, by using a browser exam key [43]. During the prestudy, I tested the security of SEB without a LMS and was not able to test the functionality of the browser exam key. The tests were conducted in a lab environment with up to two laptops (See figure 2.1). The fact that some of these tests where successful in this test environment, does not mean they will be possible during an exam. The tests do not consider network access restrictions or monitoring, invigilators, what other students might notice, or security functions provided by the LMS and the system operations.

The tests that were unsuccessful, was to share the screen using a chat program, running on virtual machine and the use of clipboard, by coping and pasting crib notes. This does not mean that these threats are not viable, there might exist more advanced technologies that could bypass existing security features of SEB. The virtual machine detection settings can be changed, to be able to use SEB in virtual desktop infrastructure exams. The settings can be set to either exit SEB if a VM is detected, to give a warning that a VM is detected, or to run with not restrictions on the VM. SEB does not disable the copy/paste shortcuts for Windows machines, but any clipboard content that was copied before starting SEB is deleted, it is still possible to copy/paste within the exam environment.

The unconfirmed tests, shown in table 2.1, was not possible to test sufficiently without a way to test the browser exam key functionality in practice. This can be

Confirmed	**Unsuccessful**	**Unconfirmed**
USB key injector	Run exam environment on virtual machine	Modify source code to access resources and/or communication tools
Run SEB on remote computer to access resources	Use clipboard to access crib notes	Modify configuration settings to access resources and/or communication tools
Accomplice accesses exam environment by connecting remotely	Use a sharing desktop tool, in the background, to get assistance	
Audio/video communication in the background		

Table 2.1: Overview of tests

tested by using a LMS, with a test exam and configuration file setup.

The USB key injector is hard to mitigate without disabling external keyboards, as it is perceived by the computer as a regular keyboard. It will be possible to avoid by ensuring that invigilators will discover and confiscate any USB flash drives that are being used. An external keyboard could maybe be modified to be a USB key injector, which has the potential to make this attack even harder to detect.

The remote desktop connection attacks were successful, but could easily be mitigated by surveillance and/or restricting the Internet access during the exam. Network restrictions can also mitigate the audio/video communication attack. Another method to mitigate communication could be to implement the exam environment to block web camera, microphone and audio on the examinee's computer.

New technologies for malicious purposes are continuously being developed and if digital exams become more normal, more people will start inventing and investigating different cheating tactics, using technology. This is similar to the race between software developers and malicious attackers. "As better defensive measures get deployed, more sophisticated attacks are developed, leading to an endless arms race and an increasing complex system" [28].

2.8 Penetration testing

The most frequently used practice for software security is penetration testing [7]. Testing is used to ensure sufficient security throughout a system's life cycle, and enables developers to get a better view of how an attacker can exploit existing vulnerabilities. It is also a tool to help identify, assess and mitigate the vulnerabilities in a system. There are no common standard process to software security assessment, and practices varies in method and frequence. Akin et al. [7] discuss a better approach towards creating a secure system. They suggest to apply well known approaches in a software development life cycle, which include abuse cases, security requirements risk analysis and penetration testing. To continue to apply security testing, both automated and manual, to find and review existing and future security flaws in an application. Penetration testing can be made more effective by basing it on well structured methods and approaches to discover vulnerabilities, threats and risks.

Wang, Wong and Xu [53] present a threat model driven approach to detect undesirable threat behavior at runtime. They suggest to us Unified Modeling Language (UML)sequence diagrams to model the threats towards an application's security. Then use these models to make test cases and identify threats that are present in the system.

2.9 HARM

The original HARM method was presented as a combination of well-known modern security modeling techniques, used to represent complex and creative hacker attacks from multiple perspectives [24]. The HARM method intend to create and overview of the attack surface, by using modeling techniques such as for example Attack Sequence Descriptions (ASDs), Attack Defence Trees (ADTs), Misuse Case Maps (MUCMs) and Misuse Sequence Diagrams (MUSDs). Firstly, the outline of the intrusion is described with ASDs and MUSDs representations. Then, for more details, the attacking activities should be modeled as MUCMs, but one can also use the same modeling approaches that was used for the initial step. The functional context should be represented by misuse case diagram and attack trees.

Vegendla et al. propose to extend the HARM method to help plan penetration tests [52]. Their approach is to use predefined security requirements to develop and model figures, and then use the models to determine the vulnerabilities. These vulnerabilities are meant to be used to develop the penetration test cases. They present different approaches, either a top-down or bottom-up approach, or a free combination os these. The top-down approach is to use each security requirement to develop attack trees, and then use these trees to make misuse case diagrams, ASDs, MUCMs and MUSDs. The bottom down approach is to first use existing attacks, aimed at the security requirements, and represent them in diagrams. Then these attack are categorized and used to create an attack tree [52]

In this thesis, I will apply the bottom-approach proposed by Vegendla et al. In the prestudy [46], I defined attack goals, which can be redefined into security requirements. I will model the attacks that were performed in the prestudy to create a basis for potential attacks. I will modify the unsuccessful attacks, to make them more likely to succeed in the new test environment. The remaining attacks should also be adjusted, to target the changed attack surface.

Attack trees have the potential to present new ideas for attack approaches. Kordy et al. [28] suggest a different attack tree modeling approach, by adding mitigation strategies. They call these models for attack defense trees, and help to create a better overview for mitigation strategies. The main difference between the original attack trees approach, is the addition of defense nodes, and it becomes easier to see how to mitigate vulnerabilities at a higher level. For example if an examinee should attempt to use Skype, to communicate during an exam, SEB has the possibility to block Skype. In an attack defense tree, which makes visible, that it is more efficient to mitigate several communication attacks by restricting the overall Internet access in the exam venue. I will use attack defense trees instead of regular attack trees, to also be able to investigate mitigating possibilities.

Chapter 3

SEB: Code and Architectural Overview

In this chapter I present the code and architecture of Safe Exam Browser. I have read through user manuals, developer guides and documentation regarding SEB, and I have downloaded and explored the source code. The first section presents SEB, what it can provide and how to use it. I also describe the different log files that SEB generates and what they contain. The second section gives an architectural overview, to make it easier to understand what the different elements of SEB provide to the overall application. Then sections 3, 4 and 5 illustrate the architecture and functionalities of the three sub applications, that are parts of the default SEB installation package. Section 3 describes the SEB Windows service and registry resetter, which is used to edit registry settings to lock/unlock certain Windows options that can potentially compromise the locked-down environment. Section 4 describes the SEB ConfigTool user interface, and section 5 illustrates the ConfigTool's architecture and functionalities. The main SEB application's architecture, and how it benefits from the sub application is described in section 6.

3.1 General

The source code for SEB is located on GitHub [14] and it is free to view and download. In reality, SEB consists of 4 main applications, that cooperate and take advantage of each other to provide a secured lock-down browser application. The 4 applications are SebRegistryResetter, SebWindowsClient, SebWindowsConfig and SebWindowsServiceWCF, but I will refer to them as SEB Resetter, SEB Client, SEB Config and SEB Service, respectively.

The SEB Service provides a service that is permanently running on the host computer and the SEB lock-down browser normally will not run unless the service is running. The main task of the SEB Service is to provide the registry editing, that is needed to ensure a secured locked down environment for SEB. The SEB Service will be discussed further in section 3.3.

When SEB is initiated, it sends a signal to the SEB Service and tells it to change the registry so that for example access to the task manager is disabled. When SEB is exited, it asks the SEB Service to change the registry back to normal. There has been an issue with this when for example SEB crashes, and the registry is not reset. That is the reason for the SEB Resetter, which is now included in the SEB installation. It is a simple command prompt application that changes the registry back to normal, by calling the SEB Service.

The SEB Config contains the source code for the SEB ConfigTool. This aspect of SEB will be discussed in detail in section 3.5. Finally we will review the code in SEB Client, which contains the main application. This is the part of the application that currently uses the browser engine Gecko, by Mozilla, and takes advantage of the small applications previously mentioned; SEB Service and SEB Config. This part will be described in section 3.6.

The configuration files that can be generated with the SEB ConfigTool has the file ending '.seb', and can be used to initiate SEB. SEB ConfigTool presents the option to encrypt this file, which is recommended to avoid student tampering. An unencrypted version of the configuration file is in the form of XML, and an example can be found in B.2. It shows the SEB version, the startup url and the browser exam key, and other settings elements on XML format.

In a legal report [15], the subject of monitoring devices during digital exams is discussed, particularly with regard to BYOD exams. The discussion concludes that the surveillance of students' computers are only allowed for technical purposes, for example to discover what went wrong if something crashed. Norwegian laws are unclear on whether student owned laptops can be monitored legally, but the report does not recommend it from a legal standpoint [15]. With regard to network usage, the rules of surveillance are different, because of the fact that the network (usually) is the property of the institution that organize the exam.

SEB generates a number of different log files, that could be used for this purpose. Appendix C contains examples of the content in different logs that SEB generates. There are 4 type of log files, 'seb.log', 'SebClient.log', 'sebswindowservice.log' and 'SebConfig.log'. The 'seb.log' file, shown in C.1, contain the browser log. This log shows that after a force shutdown signal has been sent, the browser attempts to clear all of the user data. There are limited data that could be used to prove any malicious use during an exam session. The 'SebClient.log', on the other hand, provides information on different events that are initiated by SEB(see C.2). As an example, it shows the attempt to load the settings file, where it is stored and what it is called. It also shows the computers user name, OS, computer model, and that the clipboard has been cleaned, which is only parts of the information. The service log saves when and what have been changed in the Windows registry,

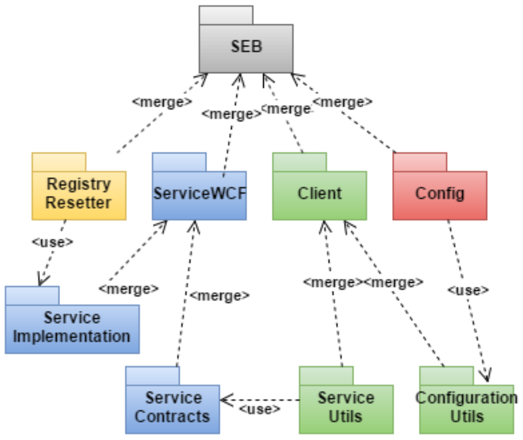


Figure 3.1: An overview of the dependencies between the different SEB applications

an example can be found in C.4. The final log file, 'SebConfig.log'(see C.3) does, as far as I can see, only contain the date and time of when a SEB session has loaded a configuration file (usually the default). These log files are stored on the host computer, and has to be manually sent from an examinee to the exam and/or technical staff for them to review the logs.

3.2 Architectural Overview

In this section, I will present an overview of the architecture of SEB, and the main dependencies between the different partitions. I will also lay the groundwork for understanding the models used in the rest of this chapter. The following models in this chapter are simplified UML class and/or package diagrams, to formally document and explain the architecture of SEB. The models does not contain all of the packages, files, classes and functions in SEB, but should represent the functionalities and architecture in an understandable way. The application has a complex code basis, with many dependencies, that are not significant for understanding the architecture of the cheating mitigation mechanisms. Some of the methods and dependencies have been simplified, or excluded, to reduce the complexity of the models. Some of the names of attributes within different classes are marked with quotation marks("") to emphasize that they are group descriptions for several similar attributes that are present in the class.

Figure 3.1 shows a simplified model to give an overview of the application. As I discussed in the previous section, SEB consists of 4 applications. These applications are organized as packages, similar to how it is displayed in figure 3.1. I have chosen to represent the different applications in different colors, to emphasize which sub-package is a partition of which application package. The ServiceWCF application package, represented in blue, has two sub-packages that have been modeled. The ServiceImplementation package is used by the RegistryResetter(yellow), to edit the registry. The ServiceContracts package is used by the ServiceUtils, to signal the WCF service. The Config application package uses the ConfigurationUtils(red) package to access and manage the configuration settings. The main application, that contains the Lock-Down Browser, is in the Client application package. This is the biggest application, and has several sub-packages that are not represented in this overview. The Client application will be discussed in more detail in section 3.6. Figure 3.1 shows the two sub-packages of the client application(green), that are used to communicate with other SEB applications. The registry application is a simple console application that calls methods in the Service application to provide functionality. The Config application is merely a user interface, that calls on methods from the main Client application.

3.3 SEB Windows Service and Registry Resetter

SEB creates and maintains a Windows Communication Foundation (WCF) service, that is hosted as a Windows Service. It uses the ServiceBase class in the System.ServiceProcess package, provided by the .NET framework. This service runs permanently as one of your computer's services, and is named the 'SEB Windows Service'. The service can modify your registry, C.4 shows an overview of which registry entries and values are being modified. It disables the possibility to lock the work station, access to the task manager, changing of passwords, user switching, log out and close. The entries that are named EnableShade and EnableShadeHorizon disable the drop down menu at the top of the screen if the exam is, for example, conducted with VDI. There are two entries for this to ensure that it covers different OS versions and different VM tools. They input the same value and variable, but they use different paths. An example of a registry entry path is: 'HKEY_USERS{UserID}\Software\Policies\VMware, Inc.\VMware VDM\Client' The EaseOfAccess entry controls the ease of access option on the log on screen of a virtual workspaces. These three registry settings are not relevant for an environment that is not created with some sort of virtual machine or client.

The ServiceBase class provides functions and methods to help run a service, and the main SEB Service class (SebWindowsServiceWCF) inherits from the service base class. The SEB Service class creates a servicehost, which is used by the ServiceBase. The servicehost uses the RegistryService class to define the service type, and a service contract, which consists of the IRegistryServiceContract interface, and the RegistryIdentifiers enum. The RegistryIdentifiers are used to identify separate RegistryEntry object, for example 'RegNoLogOff' and 'RegNoClose'. These objects represents specific registry items, and can enable or disable their system settings to increase the security of the locked-down exam environment. The registry editing methods inherit the RegistryEntry class. The enum identifiers and RegistryEntry objects are stored in a dictionary, which again is stored as a file. The file content is accessed and edited by methods in the PersistenRegistryFile

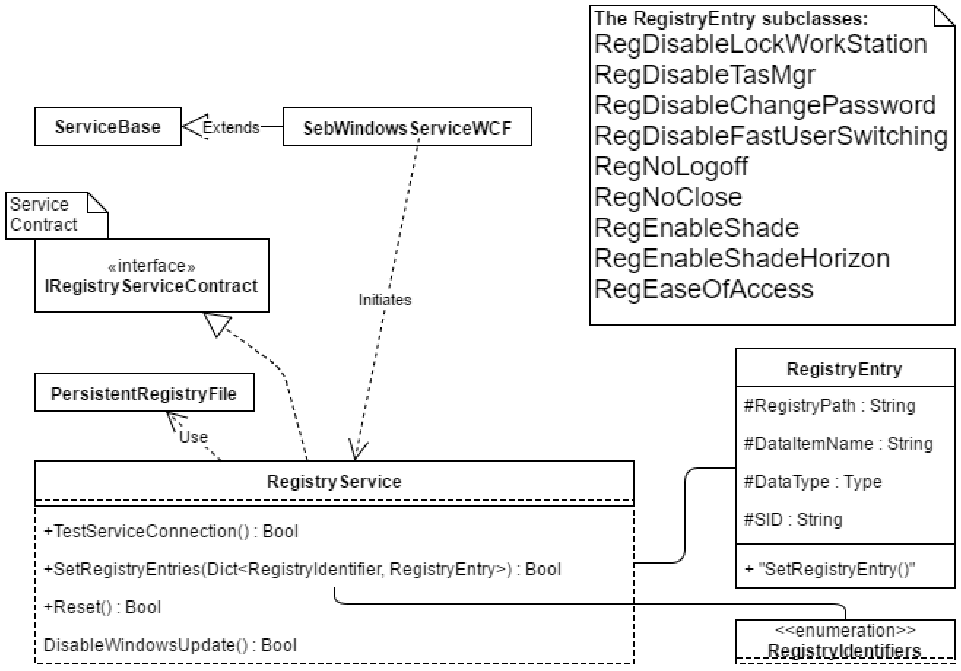


Figure 3.2: A class diagram representation of the architecture behind SEB Service

class, which is controlled by the RegistryService class. Figure 3.2 shows a model of the the architecture of the SEB Service in a simplified class diagram.

The Service also provides features for disabling and enabling of Windows Update, and for resetting the registry to its default Windows settings. The SEB Resetter consists of only one main class and that is it. It takes advantage of the functionalities of the SEB Service, to reset the host computer registry settings to its original default state. It is a simple, console application, that needs to be run in administrator mode. First it attempts to locate the file created by the PersistentRegistryFile, and use the already existing functions to reset the registry values, and enable Windows Update.

3.4 SEB ConfigTool

In this section I will go through the different settings that can be decided, created and distributed to examinees, prior to an exam. The configurations can be changed to better suit the exam environment that SEB is used with, for example thin clients and/or for BYOD exams. I will use the SEB ConfigTool as a graphical representation of the contents in a configuration file. B.1 in the appendix contains screenshots from each of the categories (represented as tabs) of the ConfigTool.

The General tab, shown in figure B.1, contains a start URL, administrator and quit passwords and exit settings. The start URL is the Internet address of the first page that SEB visits when it is started with these settings. This should contain the address of the LMS exam. The administrator password is used to open the settings file with the ConfigTool, to be able to edit the settings. This is to protect the settings from being modified by the examinees, as they need to download a copy of the settings file. There is also an option as to whether or not the examinee should be able to quit SEB during an exam, this could give access to all underlying resources. An examinee should be able to exit the program when he/she delivers, this could be possible to control by setting a quit password, which have to be correct for the student to be able to exit the program. It is possible to set a sequence of keys, that will exit the browser, but SEB also has an exit button in the user interface for this purpose.

Figure B.2 contains settings regarding the configuration file. The administrator needs to chose what the settings file should be used for, to either start an exam or to configure a client. This is also were the encryption settings of the file can be set, and an decryption password needs to be set. This password is prompted for when starting an instance of SEB with these settings. There are also some options for saving, loading and use the current settings, as well as to revert them to default values.

The user interface(see figure B.3) can be configured for specific needs, for example to adjust the screen size and the task bar menu options. Figure B.4 shows the browser options. Plug-ins can be enabled and disabled, such as javaScript. The user agent is by default set to Mozilla's Gecko, but this can be changed in the configuration file. It is also possible to disable the entire browser application of SEB and only use the kiosk environment to run third party applications, for example a thin client application or a text editor.

Most of the Down/uploads settings in figure B.5 are only available for Mac OS and I will not discuss these as I am mainly working in Windows OS. The only available Windows option is whether to allow the download of a new SEB configuration file, using the SEB browser. The exam tab (see figure B.6)shows the browser exam key, which has to be copied into the settings of the SEB plug-in for the exam in the LMS. This key is generated by the ConfigTool, and should not be edited manually. There are also some options to include different URLs to, for example, restart and quit SEB. When an examinee clicks on a link(within the LMS) that equals the quit URL stated here, SEB will quit automatically.

The application tab (see figure B.7) presents settings regarding the management of processes on the host computer. The settings can be set to allow or prohibit given processes from running in the background. By default, these setting permits XULRunner to run in Windows, to be able to run the browser engine Gecko that is used by the SEB browser. These settings also include options for third party applications. The network settings, in figure B.8, allows to blacklist and whitelist certain URL's. This feature enables the exam administration to allow access to certain allowed resources, while at the same time ensuring no malicious network usage. This tab also manages network proxies and certificates.

The security settings, managed in the security tab in figure B.9, presents the possibility of allowing, for example, the use of a virtual machine. The administrator can also manage the logging options. In this tab, one can choose which type of kiosk mode to use. It can create an entirely new Windows desktop, or just disable the explorer shell. The explorer shell is the menu with tabs to open running applications and the Windows start menu. There is also an option to not use kiosk mode at all. This setting is mainly designed for debugging. Figure B.10 showa which registry options that should be disabled/enabled by the SEB Service. Finally, the hooked keys tab (figure B.11) allows management the use of special keys, key shortcuts and function keys.

3.5 SEB Configuration settings

A relatively new feature of SEB is the SEB ConfigTool, witch assists exam distributors and operators to create a configuration file for SEB without having extensive technological skills. Most of the logic behind the SEB ConfigTool and the configuration file is located in the main SEB Client application. In this section we will look into this logic, and also give an overview of the different settings that are used, to ensure a secured exam environment.

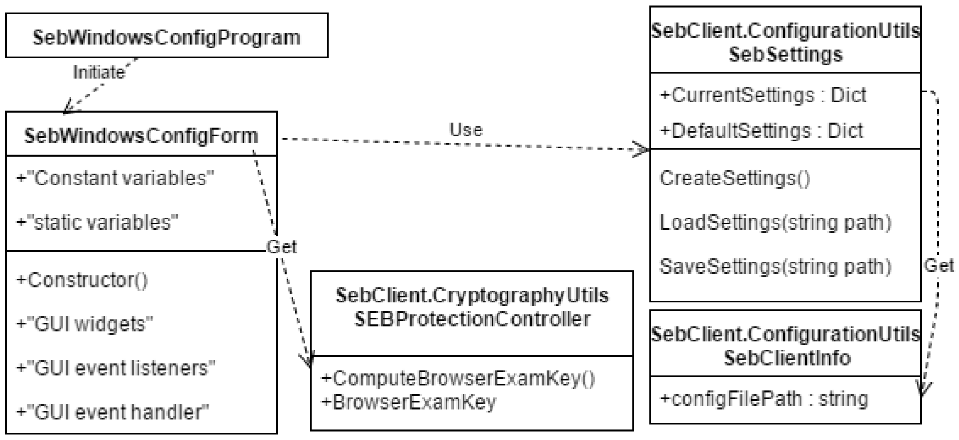


Figure 3.3: A class diagram representation of the architecture behind SEB Config

The source code folder called SebWindowsConfig, contains the GUI utilities for the user interface of SEB ConfigTool. There is one main class (SebWindowsConfigProgram), that initiates the program and calls the SebWindowsConfigForm. The ConfigForm class contains all of the GUI objects, with their global and static variables. When the SEB ConfigTool is initiated, the application 'asks' the client application to provide either the default settings or upload the settings of a given settings file. This is provided by the SebSettings class in the SEB client application, as shown in figure 3.3. The SebSetting class maintains two versions of the settings, one with default and one with current values. The configuration tool stores the changed settings by changing the values of the current settings in SebSettings, and use SebSettings methods for storing, either encrypted or unencrypted XML in a .seb file. The SebClientInfo class, also from SEB client, provides the configuration application with paths to stored files and log files(String Constants), together with info about the SEB version and methods to help with logging. Finally, as shown in figure 3.3, the configuration application generates and stores a version of the browser exam key. This is done by using the SEBProtectionController class, also provided by the client application. The controller gets the settings in XML format, from SebSettings.CurrentSettings, and uses this, together with the SEB execution file to compute a browser exam key.

3.6 SEB Client Application

SebWindowsClient is the main application included with SEB. This application provides the Lock-Down Browser, by locking down the host computer and opening a browser. The other applications of SEB are merely tools to provide security to this LDB. The previous sections in this chapter should have created a basis to

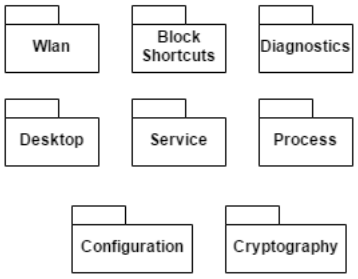


Figure 3.4: The utilities (utils) of the SEB main application, also called SEB Client

better understand the main application. SEB Client consists of many packages, most of them in the form of utilities, or tools. There are 8 utility packages in the Client application, and they are shown in figure 3.4. Some of these utility packages provide general functionalities, such as Wlan, which is conserned with wireless networking. I will focus, primarily, on utilities that provide security features to the Lock-Down Browser environment. These utilities include BlockShortcuts, Service, Process, Desktop, Configuration and Cryptography. I will not discuss the GUI implementation the gecko browser and other functionalities that are not relevant for securing the exam environment from cheating examinees.

The BlockShortcutsUtils package, contains methods for filtering keys, shortcut keys and special keys, that could potentially compromise the environment. The ServiceUtils package consists of a service controller, to communicate with and execute methods provided by the SEB Service. The ProcessUtils package manages the underlying processes, with regard to which processes are prohibited/allowed in the current settings. The ProcessUtils also controls the native Windows clipboard manager. The cryptographyUtils package includes methods for encryption and decryption of files and passwords. This also includes computing the browser exam key and managing encryption certificates. The ConfigurationUtils package manages the configuration setting. It can load and store configuration files, generate an XML string version of the settings. Manage constants, for example file path strings, configuration entries and software information. The DesktopUtils package enable and manages the kiosk mode. The settings for kiosk mode can be to either create a new desktop or to disable the explorer shell, which we explained in section 3.4.

The SEB client application is a complex application and I will attempt to present an architectural overview of the main functionality and aspects that could prove to be relevant for the penetration testing that I will to conduct. The different utility packages presented in figure 3.4, each contain classes and functionalities that are correspondent with their package names.

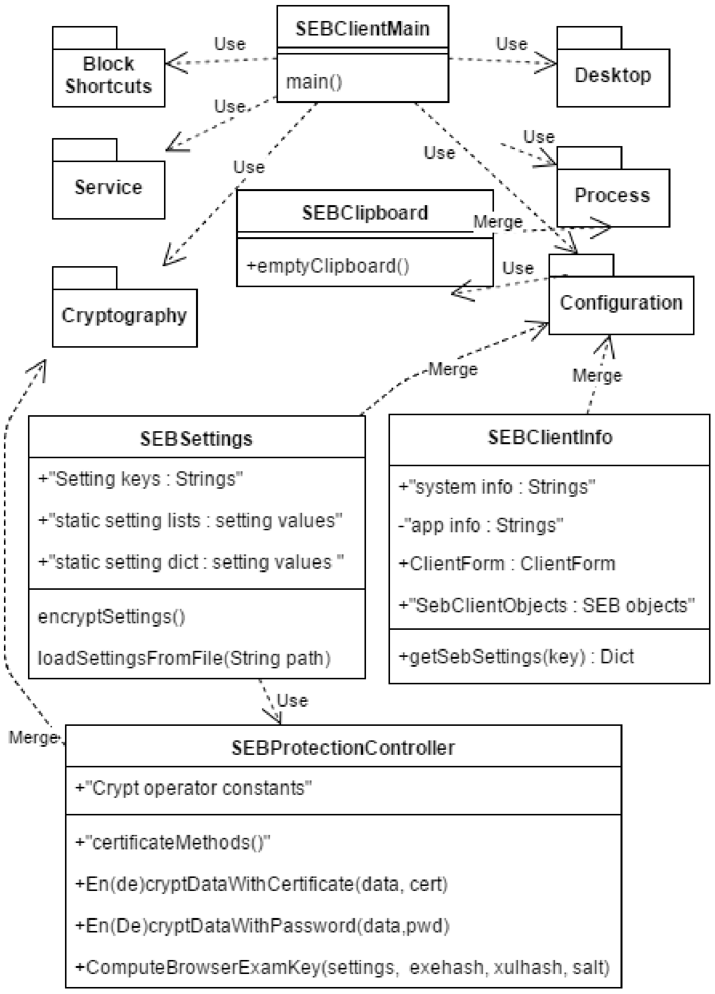

SEBClientMain
main()
Use
Block Shortcuts
Desktop
Service
Process
Cryptography
SEBClipboard
+emptyClipboard()
Merge
Configuration
SEBSettings
+"Setting keys : Strings"
+"static setting lists : setting values"
+"static setting dict : setting values "
encryptSettings()
loadSettingsFromFile(String path)
SEBClientInfo
+"system info : Strings"
-"app info : Strings"
+ClientForm : ClientForm
+"SebClientObjects : SEB objects"
+getSebSettings(key) : Dict
SEBProtectionController
+"Crypt operator constants"
+"certificateMethods()"
+En(de)cryptDataWithCertificate(data, cert)
+En(De)cryptDataWithPassword(data,pwd)
+ComputeBrowserExamKey(settings, exehash, xulhash, salt)

Figure 3.5: A simplified representation of the SEB Client application

The CryptographyUtils package consists of one class, called SEBProtectionController. This class manages the cryptography aspects and provides methods to help with encryption and decryption. It manages certificates and encryption and decryption with these certificates. It provides methods for encrypting and decrypting with a passwoerd. Finally it computes the browser exam key, using the current SEB settings, the SEB execution file, the XULRunner configurations for SEB and a salt that is stored in the settings.

The DesktopUtils package consists of two classes that help with managing the SEB desktop, and uses methods from the .NET system to do this. In the description in the code, the main objective is to encapsulate the Desktop API. The ProcessUtils package handles the allowed and prohibited lists described in section 3.4, as a part of the application tab. These are lists for allowing or blocking processes, executions and third party applications. It monitors, restricts and disables outlying resources. The ProcessUtils package also includes the class called SEBClipboard, which contains a method for emptying the clipboard content.

The ServiceUtils package contains a class called SEBWindowsServiceController, which initiates and checks the SEB Windows Service, and tells it how to edit the registry. The BlockShortcutUtils package tracks, captures and blocks key sequences and shortcuts for executing any outside resources.

The configurationUtils package contain several classes, of which 2 are worth some discussion. These are the SEBSettings and the SEBClientInfo classes (see figure 3.5). The SEBSettings class has already been shown in figure 3.3, to explain its contribution to the functionality in the SEB ConfigTool. For the client application, the SEBSettings class keeps all the settings and key strings as constants or static values, ordered and maintained in lists and dictionaries, so that other classes can access this information. It also loads and encrypts the settings from file. SEB access decryption tools from the CryptographyUtils class called SEBProtectionController. The SEBClientInfo class stores private application information as constants as well as more dynamic values, strings and objects which provides and maintains data for the application, other than settings. The clientInfo keeps track of system specific strings and application objects, for exampel the ClientForm object. The ClientInfo class gives access to settings of SEBSettings, by using a getSebSetting dictionary, that can access certain setting values from the SEBSettings class, based on their key string, stored as constants in SEBSettings. SEBClientInfo also manages DesktopUtils, process, execution and window handling, the browser engine configurations and logging.

Figure 3.5 shows a simplified version of the SEB client application architecture, with only a sample of the utilities, classes and dependencies that are present in the client application. All of the different utilities are used by the SEBWindowsClientMain to create the strict locked-down digital exam environment. These descriptions and figures are made to describe the security functionality, and I have not included

GUI aspects and the logic behind using a bowser engine, how the program uses Internet access, and/or restricts the Internet access to conform with some of the network settings that can be given with the SEB ConfigTool. The Internet settings provided by SEB can only limit the access from within the locked-down browser, once an examinee escapes SEB, the Internet settings in SEB can not control Internet access for a regular browser on the same computer.

Chapter 4

HARM, Vulnerabilities and Risks

This chapter will describe how HARM was used to develop penetration test cases, by identifying vulnerabilities and risks. The first section describes the security requirements and how HARM is used. The second section will explain the attacks and the models of the attacks from the prestudy, and which vulnerabilities the different attacks can potentially compromise. This section will also include attack defense trees for each of the security requirements. Section 3 will summarize the vulnerabilities that were discovered, how they can be mitigated and what kind of threats they include. The vulnerabilities and threats will be discussed further, to create penetration test cases, which are also presented in section 3. The fourth section discuss the risk assessment of the different threats that have been discovered.

4.1 Introduction

The prestudy (see section 2.7) describes the planning and execution of some simple penetration tests of SEB without a LMS. These tests are based on the premise that to cheat during an exam is mainly about getting access to prohibited resources and/or communication. This can be transferred into the following security requirements.

- **Security Requirement 1:** An examinee should not be able to access resources, unless specified.
- **Security Requirement 2:** An examinee should not be able to communicate with anyone, except for invigilators and other exam related personnel.

Hacker Attack Representation Method (HARM) [24] is a framework to help model and represent hacker attacks, to understand the present vulnerabilities. The extension of HARM, presented by Vegendla et al. [52] describes how to take advantage of HARM for penetration test planning (see section 2.9). It will help us to model the threats and attacks to be performed, and to brainstorm for further related vulnerabilities. The security requirements was used in the prestudy to derive the different attacks, these attacks will be extended to also consider the change of test environment. The test environment will still be a lab setting, but we introduce Moodle, with a SEB plug-in which adds security functionality. We will use the browser exam key feature with Moodle, and create test exams with similar security as for a real life exam. The main elements, which is important in an exam environment that we will not test, is detectability and network restrictions. There is always a risk that invigilators or other examinees detect suspicious activities during an exam, and report it. An exam environment should have restricted Internet access to prevent access to prohibited resources and communication. I will not attempt to evaluate such mitigation strategies, but they are important to consider for the purpose of transferring the test results to realistic vulnerabilities of digital exams with SEB.

The results from the prestudy can be divided into three categories, confirmed, unsuccessful and unconfirmed, see table 2.1. I will investigate whether cheating behaviors are prevented, logged by Moodle or SEB or by any changes in the configuration settings. The previous, unsuccessful attacks will be reviewed to see if any new insights of the source code and Moodle will help us make improved and more sophisticated attacks. I will attempt to use new approaches to the unconfirmed tests, which I was unable to test sufficiently in the prestudy, due to the test environment being too simple. These attacks are angled towards the possibility to modify source code and/or the configuration settings, while still getting access to the exam. According to Schneider [43, 44], a LMS with support for Safe Exam Browser, will detect that the correct configuration settings and SEB version, with unmodified software, are used, by comparing browser exam keys sent in the URL-header. This will be possible to test by using Moodle and SEB together.

4.2 Background for penetration test cases

I will use extended HARM [52], described in section 2.9, to develop penetration test cases. The extended method includes a bottom-down approach, which is to initiate penetration test planning, by using modeling of specific attacks. For each security requirement, one should describe the specific attacks that can be thought of, with Attack Sequence Descriptions (ASDs) first, and then with Misuse Case Maps (MUCMs) and/or Misuse Sequence Diagrams (MUSDs). The different attacks described can be categorized and grouped together in an Attack Defence Tree, to see relationships between the different attacks and possible mitigation approaches. This attack tree could also be used to encourage new attack ideas. This different descriptions and models will help create a basis for making penetration test cases.

I used the security requirements stated in section 4.1 to develop and categorize the test cases in the prestudy. I will describe these test cases using ASDs, MUCMs and MUSDs and attempt to make an Attack Defence Tree (ADT) to categorize the attacks and suggest mitigation strategies, as proposed by HARM's bottom up approach.

4.2.1 Security Requirement 1: Prohibit accessing of resources

During an exam, there are strict rules on what resources an examinee should have access to, and the rules varies between different courses. Examinees could attempt to sneak crib notes containing parts of the curriculum into the exam location and peek at it when it seems necessary. The newer risks introduced with digitization, is that the amount of notes are no longer restricted by their physical size. There are suddenly no limit to the type and amount of resources that could be accessed on the computer that runs the exam environment, and it becomes that much more important that the exam program prevents such access.

The digital exam environment can be compromised on this regard in two ways; either by getting resources into the exam program, or by escaping the exam program to access local computer resources during an exam. SEB attempts to prevent such activities, and we will attempt to evaluate their effectiveness. In the prestudy, I used a USB key injector to inject crib notes into the exam environment, the Attack Sequence Description is located in table 4.1. In figure 4.1 you see a Misuse Case Map representation of the key injection attack. The key injector should be detected by the computer as an external keyboard, and the injected notes are registered as if someone is typing on this imaginary keyboard. Thus, it is difficult to implement preventive mechanisms into the exam software, without the risk of actually disabling all or several types of keyboards. Thus the USB-port provides a serious vulnerability(V1) that is difficult to mitigate. The second major vulnerability(V2) is that suddenly a big amount of text is inserted into the exam workspace, initially faster than it is possible to type manually, without being noticed. It should be possible to have mechanisms to detect such activities. On the other hand, it is also possible to write a script with typing delay, so that the notes are injected in approximated typing speed. Figure 4.2 shows the key injection attack as a Misuse Sequence Diagram, notice that there are no invigilator actor, as I have chosen not to investigate discovery by manual detection. The V1 vulnerability is the same as for the MUCM, which is concerned with detection of the key injector. V2 and V3 are similar to V2 in the MUCM, and consider detection of suspicious typing activities, either automatic or manual.

Another way to access resources from within the exam environment is to use the built in windows function called clipboard, which enables the user to move text and resources between and within applications. This was attempted in the prestudy, where it was discovered that SEB deletes the content of the clipboard when it is started. For this attack, one has to modify the source code of SEB so that the clipboard is not emptied. The ASD for this attack is shown in table 4.2. This attack has been modeled in a MUCM (see figure 4.3) to show the different elements of the attack. The first vulnerability (V1) is concerned with which modifications SEB can do to the clipboard content. SEB allows the use of clipboard within the exam

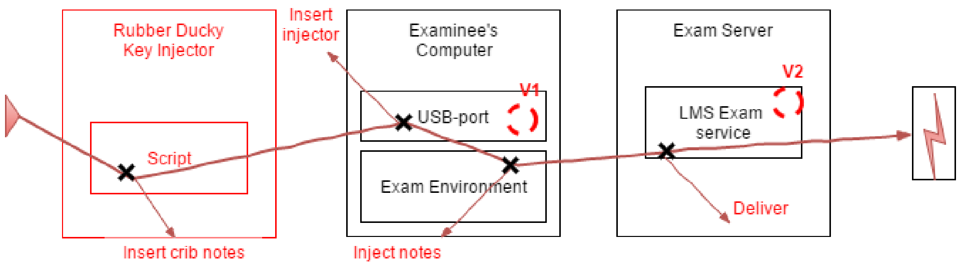


Figure 4.1: MUCM of key injector attack

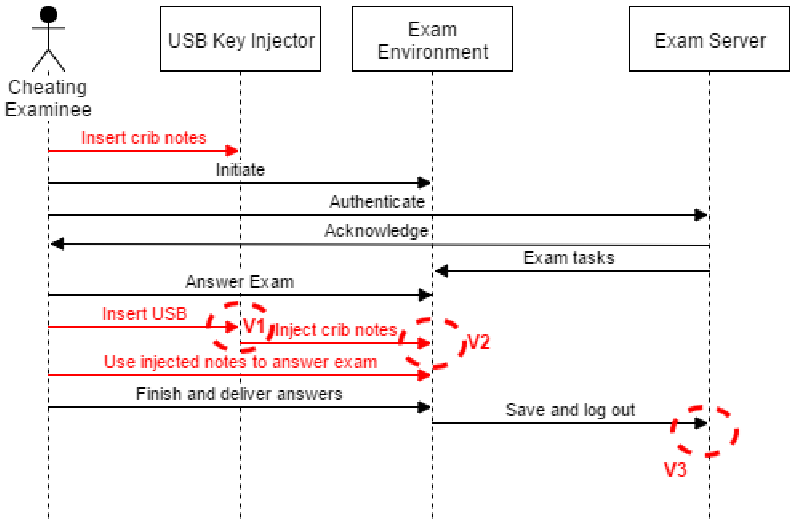


Figure 4.2: MUSD of key injection

environment, but after the content has been emptied. Can clipboard activities be prevented and/or detected? The second vulnerability (V2) is concerned with the browser exam key, which is used to protect against code modification, will the modifications done in the first step prevent the cheater access to the exam?

The attacks described here, are ways to insert prohibited resources into the exam environment, similar to bringing crib notes to a pen and paper exam. Further, I will look into ways to access resources that are already stored on the computer, but where SEB should prevent this. SEB is made to lock down your computer, so that you cannot access any locally stored resources, unless they are specified in the settings. In the configuration settings, you can choose whether to allow a student to, for example, run the exam environment inside a Virtual Machine (VM) or not. For exams at NTNU you should not be able to use VMs, because it allows you

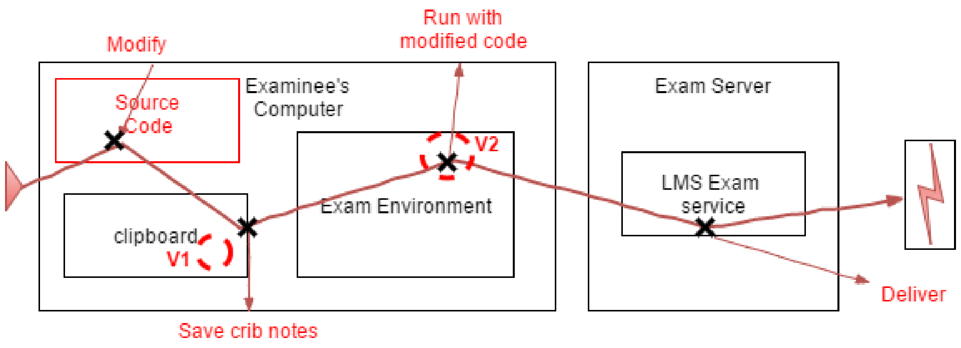


Figure 4.3: MUCM of using clipboard to inject crib notes

Steps	Description
1	Save injection script with crib notes onto USB injector => Prepared key injector
2	Start the Moodle localhost server => Can access Moodle
3	Open SEB with default settings => Exam started
4	Put USB injector into the computer's USB port => Crib notes are injected
5	Deliver exam and quit SEB => Finished

Table 4.1: An Attack Sequence Description (ASD) of the key injector attack, in a lab environment

Steps	Description
1	Modify source code => Will no longer empty out clipboard when SEB initiates
2	Copy crib notes(right click, copy) => Notes are saved onto clipboard
3	Start the Moodle localhost server => Can access Moodle
4	Open (modified) SEB, with default settings=> Exam started
5	Paste crib notes from clipboard into a text editor inside the exam (ctrl+v, right click should be disabled) =>The crib notes are in the exam text editor
5	Deliver exam and quit SEB=>Finished

Table 4.2: An ASD of using the clipboard to inject crib notes

to minimize the exam environment and access prohibited resources. What if you disable the VM blocker directly in the source code? Table 4.3 contains a Attack Sequence Description of this. Figure 4.4 shows the attack in a Misuse Case Map, with the most important vulnerability concerns. Vulnerability 1 represents that the browser exam key should detect that the SEB version has been modified, and disable access to the exam. Also, trained invigilators should be able to discover the suspicious activity on the examinees computer screen, while the VM is minimized, to access resources, which is shown as vulnerability 2. A clever examinee can make the outside environment look like the digital exam environment, to make the attack harder to spot. Finally, the exam service should be able to detect some sort of malicious activity. For example the network activity if the examinee access online resources from the local host computer.

Table 4.4 describes how to run the exam system on a remote computer, so that the examinee only needs to minimize the remote connection and access locally stored resources. Which is the same principle as for using a VM, without the risk of it being detected or blocked by the existing VM detection mechanisms of SEB. Figure 4.5 shows a Misuse Sequence Diagram of this attack. The examinee needs to have an extra computer, with Internet connection, at a remote location. This attack should get the same resource access as you get when you run the exam environment on a VM. The biggest vulnerability of this attack is the Internet access. For V1 and V2, the remote connection should be detected and prevented by the network administration, which is not directly a part of SEB's security mechanisms. Vulnerability 3 is that you should be spotted. It should not be possible to exit the entire exam environment without anyone noticing.

The last threat for this security requirement is a more general attack that enables a lot of different opportunities for a cheating examinee. As described in previous sections, there is a browser exam key, that prevents examinees from accessing the exam with the wrong configuration settings and/or modified SEB software. If we are able to bypass the browser exam key, by modifying the source code, a cheating examinee could do the exam with any form of configuration file, and code modifications that enable access to resources and communication. Table 4.5 shows a general Attack Sequence Description of such an attack. Figure 4.6 shows a Misuse Sequence Diagram of this attack, to give a better overview of the key elements and how the modifications made to bypass the security is planed. The vulnerabilities here are that the modifications are detected. Either by V1 and V2, by setting the wrong path in the source code, or overlooking some important code that makes sure these changes will work as suspected. Vulnerability 3 is concerned with that the expected browser exam key, matches the key generated by the modified code.

To follow the extended HARM bottom-up approach, these attacks have been categorized by their approaches and vulnerabilities, to make an Attack Defence Tree, shown in figure 4.7. It shows that there are several attacks that will attempt to modify the code , without being prohibited by the browser key mechanism, which

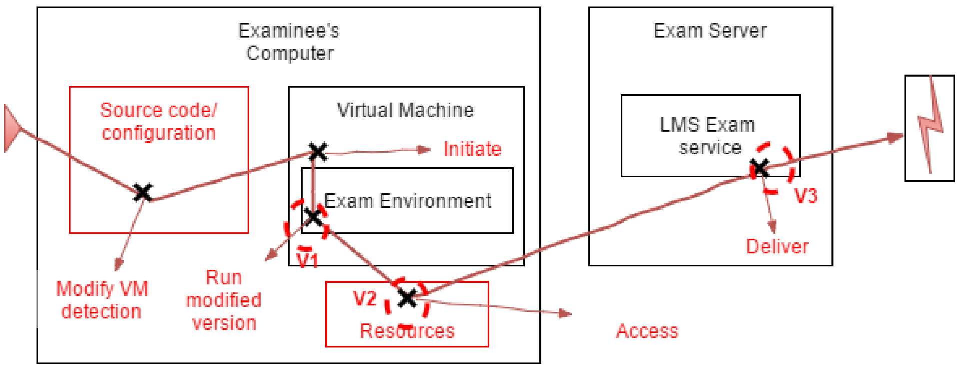


Figure 4.4: MUCM of running SEB on a Virtual Machine, to be able to access locally stored resources.

Steps	**Description**
1	Localize and disable the virtual machine detection functionality in the source code => Can run SEB on virtual machine
2	Start the Moodle localhost server => Can access Moodle
3	Start the virtual machine and send it the modified code=> Can run modified SEB on VM
4	Open (modified version of) SEB with default settings on virtual machine =>Exam started
5	Minimize the virtual machine (with SEB running) => Can access locally stored resources
6	Deliver exam and quit SEB on the VM => Finished

Table 4.3: An Attack Sequence Description (ASD) of how run SEB on a virtual machine to access local computer resources, in a lab environment.

Steps	**Description**
1	Initiate remote connection with remote computer =>Remote connection established
2	Start the Moodle localhost server(on remote computer) => Can access Moodle
3	Open SEB with default settings on remote computer => Exam started
4	Minimize the remote connection window(with SEB running) => Can access locally stored resources
5	Deliver exam and quit SEB on the remote computer => Finished

Table 4.4: An Attack Sequence Description (ASD) of using remote desktop to access local computer resources, in a lab environment

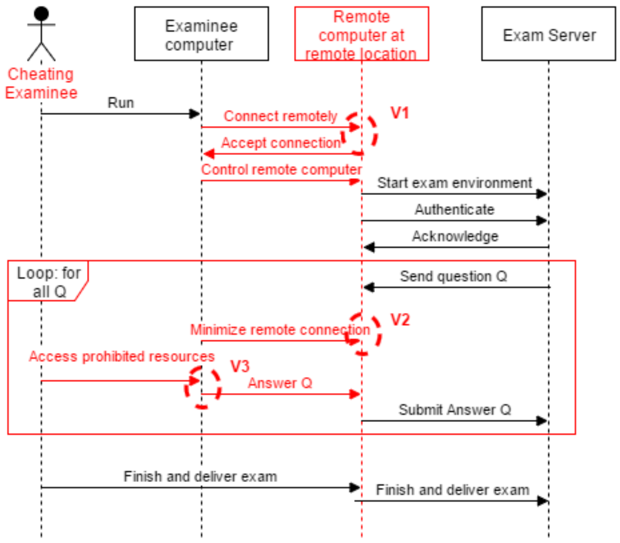


Figure 4.5: MUSD of running an exam on a remote computer to access resources

Steps	Description
1	Locate and analyze the Browser Exam key functionality in the source code => Creates a knowledge base to be able to modify the code so as to bypass this mechanism
Comment	The Browser Exam key is generated by using the (predefined)configuration file and the correct SEB execution file, using paths to the two files
2	Change the path of the configuration file in the source code, to the location where the predefined configuration file is located(NOT necessarily the configurations that are being used) => Can use self defined configurations to run the exam environment
3	Change the path of the execution file in the source code, to the location where the original approved execution file is located (NOT necessarily the execution file that are being used) => Can use modified version of the SEB software
4	Start the Moodle localhost server => Can access Moodle
5	Open SEB with configuration settings and (modified) execution file of your choice => Exam started
6	Deliver exam and quit SEB => Finished

Table 4.5: An Attack Sequence Description (ASD) of how to bypass the browser exam key security mechanism.

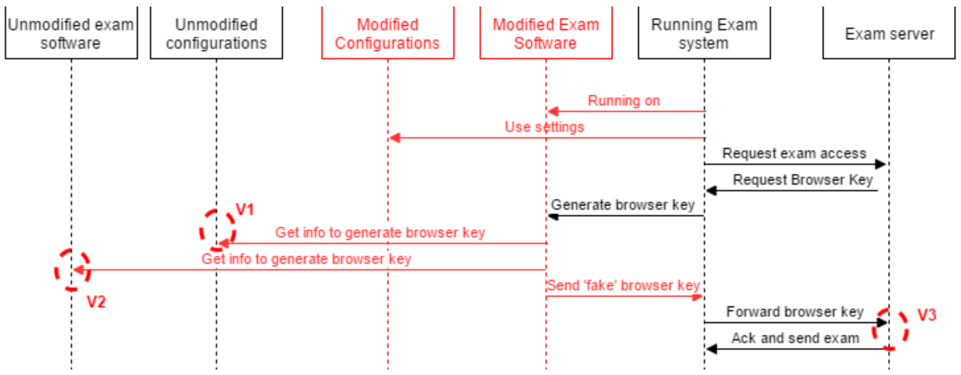


Figure 4.6: MUSD to show how to modify the source code to bypass the security mechanisms of the browser exam key.

we also have described an attack against. For the USB injection attack, there are no current protection mechanisms. The other countermeasure nodes have not been discovered as a part of our current test environment, and are only suggestions of ways to improve the security against cheating behaviors.

4.2.2 Security Requirement 2: Prohibit communication

In this section, I will discuss attacks that provide the cheating examinee with a way to communicate with others, preferably with an expert on the exam subject, to get assistance. This is also an existing security threat for paper based exams, but might be more challenging than, for example, to bring cheat notes. An examinee could maybe sneak in a mobile phone with a small, unnoticeable hearing device, to be able to hear the outside expert. Two or more examinees could also develop undetectable ways to communicate inside the exam venue, by using for example coughing or body language. They could even manage to meet up during synchronized, planned bathroom breaks. For this requirement I will focus the attacks on using the BYOD computer for the purpose of communicating. Even if an exam environment has restricted Internet access, an examinee could be able to access the Internet through an external, hidden device, such as a mobile phone. This is not a aspect of the security of SEB and will not be assessed in this thesis. Communication tools can be accessed on the BYOD computer, as a locally stored resource, which mean that some of the attacks for security requirement 1 also can be used to access communication tools. This includes all the attacks described for accessing underlying resources, not the attacks that insert crib notes into the exam environment. We will therefore focus these attacks on accessing communication, whilst running the exam environment.

Table 4.6 describes using a remote connection to communicate. This attack is similar to the remote connection attack, used to access resources. To gain resource

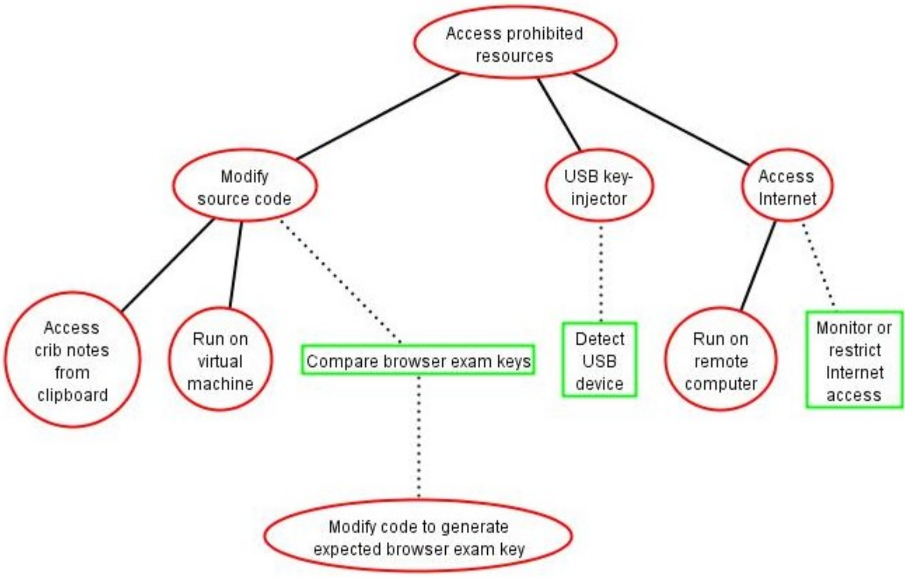

Access prohibited resources
Modify source code
USB key-injector
Access Internet
Access crib notes from clipboard
Run on virtual machine
Compare browser exam keys
Detect USB device
Run on remote computer
Monitor or restrict Internet access
Modify code to generate expected browser exam key

Figure 4.7: This is an Attack Defence Tree (ADT) [27], to show the relationships between different attacks with the aim to access prohibited resources.

access, the exam environment are run on a remotely located computer. The main difference in the communication attack is that the remote computer controls the computer used in the exam location, and that the remote computer is controlled by an accomplice. This means that for communication, the exam environment run on the examinee's computer at the exam venue. The fact that the examinee is always locked into the exam environment, even though he gets assistance, should make the suspicious activity harder to detect. Figure 4.8 is a Misuse Sequence Diagram that shows the attack in more details. The main vulnerability here is that network restrictions and/or surveillance should either detect the suspicious network activity or prevent the remote network connection from even being established.

Another approach to establish communication is to run a communication tool in the background of the exam environment. This tool should be initiated before the exam environment, to take advantage of communication with audio, video or even shared desktop. Table 4.7 shows an ASD of how to use communication tools in the background. The prestudy established that there are mechanisms in SEB that block desktop sharing, while SEB is running. This attack will attempt to disable these mechanisms. Figure 4.9 shows a detailed sequence diagram of this attack. Vulnerability 1 for this attack is that network access in the exam location should be restricted, so that no unknown network communication channels should be possible to establish. If the communication gets established, it should be detected by auto-

Steps	Description
1	Initiate remote connection from the accomplice's computer, to the examinee's computer => Remote connection established and accomplice is able to control examinee's computer
2	Start the Moodle localhost server (on examinee's computer) => Can access Moodle
3	Open SEB with default setting on examinee's computer => Exam started
4	Both examinee and accomplice can cooperate within the exam environment to answer the exam questions => Gets access to accomplices knowledge when answering the exam
5	Deliver exam and quit SEB => Finished

Table 4.6: An Attack Sequence Description (ASD) of how to use remote desktop to get assistance on an exam, in a lab environment

Steps	Description
1	Locate and disable the mechanisms that block shared desktop => Enables examinee to share exam tasks with outside accomplice
2	Establish a communication connection => Examinee can communicate (using audio and video) with accomplice
3	Activate the shared desktop option for the communication channel => Accomplice can view the examinee's computer screen
4	Start the Moodle localhost server => Can access Moodle
5	Open SEB with default settings => Exam started
6	Accomplice and Examinee communicate to finish the exam => Get outside assistance
7	Deliver exam and quits SEB => Finished

Table 4.7: An Attack Sequence Description (ASD) of how to communicate using audio/video communication and shared desktop with a background application, in a lab environment

matic network monitoring. The second vulnerability is that of the browser exam key, which should detect that the code has been modified, and thus deny access to the exam. Vulnerabilities 3 and 4 are based on the fact that SEB includes process monitoring and should be able to prevent the use of share desktop tools in a sufficient way. The audio and video based communication methods could also be used without the shared desktop since they are not blocked. This provides the cheating examinee with a different issue, which is how to communicate the exam questions to the examinee. The accomplice should be able to hear the examinee through the audio channel, but the examinee will most likely be discovered if he attempts to speak the exam questions in an invigilated exam. It could be possible to mumble or whisper the questions through a small microphone, if there already are background noises in the exam location. The accomplice can also see the examinee, while the examinee can only hear the accomplice(through an undetectable hearing device) and will see the exam environment. They can also attempt to use code signals and other creative methods for communicating without being detected.

Figure 4.10 shows the Attack Defence Tree that categorize the attacks on security requirement 2. Communication with an outsider is naturally dependent on

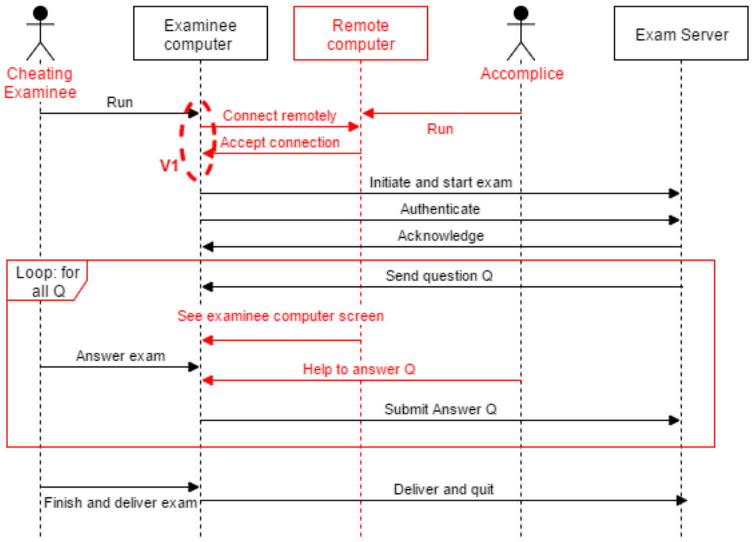


Figure 4.8: MUSD to show how to get outside assistance on the exam by using a remote connection tool.

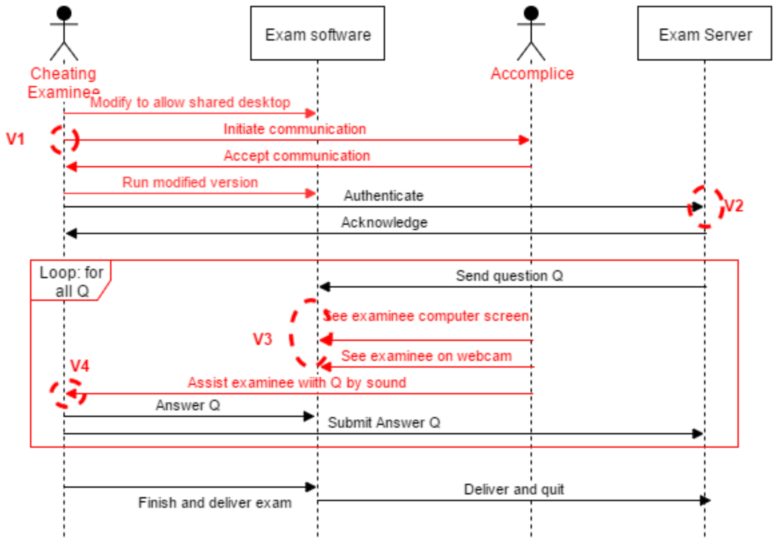


Figure 4.9: A MUSD to show how to get outside assistance by using shared desktop and audio/video communication

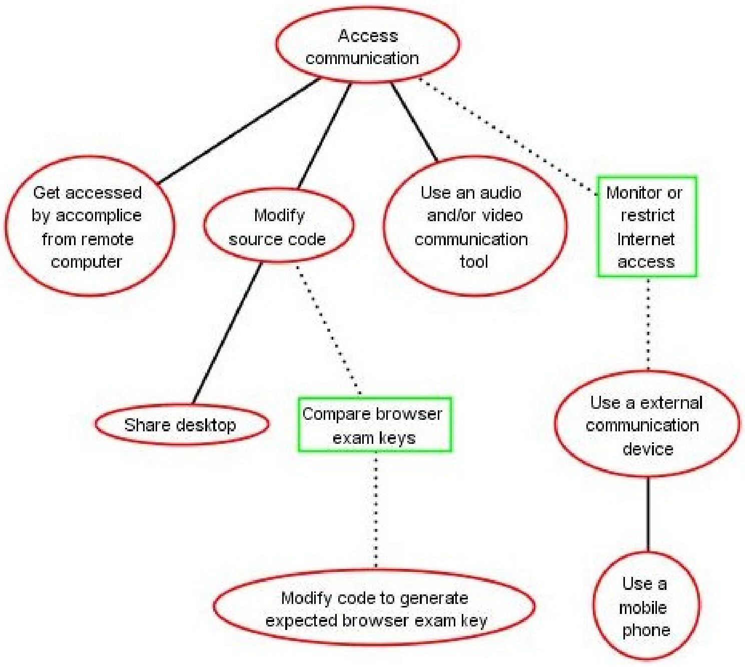

Access communication
Get accessed by accomplice from remote computer
Modify source code
Use an audio and/or video communication tool
Monitor or restrict Internet access
Share desktop
Compare browser exam keys
Use a external communication device
Modify code to generate expected browser exam key
Use a mobile phone

Figure 4.10: This is an Attack Defence Tree (ADT)[27], to show the familiarizations between different attacks with the aim to communicate during the exam.

the examinee being able to use the Internet connection at the exam venue freely, which could be a sufficient method to mitigate these vulnerabilities. One should notice that the browser exam key attack, presented in section 4.2.1, could be used to modify the source code and/or the configuration file to access communication, without being detected.

4.3 Threat Descriptions and Penetration test cases

This section will summarize the vulnerabilities that were discovered by using HARM in the previous section. I will discuss these vulnerabilities and present some of the related threats. These threats will be presented as threat descriptions, which will help further with the development of penetration test cases for this project. The tests described in this section are lab penetration tests, which focus on penetrating the security of the exam software. Table 4.8 is an informal representation of some of the vulnerabilities discussed in the last section to give an overview, and create the basis to discuss threats and test cases to be performed. As previously mentioned, I will not directly test vulnerabilities regarding detection and surveillance,

ID	**Description**	**Mitigation**	**Threat**
V1	Key injection	None	Confirmed with key injection
V2	Suspicious typing activity	None	key injection, clipboard, remote communication
V3	Can access clipboard content	Empty clipboard	Code modification, save crib notes on clipboard
V4	Run with modified source code	Browser Exam Key	Code modification
V5	Run with modified configuration	Browser Exam Key	Code modification
V6	Escape SEB	Locked down environment	VM, Remote desktop, switch user, access task manager
V7	Internet access	Internet restrictions	Remote resource and communication, audio/video communication, shared desktop,
V8	Run in VM	VM detection	Modify source code, modify configuration
V9	Run background processes	Process monitoring?	Confirmed by audio/video communication attack, get assistance by being remotely connected
V10	Modify SEB Service	None	Modify service source code, run with all registry settings disabled, escape SEB

Table 4.8: An informal summary of the vulnerabilities discovered with HARM and code reviewing

even though it has been discussed to a certain degree in the previous sections. The risk of being discovered by for example another examinee and/or an invigilator should be considered and discussed. I will attempt to research whether some of the malicious activities have the possibility of being detected, through network surveillance and existing logging functions, by going through this information during a test case. Vulnerability **V7**(Internet access) in table 4.8 will therefore only be discussed, based on this information, but it will not directly be assessed. This is mainly due to the fact that there are no detailed information available about how this vulnerability is mitigated in digital exams at NTNU, and I have no basis to be able to set up an equivalent test environment.

Some of the vulnerabilities have already been successfully exploited in the prestudy [46], and there is no indication that these will be better mitigated with Moodle. The confirmed vulnerabilities are **V1**, **V6** and **V9** from table 4.8. **V1** were proven by using a USB rubber ducky key injector, and can not be mitigated in an exam environment by network restrictions.**V6** was confirmed in a lab setting by connecting with a remote computer, but this attack's success depends on which network restrictions are used at the exam venue. Ideally, one could get the same result without any network access, by using a VM to run the exam environment on. **V9** has also been successfully penetrated with two different communicating tools in the background of SEB, and we know that it is possible to use partial audio and video communication while an exam is running. This attack also depends on unrestricted access to the Internet, and it provides limited ways of communicating, where the accomplice can see and hear the examinee, but the examinee can only hear the accomplice, and talking should be difficult in a traditional exam setting, without causing suspicion. **V9** also proved vulnerable to remote desktop connection, as the accomplice was able maintain a remote connection with the examinee during an exam, and could participate in answering the exam by controlling the examinee's computer. This attack is also dependent on unrestricted Internet access.

Something that becomes more clear with the HARM approach and this table of vulnerabilities, is that some mitigation mechanisms and threat approaches are repeated for different vulnerabilities. These are source code modification, and the use of Internet. Most of the vulnerabilities that has not yet been successfully exploited, has the possibility of being compromised by modification of the source code. These threats are only relevant because of the fact that SEB is an OSS that is installed on the student's BYOD computer, and anyone could be able to make and/or distribute a modified version of the software, legally. If **V4** and **V5** in table 4.8 are compromised, it opens for the possibility for the attacker to disable mitigation mechanisms that already have been proven sufficient. Examples of such vulnerabilities can be found as **V3**, **V6**, **V8** and **V9**. There is also left to prove that the browser exam key protects against small code modifications, such as disabling the VM detection mechanism or to disable the method that empties the clipboard. But if it is possible to manipulate the browser exam key, one could in theory disable and enable mechanisms freely, both in the source code, and in the

Description	**Inject crib notes using clipboard**
Threat target	Access content on clipboard
Risk	
Attack techniques	Disable the empty clipboard method in the source code
Tool usage	Visual Studio 2015(for code editing), existing operating system function(clipboard)
Countermeasures	Disable right clicking, empty clipboard, browser exam key
Expected outcome	Will not be able to access the exam, due to wrong browser exam key

Table 4.9: Threat description

Description	**Run SEB on VM to access local resources**
Threat target	Access local resources
Risk	
Attack techniques	Disable the VM detection method in the source code
Tool usage	Visual Studio 2015, Virtual Box 5.0.10, Oracle
Countermeasures	VM detection, browser exam key
Expected outcome	Will not be able to access the exam, due to wrong browser exam key

Table 4.10: Threat description

configuration settings. The browser exam key is computed by using the settings file, the execution file for the SEB client application and some XULsettings that I have not looked into in details. The SEB Windows Service has it's own execution file, that is not being used to compute the browser exam key. If this was modified, without changing it's behavior towards the main application, it would most likely not be detected by the browser exam key. This is presented as vulnerability **V10**

Threat risk modeling is an important part of developing and maintaining a secure application [41]. The objective is to identify vulnerabilities and threats, and to mitigate them to develop a more secure application. The OWASP guide recommends using a structured method for threat risk modeling, to be able to maintain a secure application, and refer to the Microsoft threat modeling process as a suggested approach. This method is mainly designed for assessing web applications from being exploited by malicious users, and SEB is not a web application. It is still possible to apply an approach of the Microsoft threat modeling to assess the threats of SEB. The main steps of this methodology are to (1) identify the assets, (2) create an architectural overview, (3) decompose application, (4) identify threats, (5) document threats and (6) Rate the threats [31]. The assets have already been identified as the security requirements in section 4.1. Chapter 3 presents an architectural overview and decomposition of SEB. The application threats and countermeasures are discussed in sections 4.2.1 and 4.2.2, with the help from HARM.

The threats that are derived from the HARM analysis is to try and disable the browser exam key, and/or disable emptying of the clipboard and prevention of running on a virtual machine. To be able to recommend sufficient Internet settings during an examination, it will also be beneficial to look into the threat of

Description	**Communicate by shared desktop, audio and video**
Threat target	Accomplice can talk to examinee and view exam assignment
Risk	
Attack techniques	Enable sharing of desktop by modifying source code
Tool usage	Visual Studio 2015, Skype V.7.23
Countermeasures	New desktop kiosk mode
Expected outcome	Will not be able to access the exam, due to wrong browser exam key

Table 4.11: Threat description

Description	**Modify SEB Windows Service**
Threat target	Run SEB without modified registry settings
Risk	
Attack techniques	Avoid registry editing by modifying the source code
Tool usage	Visual Studio 2015
Countermeasures	None
Expected outcome	Should be able to run undetected

Table 4.12: Threat description

Description	**Generate the expected browser exam key for the url header**
Threat target	Enables the use of cheating settings and software modifications during an exam
Risk	
Attack techniques	Modify the browser exam key generation mechanisms, to generate the key as if the code and configuration are correct
Tool usage	Visual studio 2015
Countermeasures	browser exam key
Expected outcome	If correct modification; success, else; no access to the exam

Table 4.13: Threat description

	Action	**Success criterion**
1	Modify the SEB source code	Browser Exam Key gets generated by using original, unmodified versions of SEB and the configuration file, runs error free
2	Start the Moodle server and open SEB with a manipulated configuration file of you choice	Access to exam granted

Table 4.14: Penetration test case for disabling the Exam Browser Key mechanism for code and setting control.

	Action	**Success criterion**
1	Disable the empty clipboard method in the source code and save crib notes on clipboard	Runs modified code error free
2	Start the Moodle server and open SEB with correct configuration settings	Access to exam granted
3	Paste notes from clipboard into exam	Confirms that clipboard is no longer deleted

Table 4.15: Penetration test case for using clipboard by disabling the empty clipboard method in the source code.

sharing desktop with audio and video communication. The threat descriptions are presented in tables 4.9, 4.10, 4.11, 4.12 and 4.13. I will attempt to exploit these threats by applying penetration tests, which are described in tables 4.14, 4.15, 4.16, 4.17 and 4.18

4.4 Risk Assessment

This section will contain risk assessment, which is the final step of the Microsoft threat modeling method [31], introduced in section 4.3. In software security, the generic way of assessing risk, is to consider the two factors, likelihood and impact. Likelihood is the probability of exploitation, which considers how easy the threat is to exploit and discover. The impact is the damage, caused by an attack, to the application. I will use Microsoft's DREAD threat-risk ranking model, to assess the risks in this thesis[31, 41]. This method uses 5 different factors, and rank these factors on a scale from 1(lowest) to 3(highest). The rankings of the different factors are added to calculate the overall risk of the threat. If the score is between 5 and

	Action	**Success criterion**
1	Disable the VM detector in the source code	Modified source code runs error free
2	Start Moodle server and run SEB on virtual machine with correct configuration settings	SEB is not blocked and access to exam is granted

Table 4.16: Penetration test case for running exam environment on a VM by disabling the VM detector in the source code

	Action	**Success criterion**
1	Modify source code to enable shared desktop	Modified program runs error free
2	Establish skype connection between accomplice' and examinee's PCs, examinee shares his screen with accomplice	Connection established
3	Start Moodle server and run SEB with correct configuration settings	Access to exam is granted while still being connected through skype in the background
4	Communicate	Examinee can hear accomplice, accomplice can talk to examinee, accomplice can see the exam environment, on the examinees computer, through the shared desktop feature. Accomplice can see examinee

Table 4.17: Penetration test case for the sharing desktop, while running SEB, by enabling it in the source code

	Action	**Success criterion**
1	Modify the SEB Service source code	Run Service without modifying registry enabled
2	Start the Moodle server and open SEB	Access exam, can use for example task manager and switch user

Table 4.18: Penetration test case for disabling registry editing

7, the risk is low, between 8 and 11 gives a medium risk, and a threat of high risk should have a score between 12 and 15.

The factors of DREAD are Damage, Reproducibility, Exploitability, Affected users, and Discoverability. The damage potential depends on how bad the damage will be, similar to the generic factor of impact, if the vulnerability gets exploited. The reproducibility of a threat is how easy it is to reproduce, dependent on factors that could be outside of the attacker's control, such as a race condition. Whether the attacker needs to have extensive technical skills and knowledge or not is measured with the exploitability factor. The affected user factor measure the amount of neutral users, in this case examinees, which may be affected by an exploitation. The discoverability factor can easily be misinterpreted in the context of this thesis. I have used the term discoverability to describe how a cheating examinee can be discovered by, for example network monitoring and invigilators. In DREAD context, discoverability is concerned with how easy it is for the attacker/cheater to discover the vulnerability that they can exploit.

Table 4.19 contains the DREAD risk assessment model of the threats I have described. They all where rated as of medium or high risks. The overall damage factor were high, I chose to interpret the damage factor in this context as the increased performance one can gain by a successful exploit of the vulnerability. All the threats, except for **T1**(Clipboard) and **T3**(Skype). The attacks should be

Threat	**D**	**R**	**E**	**A**	**D**	**Total**	**Rating**
T1. Inject crib notes using clipboard	2	2	2	2	3	11	Medium
T2. Run SEB on VM to access local resources	3	2	2	2	3	12	High
T3. Communicate by shared desktop, audio and video	2	2	2	2	3	11	Medium
T4. Generate the expected browser exam key for the url header	3	2	1	2	2	10	Medium
T5. Modify SEB Windows Service to not modify registry	3	2	2	2	2	1	Medium

Table 4.19: DREAD model [31]

easy to replicate, as there are now race conditions or network traffic dependencies. There is the issue of Internet access, which may or may not affect the reproducibility factor for different set ups of an exam environment. There are also some degree of preparations involved with all of the attacks. The skill levels necessary for these attacks is high. One needs to know how to read, interpret and modify the source code, however. It might be possible to get access to or buy already modified software versions of the code, but I assume that there are limited options available if one is not able to to the modifications independently. An attack aimed against the security features of SEB as a single, locally installed software on a personal computer will have little or no impact on other users at the same location with their own SEB version installed on their own separate computers. On the other hand, if a student gain better results on an exam due to cheating, it could affect the grading statistics, so that other students receive a lower grade. In the bigger picture, cheating students, with better grades than non cheating students, could get better jobs, they could get into a better educational institutions. The probability of discovering vulnerabilities related to the fact that SEB is open source is small. The most likely of the threats to be discovered (DREAD definition of discovery) is probably **T3**, but because it is known that SEB prevents VM and clipboard, it is also possible for an attacker to see this as an opportunity.

Chapter 5

Penetration Testing

This chapter describes the test setup and test results of the penetration test cases that were described in chapter 4. The first section describes how the test environment were set up and which tools and software that was used. Section 2 presents the configuration settings that was used for all of the tests and section 3 describes the test results. The final section presents the final results together, discuss their consequences and generalizes the findings.

5.1 Test setup

To be able to run through the tests as painless as possible, we prepared by setting up all of the underlying tools beforehand. This section describes which tools we used, which versions and how they were set up. Safe Exam Browser is the software that is being tested, and the software is described in detail in section 2.4 and in chapter 3. The testing will be done on version 2.1 of SEB for Windows, because this software is open source, any examinee can access the code and information on the Internet and use this knowledge to potentialy cheat. This is the reason why some of the tests are based in attempting to modify the source code and/or the configuration settings used during an exam. The test computer is an approximately 5 year old Acer Aspire with 64-bit Windows 10 OS installed. It has previously been used by the author to do school work and should be a good example of a computer that a student could bring to conduct a BYOD exam on. I work under the assumption that, everything I am able to accomplish by modifying and testing, could also be done by a cheating examinee. On the other hand, the attacks presented in this thesis is unlikely to achieve the exact same outcome during an exam, as I do not test network restrictions, and the risk of being detected.

Safe Exam Browser provides a tool called SEB ConfigTool, which is described in section 3.4. This configuration tool will be used for creating a default, exam configuration file for all the tests, and for creating modified, 'malicious' settings. The SEB developers recommend to encrypt configuration files that are to be distributed to examinees, or they can be directly manipulated in a text editor. Description of the settings used for the penetration tests can be found in section 5.2

NTNU uses IA as the LMS to create, distribute, deliver and authenticate the exams, and more exam related services to assist the administration. I made attempts to get access to IA to use for the penetration tests. I had some e-mail correspondence with Inspera as well as a phone call, regarding the possibility of cooperation. They seemed optimistic and interested in the possibility, but due to the long response time I experienced, I had to continue my work, and find a tool to replace IA. For the tests, I ended up using the LMS called Moodle(described in section 2.5), which is open source and the code can be downloaded and installed as preferred. For this project, Moodle was installed as a local web server on the test machine, using the lightweight, cross platform web server solution stack package XXAMP [54]. Moodle was configured using default settings. The SEB Moodle plugin was installed on the local server [22], to be able to make exams in Moodle that can only be accessed when using unmodified SEB and the predefined configuration settings. I made test users and test exams in Moodle with different types of questions to run the penetration tests with

I installed Skype v. 7.23 to use for audio, video and sharing of desktop, but there was a port number conflict with XXAMP, that was also using port 80, and I could not use the two applications at the same time. I had to change the port numbers used by XXAMP, so as to be able to run my Moodle server at the same time as I ran Skype. I installed Oracle's free VM tool called virtualBox 5.0.10, where I installed a machine with Microsoft Windows 8.1 OS. I also had to download a version of the SEB source code [14], in addition to the SEB software I already had installed [5]. The descriptions and recommendations I found related to of SEB source code, recommended to use the Visual Studio 2015 IDE [2] which is specialized for C# and .NET software development, which is the coding languages used to create SEB.

An empty clipboard method are called from the method InitSEBDesktop, in the class called SebWindowsClientMain, and that method is located in SEBClipboard, which is located in ProcessUtils. It imports functions, by using Dllimport on the user32.dll system file, for emptying, opening and closing the clipboard, which are the Windows native clipboard functions [32]. The OpenClipboard() function reserves the clipboard content from other applications and enables editing. The EmptyClipboard() function empties the clipboard and all data references contained in it. Finally, SEB evokes the CloseClipboard() function, which close and release the clipboard for regular use. It should be sufficient to only remove the call to the EmptyClipboard() function for this test.

The VM detection method is also located in the class called SebWindowsClientMain, and the method are called CheckVMService(). This method checks if VM is allowed by the configuration file, and whether the program is being run inside a virtual machine. It finds this information using a ManagementObjectSearcher, which is used to query for management information [33]. It finds the manufacturer and model of the OS and searches for known distributors and services, such as VMWare, parallels software, xen and virtualbox. Not only does this make it easy to discover a potential VM tool, that are not present in the system query, just by looking at the code. Any VM tool not checked in the source code will probably not be detected by SEB. For this test, where I use virtualbox, I can remove the fracture of code that asks for that and it may not be noticed with the browser exam key checker.

The methods in the source code that disable shared desktop is harder to locate. I will have to test several mechanisms that could maybe be responsible for the shared desktop blocking. In the prestudy, I discovered that the shared screen that the accomplice could see was either black, or shows the same screen as from before SEB was initiated. This result gives me reason to assume that there is some code that assures this. After the detailed code review I conducted (described in chater 3), I have an idea of what mechanisms could be doing it. There are a file called Interop.cs in the WlanUtils folders that flags and alert settings that could be resulting in the blocking. There is also several system functions that are imported and called, which should be researched further. They are mainly located in the SebWindowsClientMain. I have spent a lot of time researching the registry service and are reasonably sure that this is not what block the desktop sharing.

For network surveillance, I have installed Fiddler for .NET4 version 4.6.23. I also want to look at the traffic between the SEB browser and the localhost, and therefor, I have installed a socket sniffer called CapRaw v. 0.1.5.0 from NETRESEC, which creates capture files that I can view in Fiddler.

5.2 Configuration Settings

For the penetration tests I will use the test exam that I made in Moodle, and I need to have a configuration file which represents the configuration settings for a real-life BYOD exam. I will create separate configuration files for running SEB in VM and running on host machine. This is only to change the start URL which will be different when accessing a local host server from a VM. There will be one encrypted and one unencrypted version of the settings file, but I will mainly use the encrypted versions, encryption will be done using a password. The start URL of the settings for use without a VM is http://127.0.0.1:82/mod/quiz/view.php?id=2, to get access to the test quiz on the local host XXAMP server. The port number has been changed from 80 (default) to 82, to avoid conflict with skype. The settings will have an administrator password, to avoid modification by examinees, and it will be possible for the examinee to exit the browser, by using a specified password. The exit key sequence is disabled. The IP address used to access the Moodle server from the virtual machine is 10.0.2.2, and has to be changed for the startup URL. It is also important to use the original SEB, execution file when the browser exam key is generated in the config tool. If the browser exam key is generated with the modified version of SEB it defeats the purpose of these tests and will create a corrupt and unrepresentative outcome. When administrators create the configuration file and browser exam key, they use the original SEB version, and this is what I am trying to imitate in this setup.

The browser will be in full screen mode, with all available SEB taskbar options visible. The browser security will disable plug ins and popups, and delete user information. Down/uploading is disabled and there are no restart and quit URL's defined. The option to send the browser exam key in the header is checked. In the permitted processes table, I added two network surveillance application, for documenting of the network activity. They will not directly affect the security in any way. These processes are CapRaw and Fiddler, presented earlier. There will be no processes in the prohibited list, and I will use default proxy, network and certificate settings. SEB will only run if the SEB Windows Service is detected and running. VM is not allowed. Logging is enabled and saved at a default location on the computer that runs the exam environment. The registry settings are all checked, and should ensure that all the registry settings are changed for the duration of the exam. Finally, there are no special keys enabled and no print-screen possibilities. The chosen kiosk mode are to create a new desktop entirely.

5.3 Test Results

5.3.1 Inject crib notes using clipboard

This test is the simplest, and hardest to detect with both digital monitoring and invigilating. The objective of this test is to get a grasp on the code details and modifications that will be detected by the browser exam key functionality. This digital cheating behavior can be compared with bringing prohibited crib notes to a pen and paper based exam. The notes that are used to cheat, have to be created and prepared prior to the exam and, in this case, saved on the clipboard.

First, I ran a test for comparison, with the original code and configuration settings as described in 5.2. I stored test notes on the clipboard prior to initializing SEB. The exam environment ran as expected and the notes, that was stored on the clipboard, was gone. It was, however possible to use the clipboard within the environment. This is a useful tool for exams, particularly with bigger writing assignments. The browser exam key check was successful, and access to the exam was granted.

I also tested to run the exam with the wrong configuration file, both for the modified and unmodified SEB version. The tests ran as expected, and I was able to run the program, but access to the exam was denied by the LMS. The exam's entry web page, suggested to use the correct SEB version to get access to the exam, and the access button was changed to a ”go back” button.

For the main test, I had to do code editing. There is a separate class called SEBClipboard, which is located in the processUtils package of SEB Client. For this attack I commented out the functionality that empties the clipboard, which is called from the main class of SEB Client. I also made sure that the entry ”Clipboard cleaned” still appeared in the client log. I saved some test notes on the clipboard. Then I replaced the original with the modified execution file, and ran SEB as an examinee normally would. The program ran as expected, I could paste my test notes into the system. The most surprising result was that I could still access the exam. Which implies that the browser exam key functionality did not detect the minor code modifications that had been done. For this test, there were no malicious network activities and therefore network surveillance was unnecessary. The logs did not detect anything suspicious, as was expected.

5.3.2 Run SEB on VM to access local resources

This test also includes code modification. First, I attempted to run anunmodified SEB version on the VM. First, I used a configuration file, with settings to allow virtual machine. For this test I was able to run SEB, but I was not granted access to the exam. This was because the browser exam key functionality detected the use of the wrong configuration settings. Then I attempted to run on the VM, with the correct configuration file. SEB refused to run, it gave a warning pop-up box and exited at once.

Then I made a version of the software with the VM checker disabled. I attempted to run the modified code on the VM with a modified configuration file, and was denied access to the exam, just as with the same test and the unmodified software version. Finally, I ran the modified code with the correct configurations on the VM. This was a success. I was allowed to run the software, regardless of the VM detector. I was also granted access to the exam and could minimize the Virtual Machine to access resources.

This attack allows the cheating examinee to access both local resources, Internet resources and communication. It is an attack that can be done either with or without free Internet access. The interesting question is whether the allowed Internet activity looks different, than when the examinee is not using VM. The answer is no. The VM is undetectable through network surveillance, and the network traffic will look like it came from the host machine. On the other hand, if the examinee attempts to access the Internet from the host machine, it can be detected. This attack can still grant access to locally stored resources.

The log files have the potential to disclose suspicious activity though. The log files will be stored on the virtual machine, and not on the host computer. The log files on the host machine, if they exist, will not contain information about SEB that is being run inside a VM. The log files stored on the virtual machine, may disclose that a VM has been used. In the 'SEBClient' log, the manufacturer is shown, and in this case that is VirtualBox.

5.3.3 Communicate by shared desktop, audio and video

This attack was attempted in the prestudy (see section 2.7), and was proven unsuccessful. It was also unclear in the prestudy, why this attack failed, but remote desktop worked. By intensive code reviewing and research I suspect that this is caused by the kiosk mode. The settings can be configured to choose what type of kiosk mode the exam environment will to use. The choices are; to create a new desktop, to kill explorer shell or to use nothing. The last option is for debugging purposes only. By default, SEB creates a new desktop. This could explain what happened in the prestudy. When I attempted to share the examinee's desktop, using Skype, the accomplice could still see the underlying windows. When you use the kiosk mode setting called kill explorer shell, SEB closes or minimizes all other applications. The Microsoft start menu, taskbar and file explorer windows are minimized, closed or blocked. The examinee ca neither access minimized applications or start new ones, because the taskbar and start menu is blocked.

I attempted to run shared desktop, using Skype, while running the original SEB software with the kill explorer shell kiosk mode chosen in the settings. The accomplice was then able to see the examinee's screen including the SEB browser window. The examinee could not access the exam in this kiosk mode, because of the changed settings being detected by the browser exam key functionality.

These two kiosk mode options, in the SEB settings, creates two major different executions of SEB, and thereby also requires code modifications in several areas. The simplest ways to fix this, would be to override the values of the current settings variables. There are two relevant setting variables that hold these settings, but the browser exam key is generated by using all setting variables, including these. I first attempted to do this, and ran the correct configuration with these variables being overwritten in the source code. I was not granted access to the exam by Moodle. This code manipulation will appear as if the wrong configuration settings were used.

I then had to locate and modify the code, whenever these variables were checked, without actually changing them. This resulted in modification of at least 4 different classes in the SEB Client source code. I ran the modified code, with the intended configuration settings, and SEB opened by killing explorer shell, even if the settings said to create a new desktop. The browser exam key detected these modifications and denied access to the exam. This attack turned out to be unsuccessful. However, if the exam environment kiosk mode is set to kill explorer shell by an exam administrator, remote desktop communication will be possible.

This attack requires network usage, and the Skype activities with detected by

Fiddler, and will hopefully be blocked in a real-life exam environment.

5.3.4 Modify the SEB Windows Service

This test is based on the new knowledge that was acquired by code reviewing and architectural inside into the application, which is presented in chapter 3. The knowledge that created the basis for this test is regarding the browser exam key. This key is generated by the current settings that are used, the main client application execution file and XULRunner configurations. The SEB Windows Service however, has it's own execution file and code basis, which is not assessed by the browser exam key. This service can modify the Windows registry, which is concerned with options such as logging off, switching user, and accessing the task manager. If one can change the features of the service, to not modify the registry, it should not be detected by the browser exam key, and affect whether access to the exam is granted.

I started by researching how the service works and attempted to run SEB with and without the service running. When I used the test settings configuration file described in section 5.2, without the service running, I got a warning about the missing service and SEB did not even start. I attempted to to this with the wrong configuration file, and SEB started normally but Moodle denied access to the exam due to the wrong browser exam key. If I ran the service and used the correct settings, everything worked just as expected, SEB ran smoothly and access to the exam was granted.

For the major test, I modified the source code of the SEB Windows service to not edit any registry settings, but I still wanted it to act as expected toward the main SEB application. I switched the original service with the modified version, and ran SEB. SEB started without any problems and access to the exam was granted. I was now able to for example access the task manager, log off, close the computer and switch users while answering an exam in the exam environment. The most advantageous of the new possibilities appears to be the switch user option. An examinee can switch the active user, so that SEB runs on one user, and a different user can be used to access resources and even the Internet. On the other side, this cheating behavior will probably be easy to detect for invigilators and/or other examinees.

This attack was detectable in the service log. The service log records every registry editing, as shown in the appendix, section C.4. While applying this attack, the only entries added to the log, is that of disabling and enabling the windows updater, which is not a registry editing feature of the SEB Service, this is also shown in the appendix (C.4). It should be possible to modify the service so that the log will show that the registry settings has been changed, even they have not. This is not tested further as it is not legal to use log files to detect cheating behaviors.

This attack is independent of Internet access, restrictions and/or monitoring. I will therefor not assess the network usage of this attack.

5.3.5 Generate the expected browser exam key for the url header

This attack is the most ambitious of the attacks in this thesis and, it turns out, requires major code editing and knowledge of the application, libraries that are used, contents of SEB related files and folder, and the coding convention and architecture of the source code. This attack did not require Internet access, which is why I did not use network monitoring tools.

The motivation for this attack is that all of the source code should be available to any examinee with Internet access, and he can attempt any code modification of his choice, only limited by skills, knowledge and creativity. The browser exam key is the main functionality to ensure the solution's integrity, and to compromise it, makes all sorts of cheating activities possible.

The browser exam key is generated by creating a hash from the currentSettings dictionary saved by the SEBSettings class, the Safe Exam Browser execution file, and a string called XULRunner hash. The XULRunner hash is generated by creating a string from all files that contain the string "SafeExamBrowser". The idea of this attack is to create a fake current settings dictionary, from the intended settings file, while actually using freely chosen settings. Because these code modifications will most likely change the execution file, it should also be necessary to use the original execution file for SEB, even though this is not the version of SEB that is being run.

I copied the SEBSettings class, and made a completely identical class called SEBFakeSettings. I used this class, and the methods within it to load and store the intended settings from a file and into the variables of the fake class. I then saved the original SEB execution file with a different name, and changed the pathname in the ComputeBrowserExamKey() method accordingly. Then I ran the modified version with a configuration file that should not be accepted by the browser exam key check.

SEB ran normally, but I was unable to access the exam. I attempted several different code modification approaches and fixes, but was unable to make it work. There could be several reasons why this attack failed. The fake SEBSettings class use other classes of SEB to modify the settings, and they modify the settings directy, using "SEBSettings.variablename". I attempted to follow all these dependencies and write fixes so this would work for the fake settings, but some of the dependencies may have been overlooked. The name of the original execution file, was changed, and could maybe affect how it is hashed by the hash function. The changes of the code, and file organization could have changed the XULRunner hash. The SEB Browser exam key is salted by a variable which is stored in the settings. I attempted to modify the fake settings class to store the correct salt for the original settings, as well as to modify the browser exam key to use the "fake" salt, but this may not have worked.

Attack	**Result**	**Conclusion**
T1:Clipboard	The modified SEB worked, access to exam was granted	**Success**
T2:VM	The modified SEB on VM worked, access to exam was granted	**Success**
T3:Shared Desktop	The modified SEB worked, access to exam denied	**Failure**
T4:Browser Exam Key	The modified SEB worked, access to exam denied	**Failure**
T5:Service	The modified service worked, original SEB worked, access to exam granted	**Success**

Table 5.1: Summary of the results of the penetration tests.

The number of different setting values stored by SEBSettings is extensive, and it is difficult to debug them and those of the fake settings class. The time limitations of this project affected the amount of time that could be directed towards this particular attack.

Even though this attack was unsuccessful, it is not evidence that the browser exam key function is bullet proof. There are several aspects of this attack that I would have liked to investigate further. The results of this attack does confirm that to perform this attack, one needs to be an expert in the related libraries, configuration and SEB software. To perform this attack without extensive prior knowledge, would probably be more demanding than for an examinee to actually prepare for the exam, without cheating.

There are aspects of the preparation for this attack that could have been better, to maybe make it succeed. I could have done more extensive research into the XULRunner hash and the hashing libraries that are used. I also had limited experience with C#, .NET and Visual Studios, prior to the thesis. If I had had more knowledge beforehand, I probably would have had fewer struggles with the code modifications.

5.4 Result Summary

Table 5.1 summarizes the results from the different tests. The attacks that include a small number of code modifications were generally successful, when checked against the browser exam key. **T1** and **T2** are threats that were able to exploit vulnerabilities. **T5** required more code modification than **T1** and **T2**, but there are not mitigation mechanisms discovered to prevent the use of a modified version. The remaining attacks required more code modification, and they were prevented by existing mitigation mechanisms.

The browser exam key certainly provide important security to Safe Exam Browser. Unfortunately, it does not work to detect the smallest changes. The browser exam key limits the possibilities for the cheater to take advantage of the fact that the software is freely available, and the key is able to prevent significant code modifications. The vulnerability of accessing crib notes in a digital exam, can be compared with bringing physical paper based crib notes to a pen and paper exam. This is one of the more used cheating behaviors and is relatively simple to do, but it also limits the amount and content of sources that the student can access. **T2** and **T5** could have a bigger impact on the amount of resources that become accessible. Even if there is restriction on the Internet access, the cheating examinee could have prepared to cheat by downloading all sources that could prove useful.

It is evident that the browser exam key is efficient to protect against using malicious setting files, but the way in which it checks the execution file is not sufficient to detect malicious software modifications. It proved to be challenging to get usable network surveillance data during the testing. The major issue with this was because I ran the LMS on a local host which made the network monitoring data regarding Moodle difficult to relate to a real life situation. The general conclusion that can be made from the Internet traffic, is that network restriction can mitigate efficiently against communication, and limit the consequences of attacks that create the possibility to give the examinee access to a Internet browser outside of the kiosk environment.

Chapter 6

Discussion

This chapter will discuss the processes and the results that have been presented in this report. The first section is a discussion of the test results, and how it contributes to the discussion about cheating behaviors in digital exams. The second section will discuss mitigation strategies to limit vulnerabilities of the exam environment, both technical and non-technical approaches. Section 3 will compare the solution that are used at NTNU, which I have elaborated and tested in this thesis, with other digital and non-digital exam solutions that have been presented. Finally, I will discuss the benefits and weaknesses I have experienced when using HARM to plan penetration testing.

6.1 Test Results

The results from the penetration tests have been summarized in section 5.4, and there are to major vulnerabilities that have been discovered. One of these vulnerabilities is the SEB Windows service, which has no mechanism to ensure that it has not been compromised. As long as the service can be detected and is running, there should be no indication that the service is modified. This vulnerability has the potential to give a cheating examinee access to all locally stored resources, and even though the attack could be easily spotted by invigilators, it should be discussed and potentially mitigated better.

The other vulnerability that was discovered is regarding the browser exam key. The browser exam key does prevent the use of the wrong configuration file, but it does not sufficiently protect against code manipulation of the main application. The code is open source, and it is necessary to assume that a student, or outsiders who want to sell malicious exam software, can manipulate and distribute code that can enable cheating behaviors. That the code is open source enables institutions to make specified adapted versions to better suit their exam setup, and it allows outside developers to contribute with useful functionality. It is free to use, which encourages institutions to choose it for their digital exams, while still maybe making donations towards the development of SEB. The downside is that, because it has to be installed individually on each device, which is owned by student, there is a risk of the installations to be corrupt. The more SEB is used and known, the more people can possibly make contributions and donation to make it an even better solution. Unfortunately, the more regular users, the more malicious users, and it is more likely that modified versions can be developed, distributed and even sold for a profit.

To be able to pull off any of the attacks that have been presented in this thesis, the attacker needs to have at least some basic programming skills, which may limit the extend of the threats. As previously mentioned, an examinee can potentially purchase modified versions of SEB. All of the attacks require some technical knowledge, but if the program gets more used, more malicious, modified versions can be distributed online. The possibilities for attacks in digital exams should be continuously mitigated, and institutions that conduct digital exams needs to be aware of the existing and new threats and vulnerabilities.

6.2 Mitigation Strategies

First, I will discuss how to mitigate the attacks that were successful in the prestudy (see 2.1), and I will discuss mitigation strategies for the vulnerabilities that were exploited during this thesis. Finally I will discuss general mitigation strategies that should always be used to create a sufficiently secured digital exam environment.

In the prestudy [46] the simplest attack that was successful was the Key Injector. This attack was inspired by Dawson [13], and he discuss the possibilities to mitigate this attack. To disable USB typing devices is not recommended, as this can disable certain keyboards, which is not ideal for an exam in which the examinee has to type longer answers. The exam environment could monitor the typing speed, and give a notification if the typing speed is suspiciously fast. The key injector script can, unfortunately, be written with a typing delay. Which makes it harder to detect by any technical monitoring, even screen monitoring would not be able to detect whether the typing is done by the examinee or an injector. This attack can be detected by invigilators, if the key injector have not been sufficiently hidden or disguised as a regular keyboard. This attack is similar to the clipboard attack that has been performed in this thesis. The clipboard attack can not adapt the speed in which the text is pasted into the environment, which makes it vulnerable to screen monitoring. Whether device monitoring and logging can be used to mitigate cheating raises ethical and legal questions. Monitoring of student devices, will probably only discover a fraction of cheating behaviors that could not have been caught by invigilators. The legal issues, and costs, are valid arguments to prevent device monitoring and logging to be used to mitigate cheating.

The attack, using Skype for audio and video communication have been extended in this thesis to include shared desktop. The shared desktop was unsuccessful, and has proven difficult to accomplish, but communicating with audio and video need to be better mitigated. A locked-down digital exam environment should disable the use of microphone, sound and camera, which is not done sufficiently with SEB. This can be done by, for example registry editing, which have been discussed as part of the SEB Windows service. Another approach could be to better monitor and close background programs. SEB documentations claim that SEB closes existing processes if they are in the prohibited list, but this list would have to be extensive to include all possible applications that can possibly be used to cheat. This is a vulnerability in SEB for BYOD exams that was admitted by Schneider during the Inspera seminar [44]. If possible, it could maybe be beneficial to close all background processes that are detected.

Attacks that attempt to use a remote desktop connection are dependent on unrestricted Internet access, which makes Internet restrictions the best approach to mitigate for these attacks. Restrictions of the Internet access, by for example only allowing examinees to access the LMS, will mitigate all kinds of communication attacks that does not use an external device. The use of an external device, should be mitigated by using well trained invigilators to detect and remove such devices. To restrict the Internet access can also contribute the consequences of attacks that are able to escape the exam environment, by using for example a virtual machine or switching users. The cheaters may still access local resources, but they will not be able to communicate or access Internet based resources. If Internet access is restricted, it may not be necessary to disable camera, microphone and audio in the exam environment, unless external devices with Internet access are used.

The attacks that have been conducted in this thesis already have existing mitigation strategies, and the attacks that succeeded revealed which of these mitigation mechanisms that are insufficient. I would like to suggest that the browser key gets improved. It works perfectly when checking the configuration file, but it does not work sufficiently when checking execution files. If the browser exam key finds a better approach to check the source code of the running SEB system, it would help mitigate the simplest attacks, such as clipboard and VM. The key could also be changed to include checking that the SEB Service has not been modified a well. Event hough it should be possible to develop a version of SEB that avoids detection of the wrong browser exam key, this is a lot more difficult to accomplish, than for example the clipboard attack.

6.3 Exam Solution

At NTNU, and several Norwegian institutions, they run SEB directly on the students laptops to create a locked down environment. Other solutions that have been used include running SEB in an OS on a bootable device on a student laptop, running SEB on a virtual client desktop either on a student laptop or on the institutions own equipment. To use a different independent OS, either with a bootable device or on institution owned equipment, ensures that local resources are completely inaccessible from the exam OS. On the other hand, to run SEB with VDI on a student computer, uses the original OS to run the virtual client and the resources will, in theory, be available, similar to just running SEB directly on the laptop. One BYOD solution is to use SEB to run the virtual client as a third party application, and then run SEB again inside the VDI to access the exam. This solution could present the same vulnerabilities as if SEB was run directly on the computer, and the investment costs of acquiring necessary servers may not be worth it. The more secure solutions, using institution equipment or a bootable device, is likely to cost more. If the cheaper solutions are presented as secure enough to use for digital exams, they will probably be more widely used. The more users, the more resources will become available to improve the solution.

Paper based exams are also vulnerable to cheating behaviors, and it is not that difficult to be able to use crib notes for a written exam at NTNU. I personally know people who have used prohibited crib notes on a written exam, and I know other people who would never do this even if the possibility was there. For digital exams, with no paper on the examinees desk, it becomes a bigger challenge to access and use physical crib notes to cheat. To bring crib notes on paper will be detected by invigilators, and to be able to access crib notes on the computer, they would either have to bring a key injector or modified software. This requires more work than to bring a paper sheet with notes, and could mitigate the occurrence of crib notes at an exam.

The other, more complex digital attacks, are harder to accomplish and an examinee would have to be determined to cheat in advance to be able to pull any of them off. Which is also the case for more complex cheating behaviors in paper-based exams. If an examinee is determined to cheat, he will be able to, regardless of the type of exam. According to Vegendla et al. [49], a digital exam does not need to eliminate cheating, but it needs to be as reliant as paper based exams. As long as cheating require a significant effort, they should occur less often. BYOD exams with SEB and a LMS with a SEB plug-in for the browser key check, invigilators and network restrictions fulfills this requirement.

Norwegian high-schools practice open book exams on student laptops. They allow behaviors that have previously been defined as cheating, while using cheating prevention to mitigate against other behaviors. They restrict Internet access and allows students to use all the resources available on their computer. Students can even download the contents of web pages and use this, as long as they don't plagiarize. This exam environment is easier to control, and is a better representation of what real-life work situations will be like. Usually, when performing work assignments, it is important to use and find good resources in a constructive manner to produce results of good quality. This could likely be the future for assessment in higher education, and for an institution to invest in expensive equipment to provide a locked environment, may not be a profitable, long-term investment.

6.4 HARM

I used an extended HARM method to investigate system vulnerabilities and penetration test cases. HARM was used together with a architectural review and a risk assessment method called DREAD to discover vulnerabilities, plan tests and asses risks in this thesis. I used the architectural review together with security requirements and my prestudy tests and results [46] as a basis. The HARM method consisted in making diagrammatic descriptions of different attacks, and then categorize them in attack trees. I used the attacks from the prestudy to develop new attacks, that could possibly be a threat to the new test system, and described the new attacks according to HARM.

HARM helped in structuring the vulnerabilities and was helpful to see connections and similarities between different vulnerabilities, threats and attacks. The attack defense tree made it easier to discover mitigation strategies, and how to mitigate more threats at a higher level. The architectural review of the source code, was a useful tool to discover new vulnerabilities, that would not necessarily have been discovered in this application of HARM.

The models helped to create an understanding of related attack and which attacks that was aimed towards the same vulnerabilities. This made it easier to chose which attack to develop, and which would only assess the same threats. The figures and tables could also be a good way to present vulnerabilities to nontechnical stakeholders.

I adapted HARM into what I attempted to be an approach that would fit the objective of this thesis, and I would recommend to use HARM to develop a tailor-made approach to plan attacks and assess vulnerabilities. I had done previous tests that created the basis for the attacks that I modeled, which was helpful. This was the bottom -up approach, which could be used for continuous assessment of vulnerabilities and threats of a system. Most of the vulnerabilities that was presented and categorized using HARM, had already been discovered to some degree in the prestudy. The most alarming, new vulnerability that I discovered was discovered through code reviewing. The HARM models helped me develop new approaches to attack the, already known, vulnerabilities, more than it helped me discover new vulnerabilities.

Chapter 7

Conclusion and Future Work

In this chapter, I will conclude my thesis work and answer the research questions. The first section concludes the results and the answers to the research questions. Section two suggests future work that can and should be done on this topic and based on this report.

7.1 Conclusion

The thesis describes and uncovers threats and vulnerabilities of digital exams. It also presents different methods that can be used to cheat, from within a digital exam environment, or by escaping the environment during an exam. Digitization of exams in higher education is a necessary development, and helps make assessment a better representation of situations in real life, where a good education is required. It is important to work actively to prevent cheating, by using invigilators, network restrictions, a sufficiently secured digital environment and continued research. One approach to assess the technical aspect of cheating could be to model different cheating behaviors that could be possible, for example by using HARM or a similar modeling approach.

This thesis uncovers existing vulnerabilities in the exam software that is currently being used to conduct digital exams, and these vulnerabilities can, in a worst case scenario, be used by examinees to cheat during an exam. The secured exam environment, provided by SEB and Inspera for conducting digital exams at NTNU, and the tests and vulnerabilities presented in this thesis, represent only a fraction of cheating related properties. There are different types of exams where cheating also should be avoided, for example, home exams, open book exams and oral exams. Cheating is not a new phenomenon and there are other important factors to consider than just the technical aspect in which cheating is possible, and for this thesis, cheating during an exam. I have not discussed cheating behaviors that can be used before or after an exam has been conducted.

RQ1 What are the cheating vulnerabilities of SEB and how can they be mitigated?

That SEB is open source, and locally installed on the examinees' computers, present different challenges than other solutions. Examinees can make, use and/or purchase versions of the software that enables cheating behaviors that are originally prevented by SEB. There are other digital exam approaches, that also use SEB, but they can ensure that it has not been modified. However, the modifications required to cheat, are complex and time consuming to do, and a small risk of detection is always present. The consequences of being caught is also severe.

To mitigate existing threats, a combination of different approaches could be used. Risk prevention can be done, by controlling Internet access. SEB can use URL filters, and prevent access to certain web pages, but a more effective solution would be to restrict the access at the exam location. Examinees could be able to bring external Internet devices, and use them to access the Internet, which can be prevented with for example, trained invigilators.

Risk detection is another, possibly more expensive, approach. The costs of prosecuting students is significant. If examinees are aware of that, for example, digital monitoring is used and can detect cheating behaviors, they will most likely consider the risks of cheating too high. In this case, risk detection can also be a risk preventive mechanism. The laws on monitoring BYOD devices during digital exams in Norway, are non existing, but the topic should be investigated from a legal point of view, so that some sort of digital monitoring can be available for digital exam solutions in the future.

SEB should be open source, and the fact that the modifications, needed to be able to cheat, are complex. This can help prevent the risk of corrupt software usage. Developers and institutions should make themselves aware of available, corrupt software versions that could be accessed online, and prevent students from accessing and using them.

RQ2 What are the cheating vulnerabilities of SEB used with Moodle during an exam?

The browser exam key does not sufficiently check if the source code have been modified, and versions with small modifications will run undetected. The browser exam key computation should be improved to ensure that the application has not been compromised, this includes checking the SEB Windows service software. Some of the vulnerabilities that has been discovered for this solution are the same as for SEB, which is described above.

The vulnerabilities related to the use of modified code is the SEB service and other small code modifications that are not detected by the browser exam key. The browser exam key checks that the configuration settings and software have not been modified, but it does not check SEB Service for modifications. Small code modifications of the main SEB application is not detected by the browser exam key either, which have been proven through the penetration testing conducted for this thesis.

Background processes could be another vulnerability, and SEB should be able to detect and close all background processes that potentially can be used to cheat, but it currently does not. This makes it possible for an examinee to use, for example, Skype, to communicate with outside experts, during an exam.

RQ3 What are the strengths and weaknesses of HARM used for penetration test planning?

Some of the strength of HARM that was experienced during the thesis work, was the way it helped create a good overview of the vulnerabilities and to develop new approaches to attack known vulnerabilities. It also provided a good basis to better categorize vulnerabilities and plan mitigation strategies. HARM is flexible and could probably have been even better adapted to this project. The way I applied HARM in this project, it did not aid in the discovery of new vulnerabilities. HARM can be used together with other approaches and methods, such as code reviewing, which helped with discovering unknown vulnerabilities.

7.2 Future Work

To be able to continue to conduct digital exams with SEB without increasing the occurrence of cheating behaviors, it is vital to continue developing and researching. SEB will be continuously developed into newer versions, in which it is necessary to test existing, known and unknown vulnerabilities through penetration testing. Testing in a real life setting, where invigilating and network restrictions could be tested in practice, to assess which threats are most likely to be used and successful during ordinary exams. The more a digital exam solution is used, the more likely it is that malicious users will develop new approaches to exploit vulnerabilities, and an exam software can not be declared impenetrable for all future situations. Security assessment of the system needs to alway be proactive, to ensure limited cheating. The following list present some possible approaches to future work on this topic.

- Test cheating behaviors in real-real setting
- Fix the vulnerabilities that have been discovered in SEB
- Fix the vulnerabilities that have been discovered in SEB with Moodle
- Find new vulnerabilities in SEB
- Test cheating vulnerabilities in other digital exam solutions

- Investigate available resources that could be used to cheat
- Develop preventive mechanisms
- Social studies
- Social mitigation

It is necessary to mitigate the vulnerabilities that have been discovered in this thesis, regardless of whether or not I was able to exploit them. The vulnerabilities do not have to be mitigated with technical approaches, but it could be possible to use preventive mechanisms in an exam environment, outside of the digital SEB environment. New vulnerabilities can be discovered, by continuing the code analysis and testing. Also, to test in a more real-life setting, may show that some of the vulnerabilities are already mitigated, while other, new vulnerabilities becomes possible to exploit. If the use of SEB increase, the probability that one or more modified version of the software becomes available. It could be beneficial to do research regarding available versions, and how to prevent them from being used.

Finally, it could be advantageous to research the frequency and use of cheating behaviors in exams, and what makes a student choose to cheat, and how can they be motivated to avoid cheating behaviors. I find these topics particularly interesting, to be able to for example do an anonymous survey of students' experiences when it comes to cheating. A survey of these topics could present possible social mechanisms that could affect the occurrence of cheating.

Bibliography

[1] About SEB. http://safeexambrowser.org/about_overview_en.html. [Online; accessed 2016-03-29].

[2] Developer Manual SEB for Windows. http://safeexambrowser.org/windows/win_developer.html. [online; accessed 2015-10-14].

[3] Moodle. www.moodle.org. [Online; accessed 2016-4-15].

[4] Safe Exam Browser 2.1.2 for Windows, User Manual. http://safeexambrowser.org/windows/win_usermanual_en.html. [Online; accessed 2016-03-29].

[5] Safe Exam Browser for Windows, Release Notes. http://safeexambrowser.org/windows/win_release_notes_en.html. [Online; accessed 2016-04-11].

[6] Safe Exam Browser News. http://safeexambrowser.org/news_en.html. [Online; accessed 2016-03-29].

[7] Akin, B., Stender, S., and McGraw, G. (2005). Software Penetration Testing. In *IEE Security & Privacy*, volume 3, pages 84–87, Los Alamitos, CA, USA. IEEE Computer Society. [ISSN:1540-7993].

[8] Björklund, M. and Wenestam, C.-G. (1999). Academic Cheating: Frequency, Methods, and Causes. In *European Conference on Education Research ECER*, Lahti, Finland.

[9] Brenner, M. and Sandström, H. (2015). Datoriserad Tentamen.

[10] Choong, Y.-Y. and Theofanos, M. (2015). What 4,500+ People Can Tell You - Employees' Attitudes Toward Oganizational Password Policy Do Matter. In Tryfonas, T. and Askoxylakis, I., editors, *Human Aspects of Information Security, Privacy, and Trust: Third International Conference, HAS 2015, Held as Part of HCI International 2015*, pages 299–310, Los Angeles. Springer International Publishing. [ISBN: 978-3-319-20376-8].

[11] Clarke, N. (2015). A Rocky Road to Digital Assessment. http://www.inspera.com/seminar-2015. Presentation at Inspera Høstseminar.

[12] Cordova, J. L. and Thornhill, P. (2007). Academic Honesty and Electronic Assessment: Tools to Prevent Students from Cheating Online-Tutorial Presentation. *J. Comput. Sci. Coll.*, 22(5):52–54.

[13] Dawson, P. (2015). Five Ways to Hack and Cheat with Bring Your Own Device (BYOD) Electronic Examinations. In *The British Journal of Educational Technology.*

[14] Educational Development and Technology Unit of ETH Zurich (2015). Safe Exam Browser Source ode. `https://github.com/SafeExamBrowser`. [online; accessed 2015-10-28].

[15] Felton, M., Jegersberg, M., Hildring, G., Seim, M., Tho, M. L. B., Tande, L., Fjeldstad, A. G., Møller, K., Motland, J., and Veium, K. (2014). Digital Vurdering og Eksamen - en Juridisk Vurdering. `https://norgesuniversitetet.no/files/attachment/2830/digital-vurdering-eksamen-juridisk-versjon1.pdf`. Et samarbeidsprosjekt utført på oppdrag fra Ekspertgruppen for Digital vurdering og eksamen.

[16] Frankl, G., Schartner, P., and Zebedin, G. (2012). Secure Online Exams Using Students' Devices. In *Global Engineering Education Conference (EDUCON)*, pages 1–7, Marrakech. 2012 IEEE. [ISSN: 2165-9559].

[17] German, D. M. and González-Barahona, J. M. (2009). An Empirical Study of the Reuse of Software Licensed Under the GNU General Public License. In *Open Source Ecosystems: Diverse Communities Interacting: 5th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2009, Skövde, Sweden, June 3-6, 2009. Proceedings*, Berlin, Heidelberg. Springer Berlin Heidelberg.

[18] Halbherr, T., Reuter, K., Schneider, D., Schlienger, C., and Pendl, T. (2014). Making Examinations More Valid, Meaningful and Motivating: The Online Exam Service at ETH Zurich. In *European University Information Systems.* Center for Educational Development and Technologi, ETH Zurich.

[19] Hillier, M. (2014). E-Exams with Student Owned Device: Student Voices. In *Proceedings of the International Mobile Learning Festival 2015: Mobile Learning, MOOC's and 21th Century Learning*, pages 582–608, Hong Kong SAR China.

[20] Hillier, M. and Fluck, A. (2013). Arguing Again for E-exams in High Stakes Examinations. In Carter, H., Gosper, M., and Hedberg, J., editors, *Electric Dreams*, 30th ascilite Conference, pages 385–396. Sydney.

[21] Hovde, P. (2013). Digital eksamen ved NTNU, Sluttrapport fra Forstudiet. NTNU.

[22] Hunt, T. (2013). Safe Exam Browser Quiz Access Rule for Moodle - Functionality Summary. `https://github.com/moodleou/moodle-quizaccess_safeexambrowser/blob/v0.9/internaldoc/functionality.md`. [Online; accessed 2016-05-06].

[23] James, H. W. (1927). The Effect of Handwriting Upon Grading. *The English Journal*, 16(2):180–185. Published by: National Council of Teachers of English.

[24] Karpati, P., Opdahl, A. L., and Sindre, G. (2013). HARM: Hacker Attack Representation Method. In Cordeiro, J., Virvou, M., and Shishkov, B., editors, *Software and Data Technologies: 5th International Conference, ICSOFT 2010, Athens, Greece, July 22-24, 2010. Revised Selected Papers*, pages 156–175, Berlin, Heidelberg. Springer Berlin Heidelberg.

[25] King, C., Guyette, R., and Piotrowski, C. (2009). Online Exams and Cheating: An Empirical Analysis of Business Students' Views. In *The Journal of Educators Online, Volume 6, Number 1.* University of West Florida.

[26] Klein, J. and Taub, D. (2005). The Effect of Variations in Handwriting and Print on Evaluation of Student Essays. *Assessing Writing*, 10(2):134–148.

[27] Kordy, B., Kordy, P., Mauw, S., and Schweitzer, P. (2013). ADTool: Security Analysis with Attack–Defense Trees. In Joshi, K., Siegle, M., Stoelinga, M., and D'Argenio, P. R., editors, *Quantitative Evaluation of Systems: 10th International Conference, QEST 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, pages 173–176, Berlin, Heidelberg. Springer Berlin Heidelberg.

[28] Kordy, B., Mauw, S., Radomirovic, S., and Schweitzer, P. (2011). Foundations of Attack-Defense Trees. In Degano, P., Etalle, S., and Guttman, J., editors, *Formal Aspects of Security and Trust(FAST 2010)*, volume 6561 of *Lecture Notes in Computer Science*, pages 80–95. Springer.

[29] Langaard, M. (2015). Digital Eksamen på Mange Lokasjoner. `http://www.inspera.com/seminar-2015`. Presentation at Inspera Høstseminar.

[30] McCabe, D. (2005). Cheating Among College and University Students; A North American Perspective. *International Journal for Educational Integrity*, 1(1).

[31] Microsoft (2003). Chapter 3 Threat Modeling. `https://msdn.microsoft.com/en-us/library/ff648644.aspx`. [Online; accessed 2016-05-22].

[32] Microsoft (2016a). Clipboard functions. `https://msdn.microsoft.com/en-us/library/windows/desktop/ff468802(v=vs.85).aspx`. [online; accessed 2016-05-23].

[33] Microsoft (2016b). ManagementObjectSearcher Constructor (ObjectQuery). `https://msdn.microsoft.com/en-us/library/3x6at5a0(v=vs.110).aspx`. [online; accessed 2016-05-23].

[34] Mogey, N., Paterson, J., Burke, J., and Purcell, M. (2010). Typing Compared with Handwriting for Essay Rxaminations at University: Letting the Students Choose. *ALT-J, Research in Learning Technology*, 18(1):29–47.

[35] Mozilla. Mozilla Public License 1.1. `https://www.mozilla.org/en-US/MPL/1.1/`. [online; accessed 2016-05-30].

[36] Myhre, R., Brede, E., et al. (2013). UNINETT - Digital Eksamensavvikling.

[37] Nielsen, K. G., Petersen, L., Wallstedt, B., Basse, P., Hansen, P. S., Hansen, S. S., and Sørensen, D. M. (2014). Evaluation of Digital Assessment. In *EUNIS konference*. European University Association.

[38] Oates, B. (2006). *Researching Information Systems and Computing*. Sage Publications.

[39] Ogie, R. (2016). Bring Your Own Device: An Overview of Risk Assessment. *IEEE Consumer Electronics Magazine*, 5(1):114–119.

[40] O'Hara, K. J. and Kay, J. S. (2003). Open Source Software and Computer Science Education. *J. Comput. Sci. Coll.*, 18(3):1–7.

[41] OWASP (2016). Threat risk modeling. `https://www.owasp.org/index.php/Threat_Risk_Modeling`. [Online; accessed 2016-05-22].

[42] Palmer, C. C. (2001). Ethical Hacking. In *IBM Systems Journal, VOL 40, NO 3*, pages 769–780. IBM Research Division.

[43] Schneider, D. (2015a). Safe Exam Browser 2.0 How to (install, configure, deploy and use seb 2.0). `http://safeexambrowser.org/presentations/HowTo_SEB2.0.pdf`.

[44] Schneider, D. (2015b). Safe Exam Browser News and Future Roadmap. `http://www.inspera.com/seminar-2015`.

[45] Sessink, O., Beeftink, R., Tramper, J., and Hartog, R. (2004). Securing Web-Based Exams. In *Journal of Universal Computer Science*, volume 10.

[46] Søgaard, T. M. (2015). Cheating Threats in Digital BYOD Exams: A preliminary investigation. Trondheim. NTNU.

[47] Sheard, J., Dick, M., Markham, S., Macdonald, I., and Walsh, M. (2002). Cheating and Plagiarism: Perceptions and Practices of First Year IT Students. In *Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '02, pages 183–187, New York, NY, USA. ACM.

[48] Sindre, G. and Vegendla, A. (2015a). e-Exams and Exam Process Improvement. In *Utdanning og Didaktikk i IT-faga(UDIT)/ Norwegian Information Security Conference*, Ålesund, Norway. Published in Norsk Informatikkerkonferanse. ISSN 1892-0721, `http://ojs.bibsys.no/index.php/NIK/index`, 2015.

[49] Sindre, G. and Vegendla, A. (2015b). E-Exams Versus Paper Exams: A Comparative Analysis of Cheating-Related Security Threats and Countermeasures. In *Norwegian Information Security Conference (NISK)*.

[50] Sülzenbrück, S., Hegele, M., Rinkenauer, G., and Heuer, H. (2011). The Death of Handwriting: Secondary Effects of Frequent Computer Use on Basic Motor Skills. *Journal of Motor Behavior*, 43(3):247–251. DOI: 10.1080/00222895.2011.571727.

[51] Utdanningsdirektoratet (2009). Eksamen 2009 og 2010 - erfaringer og vurderinger. `http://www.udir.no/Tilstand/Forskning/Rapporter/Utdanningsdirektoratet/Erfaringer-av-eksamen-varen-2009-og-vurdering-av-eksamen-2010/`. ”[Online; accessed 2015-09-24]”.

[52] Vegendla, A., Sindre, G., and Søgaard, T. M. (2016). Extending HARM to Make Test Cases for Penetration Testing. In *The 6th International Workshop on Information Systems Security Engineering, WISSE'16*, Lubljana, Slovenia.

[53] Wang, L., Wond, E., and Xu, D. (2007). A Threat Model Driven Approach for Security Testing. In *Proceedings of the Third International Workshop on Software Engineering for Secure Systems*, SESS '07, pages 10–, Washington, DC, USA. IEEE Computer Society. [ISBN: 0-7695-2952-6].

[54] Wikipedia (2016). XXAMP. `https://en.wikipedia.org/wiki/XAMPP`. [Online; accessed: 2016-05-06].

Appendix A

Prestudy

A.1 Attack Tree

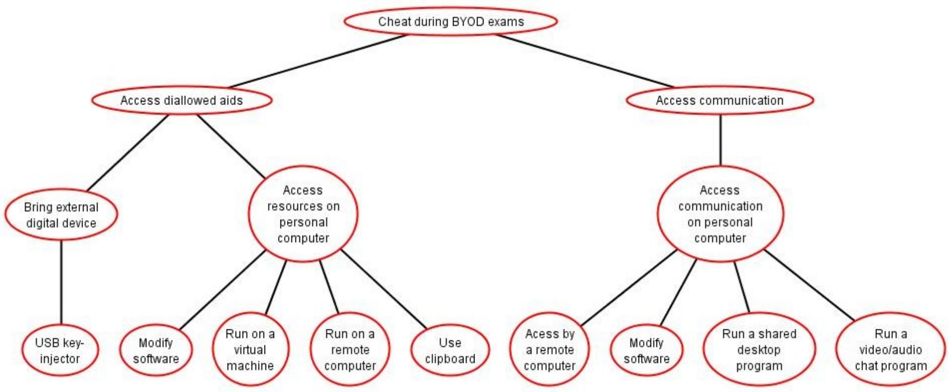

Cheat during BYOD exams
Access diallowed aids
Access communication
Bring external digital device
Access resources on personal computer
Access communication on personal computer
USB key-injector
Modify software
Run on a virtual machine
Run on a remote computer
Use clipboard
Acess by a remote computer
Modify software
Run a shared desktop program
Run a video/audio chat program

Figure A.1: Attack Tree from prestudy [46], describing the test approach.

A.2 Penetration test results

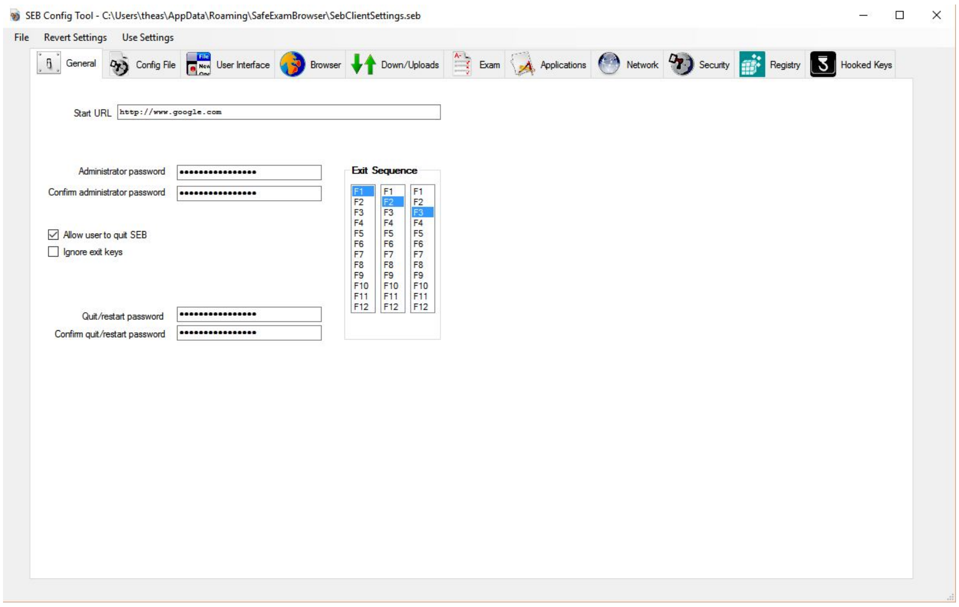
Threat Description	**Result**	**Comment**
Inject notes into exam software with USB key injector	Confirmed	We saved the script on the rubber ducky USB and the text string was injected into the web page open in SEB.
Modify source code to access resources and/or communication	Unconfirmed	This attack is impossible to confirm without testing with a LMS with a SEB Browser Exam key check.
Run SEB on a virtual machine	Unsuccessful	When initiating SEB, a pop-up window appears, informing you that SEB has detected a virtual machine and will not work. SEB will run in a virtual machine if the configuration file enables it.
Run SEB on a remote computer	Confirmed	We managed to control SEB from a remote computer, while using SEB.
Use clipboard to import notes into exam software	Unsuccessful	We were not able to right click or use CTRL+P to paste the clipboard content into SEB.
Get assistance by being accessed from a remote computer	Confirmed	We managed to control and access an SEB exam environment from a remote computer.
Get assistance by sharing desktop	Unsuccessful	Neither Google Hangout nor Skype showed SEB with remote desktop, when it was initiated.
Get assistance by communicating with audio/video	Confirmed	Both examinee and assistant can hear each other and use their microphones. The assistant can also see the examinee on camera during a video conversation, but the examinees only sees the SEB environment.
Access resources and/or communications by modifying the configuration file	Unconfirmed	This is impossible to confirm without running a LMS inside SEB with the SEB Browser key functionality activated.

Table A.1: Result description from prestudy [46]

Appendix B

Configurations

B.1 ConfigTool Screenshots



SEB Config Tool - C:\Users\theas\AppData\Roaming\SafeExamBrowser\SebClientSettings.seb

Start URL http://www.google.com

Administrator password ••••••••••••••••

Confirm administrator password ••••••••••••••••

☑ Allow user to quit SEB

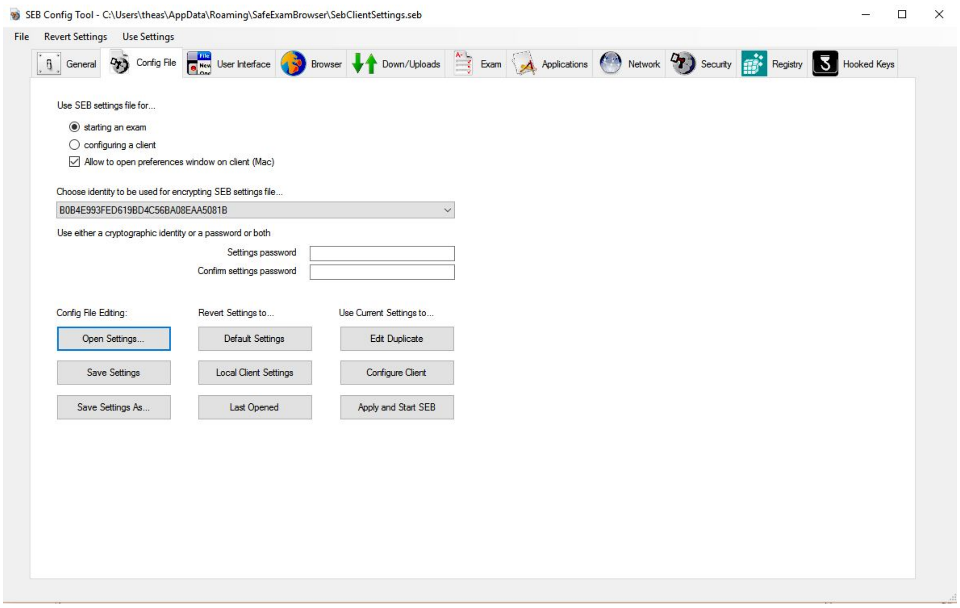
☐ Ignore exit keys

Quit/restart password ••••••••••••••••

Confirm quit/restart password ••••••••••••••••

Exit Sequence: F1, F2, F3

Figure B.1: Screenshot of SEB ConfigTool, contents of the 'General' tab.



Use SEB settings file for...

- [✓ starting an exam]
- configuring a client
- [✓] Allow to open preferences window on client (Mac)

Choose identity to be used for encrypting SEB settings file...

B0B4E993FED619BD4C56BA08EAA5081B

Use either a cryptographic identity or a password or both

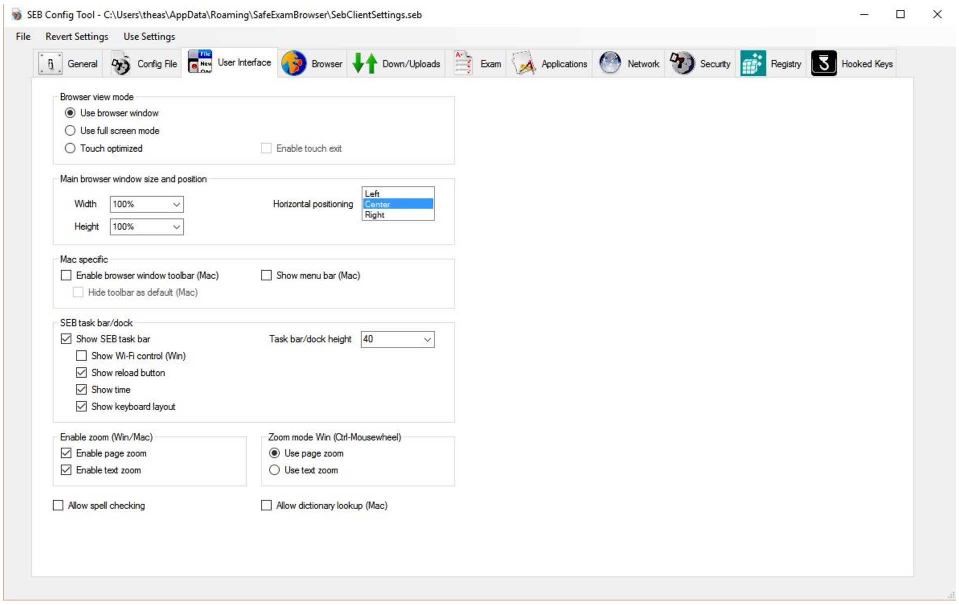
Settings password	
Confirm settings password	

Config File Editing:

Revert Settings to...

Use Current Settings to...

Figure B.2: Screenshot of SEB ConfigTool, contents of the 'Config File' tab.



Browser view mode

- [✓ Use browser window]
- Use full screen mode
- Touch optimized
- [ ] Enable touch exit

Main browser window size and position

Width	100%
Height	100%
Horizontal positioning	Left / [✓ Center] / Right

Mac specific

- [ ] Enable browser window toolbar (Mac)
- [ ] Hide toolbar as default (Mac)
- [ ] Show menu bar (Mac)

SEB task bar/dock

- [✓] Show SEB task bar
- [ ] Show Wi-Fi control (Win)
- [✓] Show reload button
- [✓] Show time
- [✓] Show keyboard layout

Task bar/dock height: 40

Enable zoom (Win/Mac)

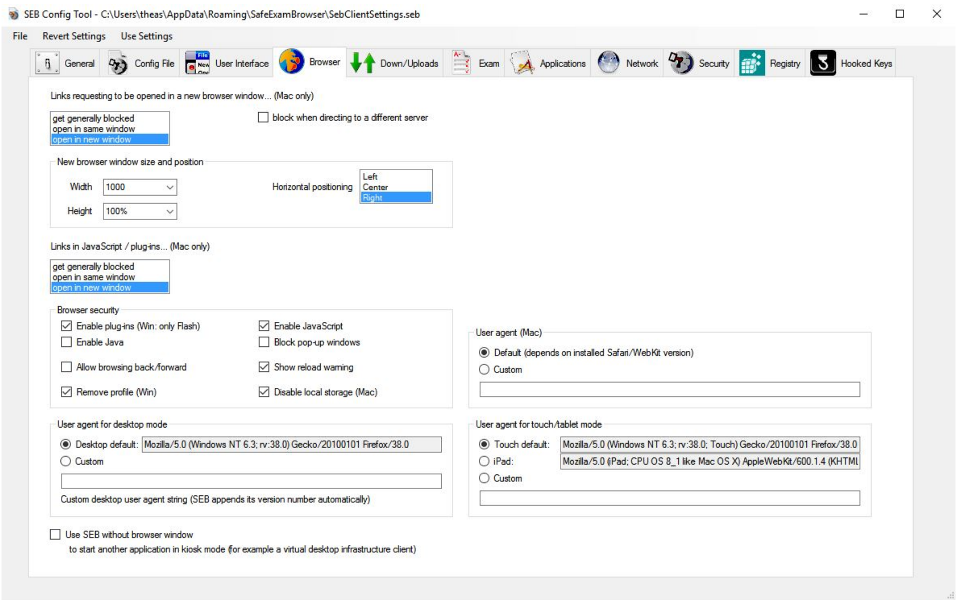
- [✓] Enable page zoom
- [✓] Enable text zoom

Zoom mode Win (Ctrl-Mousewheel)

- [✓ Use page zoom]
- Use text zoom

- [ ] Allow spell checking
- [ ] Allow dictionary lookup (Mac)

Figure B.3: Screenshot of SEB ConfigTool, contents of the 'User Interface' tab.



Links requesting to be opened in a new browser window... (Mac only)

get generally blocked
open in same window
[✓ open in new window]

☐ block when directing to a different server

New browser window size and position

Width 1000

Height 100%

Horizontal positioning: Left / Center / [✓ Right]

Links in JavaScript / plug-ins... (Mac only)

get generally blocked
open in same window
[✓ open in new window]

Browser security

☑ Enable plug-ins (Win: only Flash)
☑ Enable JavaScript
☐ Enable Java
☐ Block pop-up windows
☐ Allow browsing back/forward
☑ Show reload warning
☑ Remove profile (Win)
☑ Disable local storage (Mac)

User agent (Mac)

● Default (depends on installed Safari/WebKit version)
○ Custom

User agent for desktop mode

● Desktop default: Mozilla/5.0 (Windows NT 6.3; rv:38.0) Gecko/20100101 Firefox/38.0
○ Custom

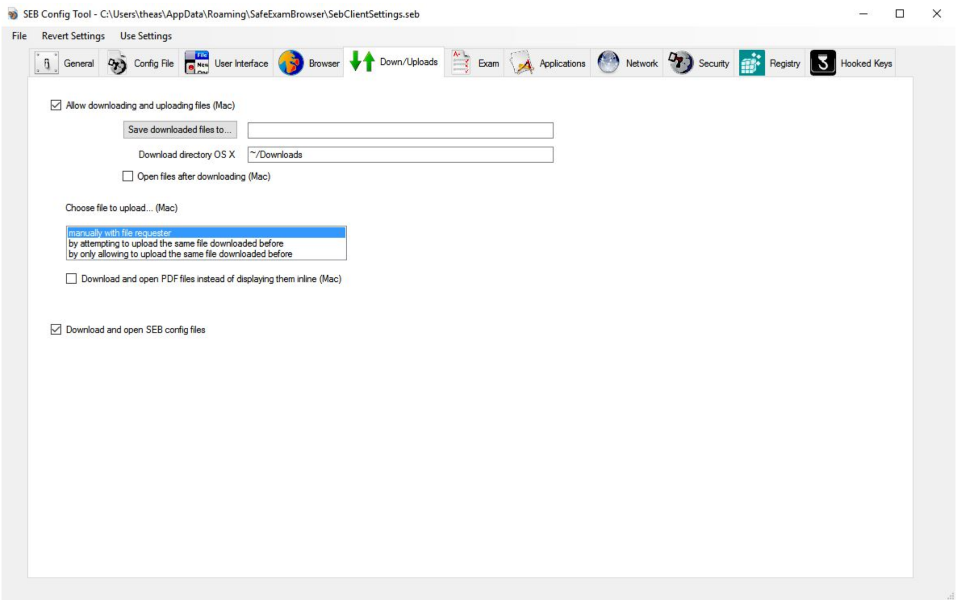
Custom desktop user agent string (SEB appends its version number automatically)

User agent for touch/tablet mode

● Touch default: Mozilla/5.0 (Windows NT 6.3; rv:38.0; Touch) Gecko/20100101 Firefox/38.0
○ iPad: Mozilla/5.0 (iPad; CPU OS 8_1 like Mac OS X) AppleWebKit/600.1.4 (KHTML
○ Custom

☐ Use SEB without browser window
to start another application in kiosk mode (for example a virtual desktop infrastructure client)

Figure B.4: Screenshot of SEB ConfigTool, contents of the 'Browser' tab.



☑ Allow downloading and uploading files (Mac)

Save downloaded files to...

Download directory OS X ~/Downloads

☐ Open files after downloading (Mac)

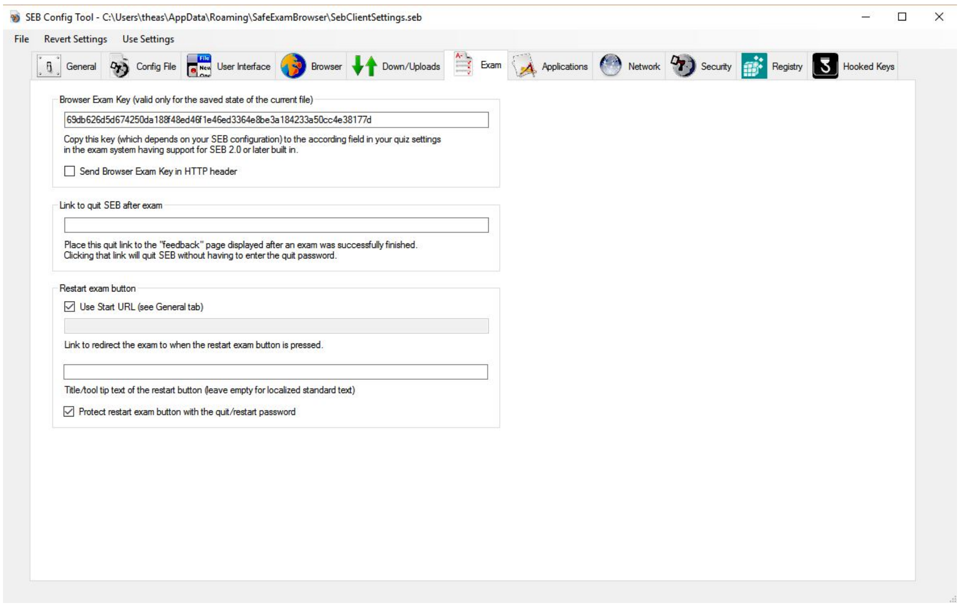
Choose file to upload... (Mac)

[✓ manually with file requester]
by attempting to upload the same file downloaded before
by only allowing to upload the same file downloaded before

☐ Download and open PDF files instead of displaying them inline (Mac)

☑ Download and open SEB config files

Figure B.5: Screenshot of SEB ConfigTool, contents of the 'Down/Uploads' tab.



Browser Exam Key (valid only for the saved state of the current file)

69db626d5d674250da188f48ed46f1e46ed3364e8be3a184233a50cc4e38177d

Copy this key (which depends on your SEB configuration) to the according field in your quiz settings in the exam system having support for SEB 2.0 or later built in.

☐ Send Browser Exam Key in HTTP header

Link to quit SEB after exam

Place this quit link to the "feedback" page displayed after an exam was successfully finished. Clicking that link will quit SEB without having to enter the quit password.

Restart exam button

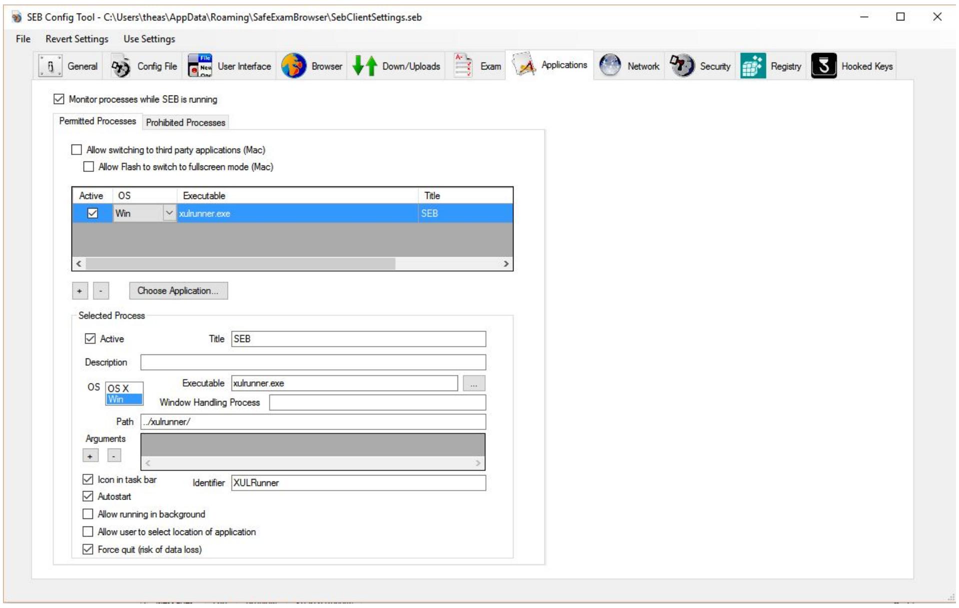
☑ Use Start URL (see General tab)

Link to redirect the exam to when the restart exam button is pressed.

Title/tool tip text of the restart button (leave empty for localized standard text)

☑ Protect restart exam button with the quit/restart password

Figure B.6: Screenshot of SEB ConfigTool, contents of the 'Exam' tab.



☑ Monitor processes while SEB is running

☐ Allow switching to third party applications (Mac)

☐ Allow Flash to switch to fullscreen mode (Mac)

Active	OS	Executable	Title
☑	Win	xulrunner.exe	SEB

Selected Process

☑ Active

Title: SEB

Description:

OS: OS X / Win

Executable: xulrunner.exe

Window Handling Process:

Path: ../xulrunner/

Arguments

☑ Icon in task bar

Identifier: XULRunner

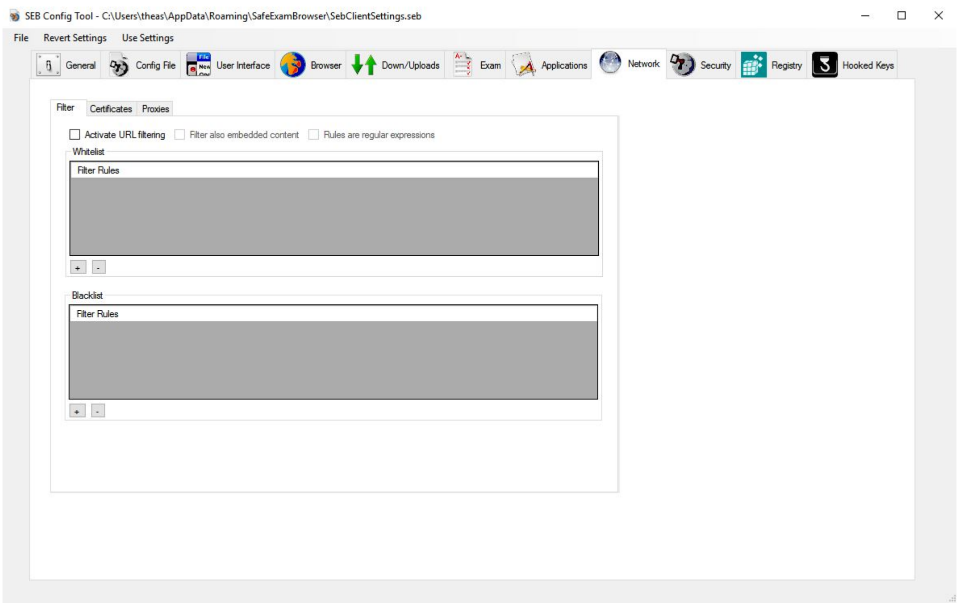
☑ Autostart

☐ Allow running in background

☐ Allow user to select location of application

☑ Force quit (risk of data loss)

Figure B.7: Screenshot of SEB ConfigTool, contents of the 'Application' tab.



☐ Activate URL filtering ☐ Filter also embedded content ☐ Rules are regular expressions

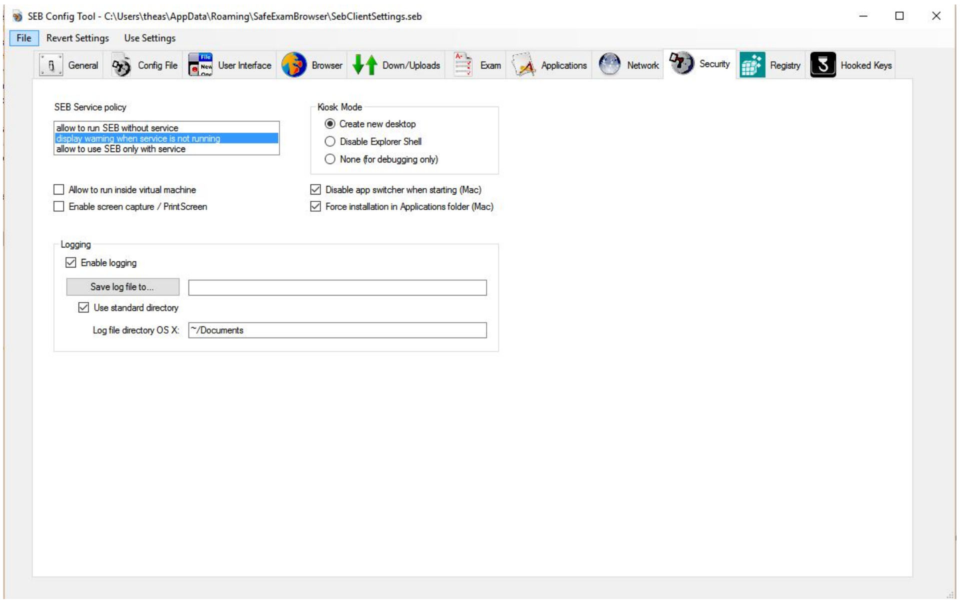
Whitelist

Filter Rules

Blacklist

Filter Rules

Figure B.8: Screenshot of SEB ConfigTool, contents of the 'Network' tab.



SEB Service policy

- allow to run SEB without service
- display warning when service is not running
- allow to use SEB only with service

Kiosk Mode

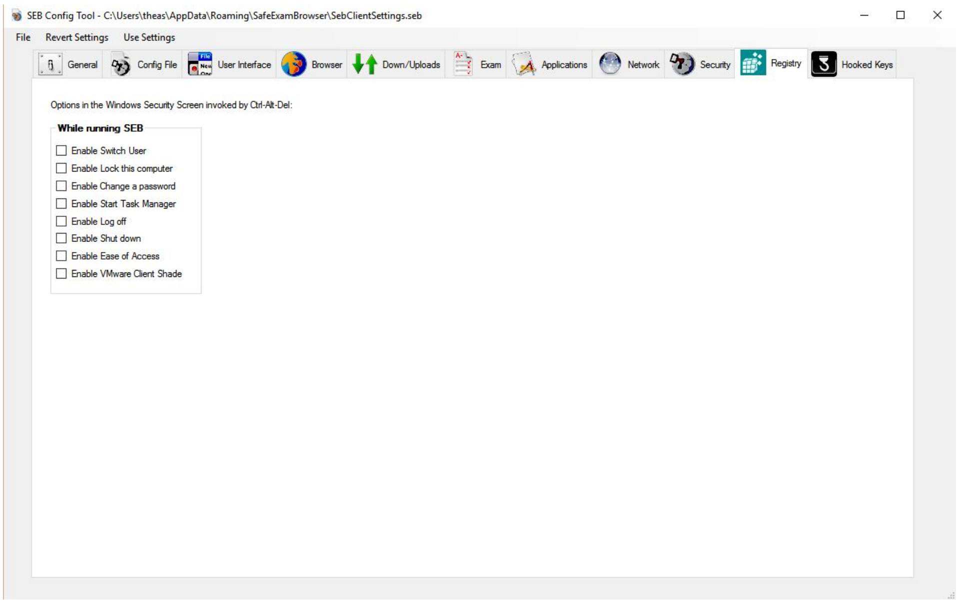
● Create new desktop
○ Disable Explorer Shell
○ None (for debugging only)

☐ Allow to run inside virtual machine
☐ Enable screen capture / PrintScreen
☑ Disable app switcher when starting (Mac)
☑ Force installation in Applications folder (Mac)

Logging

☑ Enable logging
☑ Use standard directory
Log file directory OS X: ~/Documents

Figure B.9: Screenshot of SEB ConfigTool, contents of the 'Security' tab.

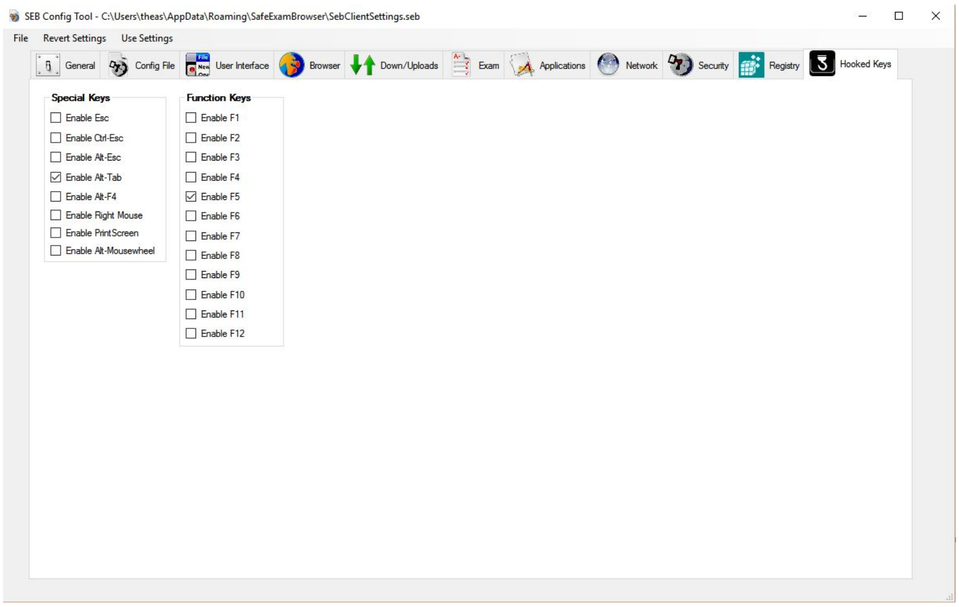


Options in the Windows Security Screen invoked by Ctrl-Alt-Del:

While running SEB

- ☐ Enable Switch User
- ☐ Enable Lock this computer
- ☐ Enable Change a password
- ☐ Enable Start Task Manager
- ☐ Enable Log off
- ☐ Enable Shut down
- ☐ Enable Ease of Access
- ☐ Enable VMware Client Shade

Figure B.10: Screenshot of SEB ConfigTool, contents of the 'Registry' tab.



Special Keys

- ☐ Enable Esc
- ☐ Enable Ctrl-Esc
- ☐ Enable Alt-Esc
- ☑ Enable Alt-Tab
- ☐ Enable Alt-F4
- ☐ Enable Right Mouse
- ☐ Enable PrintScreen
- ☐ Enable Alt-Mousewheel

Function Keys

- ☐ Enable F1
- ☐ Enable F2
- ☐ Enable F3
- ☐ Enable F4
- ☑ Enable F5
- ☐ Enable F6
- ☐ Enable F7
- ☐ Enable F8
- ☐ Enable F9
- ☐ Enable F10
- ☐ Enable F11
- ☐ Enable F12

Figure B.11: Screenshot of SEB ConfigTool, contents of the 'Hooked Keys' tab.

B.2 Configuration File (.seb-file) contents

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
        <dict>
                <key>originatorVersion</key>
                <string>SEB_Win_2.1</string>
                <key>startURL</key>
                <string>http://www.google.com</string>
                <key>sebServerURL</key>
                <string />
                <key>hashedAdminPassword</key>

<string>8C6976E5B5410415BDE908BD4DEE15DFB167A9C873FC4BB8A81F6F2AB448A918</string>
        <key>allowQuit</key>
        <true />
        <key>ignoreExitKeys</key>
        <true />
        <key>hashedQuitPassword</key>
<string>A665A45920422F9D417E4867EFDC4FB8A04A1F3FFF1FA07E998E86F7F7A27AE3</string>
        <key>exitKey1</key>
        <integer>2</integer>
        <key>exitKey2</key>
        <integer>10</integer>
        <key>exitKey3</key>
        <integer>5</integer>
        <key>sebMode</key>
        <integer>0</integer>
        <key>browserMessagingSocket</key>
        <string>ws://localhost:8706</string>
        <key>browserMessagingPingTime</key>
        <integer>120000</integer>
        <key>sebConfigPurpose</key>
        <integer>0</integer>
        <key>allowPreferencesWindow</key>
        <true />
        <key>browserViewMode</key>
        <integer>0</integer>
        <key>mainBrowserWindowWidth</key>
        <string>100%</string>
        <key>mainBrowserWindowHeight</key>
        <string>100%</string>
        <key>mainBrowserWindowPositioning</key>
        <integer>1</integer>
        <key>enableBrowserWindowToolbar</key>
        <false />
        <key>hideBrowserWindowToolbar</key>
        <false />
        <key>showMenuBar</key>
        <false />
        <key>showTaskBar</key>
        <true />
        <key>taskBarHeight</key>
        <integer>40</integer>
        <key>touchOptimized</key>
        <false />
        <key>enableZoomText</key>
        <true />
        <key>enableZoomPage</key>
        <true />
        <key>zoomMode</key>
        <integer>0</integer>
        <key>allowSpellCheck</key>
        <false />
        <key>allowDictionaryLookup</key>
        <false />
        <key>showTime</key>
        <true />
        <key>showInputLanguage</key>
        <true />
        <key>enableTouchExit</key>
        <false />
        <key>browserScreenKeyboard</key>
        <false />
        <key>newBrowserWindowByLinkPolicy</key>
        <integer>2</integer>
        <key>newBrowserWindowByScriptPolicy</key>
        <integer>2</integer>
        <key>newBrowserWindowByLinkBlockForeign</key>
        <false />
        <key>newBrowserWindowByScriptBlockForeign</key>
```

```
<false />
<key>newBrowserWindowByLinkWidth</key>
<string>50%</string>
<key>newBrowserWindowByLinkHeight</key>
<string>80%</string>
<key>newBrowserWindowByLinkPositioning</key>
<integer>2</integer>
<key>enablePlugIns</key>
<true />
<key>enableJava</key>
<false />
<key>enableJavaScript</key>
<true />
<key>blockPopUpWindows</key>
<false />
<key>allowBrowsingBackForward</key>
<true />
<key>removeBrowserProfile</key>
<true />
<key>removeLocalStorage</key>
<true />
<key>enableSebBrowser</key>
<true />
<key>showReloadButton</key>
<true />
<key>showReloadWarning</key>
<true />
<key>browserUserAgentWinDesktopMode</key>
<integer>0</integer>
<key>browserUserAgentWinDesktopModeCustom</key>
<string />
<key>browserUserAgentWinTouchMode</key>
<integer>0</integer>
<key>browserUserAgentWinTouchModeCustom</key>
<string />
<key>browserUserAgent</key>
<string />
<key>browserUserAgentMac</key>
<integer>0</integer>
<key>browserUserAgentMacCustom</key>
<string />
<key>allowDownUploads</key>
<true />
<key>downloadDirectoryOSX</key>
<string>~/Downloads</string>
<key>downloadDirectoryWin</key>
<string />
<key>openDownloads</key>
<false />
<key>chooseFileToUploadPolicy</key>
<integer>0</integer>
<key>downloadPDFFiles</key>
<false />
<key>downloadAndOpenSebConfig</key>
<true />
<key>examKeySalt</key>
<data>PDwZzUd1beCqTB3ahpZC9M2FYDoUsHwgtwTe+0NxwgA=</data>
<key>browserExamKey</key>
<string />
<key>browserURLSalt</key>
<true />
<key>sendBrowserExamKey</key>
<false />
<key>quitURL</key>
<string />
<key>restartExamURL</key>
<string />
<key>restartExamUseStartURL</key>
<true />
<key>restartExamText</key>
<string />
<key>restartExamPasswordProtected</key>
<true />
<key>monitorProcesses</key>
<true />
<key>allowSwitchToApplications</key>
<false />
<key>allowFlashFullscreen</key>
<false />
<key>permittedProcesses</key>
<array>
        <dict>
                <key>active</key>
                <true />
                <key>autostart</key>
                <true />
                <key>iconInTaskbar</key>
                <true />
```

```
                        <key>runInBackground</key>
                        <false />
                        <key>allowUserToChooseApp</key>
                        <false />
                        <key>strongKill</key>
                        <true />
                        <key>os</key>
                        <integer>1</integer>
                        <key>title</key>
                        <string>SEB</string>
                        <key>description</key>
                        <string />
                        <key>executable</key>
                        <string>xulrunner.exe</string>
                        <key>path</key>
                        <string>../xulrunner/</string>
                        <key>identifier</key>
                        <string>XULRunner</string>
                        <key>windowHandlingProcess</key>
                        <string />
                        <key>arguments</key>
                        <array></array>
                </dict>
        </array>
        <key>prohibitedProcesses</key>
        <array></array>
        <key>enableURLFilter</key>
        <false />
        <key>enableURLContentFilter</key>
        <false />
        <key>URLFilterRules</key>
        <array></array>
        <key>URLFilterEnable</key>
        <false />
        <key>URLFilterEnableContentFilter</key>
        <false />
        <key>blacklistURLFilter</key>
        <string>www.facebook.com</string>
        <key>whitelistURLFilter</key>
        <string />
        <key>urlFilterTrustedContent</key>
        <false />
        <key>urlFilterRegex</key>
        <false />
        <key>embeddedCertificates</key>
        <array></array>
        <key>proxySettingsPolicy</key>
        <integer>0</integer>
        <key>proxies</key>
        <dict>
                <key>ExceptionsList</key>
                <array></array>
                <key>ExcludeSimpleHostnames</key>
                <false />
                <key>AutoDiscoveryEnabled</key>
                <false />
                <key>AutoConfigurationEnabled</key>
                <false />
                <key>AutoConfigurationJavaScript</key>
                <string />
                <key>AutoConfigurationURL</key>
                <string />
                <key>FTPPassive</key>
                <true />
                <key>HTTPEnable</key>
                <false />
                <key>HTTPPort</key>
                <integer>80</integer>
                <key>HTTPProxy</key>
                <string />
                <key>HTTPRequiresPassword</key>
                <false />
                <key>HTTPUsername</key>
                <string />
                <key>HTTPPassword</key>
                <string />
                <key>HTTPSEnable</key>
                <false />
                <key>HTTPSPort</key>
                <integer>443</integer>
                <key>HTTPSProxy</key>
                <string />
                <key>HTTPSRequiresPassword</key>
                <false />
                <key>HTTPSUsername</key>
                <string />
                <key>HTTPSPassword</key>
                <string />
                <key>FTPEnable</key>
                <false />
                <key>FTPPort</key>
                <integer>21</integer>
                <key>FTPProxy</key>
                <string />
                <key>FTPRequiresPassword</key>
                <false />
                <key>FTPUsername</key>
                <string />
                <key>FTPPassword</key>
                <string />
                <key>SOCKSEnable</key>
                <false />
                <key>SOCKSPort</key>
                <integer>1080</integer>
                <key>SOCKSProxy</key>
                <string />
                <key>SOCKSRequiresPassword</key>
                <false />
                <key>SOCKSUsername</key>
                <string />
                <key>SOCKSPassword</key>
                <string />
                <key>RTSPEnable</key>
                <false />
                <key>RTSPPort</key>
                <integer>554</integer>
                <key>RTSPProxy</key>
                <string />
                <key>RTSPRequiresPassword</key>
                <false />
                <key>RTSPUsername</key>
                <string />
                <key>RTSPPassword</key>
                <string />
        </dict>
        <key>sebServicePolicy</key>
        <integer>1</integer>
        <key>allowVirtualMachine</key>
        <false />
        <key>createNewDesktop</key>
        <true />
        <key>killExplorerShell</key>
        <false />
        <key>allowUserSwitching</key>
        <true />
        <key>enableAppSwitcherCheck</key>
        <true />
        <key>forceAppFolderInstall</key>
        <true />
        <key>enableLogging</key>
        <true />
        <key>logDirectoryOSX</key>
        <string>~/Documents</string>
        <key>logDirectoryWin</key>
        <string />
        <key>allowWlan</key>
        <true />
        <key>insideSebEnableSwitchUser</key>
        <false />
        <key>insideSebEnableLockThisComputer</key>
        <false />
        <key>insideSebEnableChangeAPassword</key>
        <false />
        <key>insideSebEnableStartTaskManager</key>
        <false />
        <key>insideSebEnableLogOff</key>
        <false />
        <key>insideSebEnableShutDown</key>
        <false />
        <key>insideSebEnableEaseOfAccess</key>
        <false />
        <key>insideSebEnableVmWareClientShade</key>
        <false />
        <key>hookKeys</key>
        <true />
        <key>enableEsc</key>
        <false />
        <key>enableCtrlEsc</key>
        <false />
        <key>enableAltEsc</key>
        <false />
        <key>enableAltTab</key>
        <true />
        <key>enableAltF4</key>
        <false />
        <key>enableStartMenu</key>
        <false />
        <key>enableRightMouse</key>
        <false />
        <key>enablePrintScreen</key>
        <true />
        <key>enableAltMouseWheel</key>
        <true />
        <key>enableF1</key>
        <false />
        <key>enableF2</key>
        <false />
        <key>enableF3</key>
        <false />
        <key>enableF4</key>
        <false />
        <key>enableF5</key>
        <false />
        <key>enableF6</key>
        <false />
        <key>enableF7</key>
        <false />
        <key>enableF8</key>
        <false />
        <key>enableF9</key>
        <false />
        <key>enableF10</key>
        <false />
        <key>enableF11</key>
        <false />
        <key>enableF12</key>
        <false />
        </dict>
</plist>
```

Appendix C

Logfiles

C.1 Examle contens of seb.log

```
************************************
initialize logfile 4/11/2016, 1:09:02 PM
************************************
seb: messageSocket open: [object Event]
seb: set ping intervall 120000
seb: create alert controller...
seb: showContent...main
seb: messageSocket handled: SEB.close
seb: host force shutdown
seb: try to remove everything from profile folder:
C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\cache2
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\cert8.db
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\cert_override.txt
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\compatibility.ini
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\cookies.sqlite
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\crashes
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\key3.db
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\mimeTypes.rdf
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\parent.lock
err: seb: [Exception... "Component returned failure code:
0x8052000e (NS_ERROR_FILE_IS_LOCKED) [nsIFile.remove]"
nsresult: x8052000e (NS_ERROR_FILE_IS_LOCKED)"
location: "JS frame :: resource://modules/seb.jsm :: shutdownObserver.observe :: line
108"  data: no]
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\
permissions.sqlite
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\places.sqlite
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\pluginreg.dat
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\prefs.js
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\safebrowsing
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\secmod.db
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\
sessionCheckpoints.json
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\
SiteSecurityServiceState.txt
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\startupCache
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\webappsstore.sqlite
seb: remove: C:\Users\theas\AppData\Roaming\SafeExamBrowser\Profiles\xulstore.json
```

C.2 Example contents SebClient.log

```
Message: −−−−−−−−−−− INITIALIZING SEB − STARTING SESSION −−−−−−−−−−−−−−− Exception type:
Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Arguments: C:\Program Files (x86)\SafeExamBrowser\SafeExamBrowser.exe Exception type:
Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Attempting to InitSebSettings Exception type:  Details:
Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 0 Event type:  Event detail code: 0 Message:
Could not load SebClientSettigs.seb from the Program Data
directorySystem.IO.FileNotFoundExceptionCould not find file
'C:\ProgramData\SafeExamBrowser\SebClientSettings.seb'.
Exception type:  Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message: User Name:
theas Host Name: localhost Port Number: 57016
Send Interval: 100 Recv Timeout: 100 Num Messages: 3
SebClientConfigFileDirectory:
C:\Users\theas\AppData\Roaming\SafeExamBrowser\ SebClientConfigFile:
C:\Users\theas\AppData\Roaming\SafeExamBrowser\SebClientSettings.seb
Exception type:  Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 0 Event type:  Event detail code: 0 Message:
SEB client configuration set in InitSebSettings().
Exception type:  Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
OS Version: 2050 Exception type:  Details:  Event
date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Successfully InitSebSettings Exception type:
Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Attempting to InitSEBDesktop Exception type:
Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Clipboard cleaned. Exception type:  Details:  Event
date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Successfully InitSEBDesktop Exception type:  Details:
Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
entering Opensebform Exception type:  Details:  Event
date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
attempting to position the taskbar Exception type:
Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Current display DPI setting: 96 and scale factor: 1
Exception type:  Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Taskbarheight from settings: 40 Current taskbar
height: 40 Exception type:  Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 0 Event type:  Event detail code: 0 Message:
OnLoad EventArgs: C:\Program Files
(x86)\SafeExamBrowser\SafeExamBrowser.exe Exception type:
Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
finished taskbar positioning Exception type:
Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Win32_ComputerSystem Manufacturer: Acer, Model: Aspire
5951G Exception type:  Details:  Event date: 12−Apr−16 16:01:08 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message: initializing wcf service connection
```

C.3 Example contents SebConfig.log

```
Event code: 2 Event type:  Event detail code: 0 Message:
Loading the default local client settings
Exception type:  Details:  Event date: 11-Apr-16 13:46:21 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Loading the default local client settings Exception type:  Details:
Event date: 11-Apr-16 14:09:05 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Loading the default local client settings Exception type:  Details:
Event date: 11-Apr-16 16:20:45 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Loading the default local client settings Exception type:  Details:
Event date: 11-Apr-16 16:24:41 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Loading the default local client settings Exception type:  Details:
Event date: 11-Apr-16 16:29:07 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Loading the default local client settings Exception type:  Details:
Event date: 12-Apr-16 11:45:29 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Loading the default local client settings Exception type:  Details:
Event date: 12-Apr-16 12:45:54 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Loading the default local client settings Exception type:  Details:
Event date: 12-Apr-16 13:11:03 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Loading the default local client settings Exception type:  Details:
Event date: 12-Apr-16 13:15:49 Additional data:
Event code: 2 Event type:  Event detail code: 0 Message:
Loading the default local client settings Exception type:  Details:
Event date: 12-Apr-16 15:08:28 Additional data:
```

C.4 Example content from sebwindowsservice.log

```
5/11/2016 11:21:31 AM: Set Registry Key DisableLockWorkstation to 1
5/11/2016 11:21:31 AM: Set Registry Key DisableChangePassword to 1
5/11/2016 11:21:31 AM: Set Registry Key DisableTaskMgr to 1
5/11/2016 11:21:31 AM: Set Registry Key HideFastUserSwitching to 1
5/11/2016 11:21:31 AM: Set Registry Key NoLogoff to 1
5/11/2016 11:21:31 AM: Set Registry Key NoClose to 1
5/11/2016 11:21:31 AM: Set Registry Key EnableShade to 0
5/11/2016 11:21:31 AM: Set Registry Key EnableShadeHorizon to False
5/11/2016 11:21:31 AM: Set Registry Key EaseOfAccess to SebDummy.exe
5/11/2016 11:21:31 AM: Set windows update to False

*******
5/11/2016 11:24:02 AM: Set Registry Key DisableLockWorkstation to 0
5/11/2016 11:24:02 AM: Set Registry Key DisableChangePassword to 0
5/11/2016 11:24:02 AM: Set Registry Key DisableTaskMgr to 0
5/11/2016 11:24:02 AM: Set Registry Key HideFastUserSwitching to
5/11/2016 11:24:02 AM: Set Registry Key NoLogoff to 0
5/11/2016 11:24:02 AM: Set Registry Key NoClose to 0
5/11/2016 11:24:02 AM: Set Registry Key EnableShade to 1
5/11/2016 11:24:02 AM: Set Registry Key EnableShadeHorizon to True
5/11/2016 11:24:02 AM: Set Registry Key EaseOfAccess to
5/11/2016 11:24:02 AM: Set windows update to True
*******
(FROM ATTACK:)
5/27/2016 4:27:11 PM: Set windows update to False
5/27/2016 4:27:54 PM: Set windows update to True
```

Appendix D

Licences

D.1 SEB License

Safe Exam Browser for Windows 2.1

Taken from [4]

D.2 Moodle License

Taken from the Moodle source code