NTNU
Norwegian University of
Science and Technology

Cheating at Digital Exams

Vulnerabilities and Countermeasures

Aleksander Heintz

Master of Science in Computer Science
Submission date: June 2017
Supervisor: Guttorm Sindre, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

In today's world where everything is increasingly done on computers, university exams are traditionally lagging behind and are still conducted using pen and paper for the most part. Recently, however, NTNU has started a slow migration process towards digital exams using Safe Exam Browser and Inspera Assessment on private devices brought by the students on the exam day.

Digital exams have the potential to resolve some of the issues with old style pen and paper exams, however, that is not without bringing its own set of problems into the mix. In this thesis, I look at specifically how digital exams using Safe Exam Browser and Inspera Assessment software might enable students to cheat on the exams in ways that were not possible on pen and paper exams.

The main focus is on the interactions between the three components Safe Exam Browser, Inspera Assessment and the students own computer to look at both technical and theoretical ways that students can cheat during digital exams at NTNU. And where possible, I provide solutions/mitigations to any problems I do discover.

The current solution employed by NTNU to facilitate digital exams for students does have several problematic security vulnerabilities. There are several ways a student can manage to cheat on a digital exam that they would not be able to do using a traditional pen and paper exam. However, the work to improve digital exams are important, and there are definitely improvements that can be made. While

it's true that the current implementation is problematic, with improvements to Safe Exam Browser and Inspera Assessment as well as the setup used by the university it's entirely plausible to make digital exams almost as secure as traditional pen and paper exams.

Sammendrag

Vi lever i en verden hvor flere og flere oppgaver utføres på en datamaskin. Et av unntakene er universitetseksamener, hvor den vanligste fremgangsmåten fremdeles er tradisjonelle papireksamener. NTNU har startet en forsiktig migrasjonsprosess mot digitale eksamener, der det benyttes Safe Exam Browser og Inspera Assessment som installeres på studentenes egne datamaskiner.

Digitale eksamener har potensiale til å løse flere av problemene knyttet til papireksamener, men de bringer også med seg en del utfordringer. I denne oppgaven ser jeg nærmere på sikkerhetshull knyttet til bruk av Safe Exam Browser og Inspera Assessment, noe som gjør det mulig å jukse under gjennomføring av en digital eksamen.

I hovedfokus er samspillet mellom Safe Exam Browser, Inspera Assessment og studentenes egne datamaskiner, og hvordan dette kan brukes til å jukse på digitale eksamener ved NTNU. Jeg har analysert både teoretiske og praktiske former for juks og sett på mulige måter disse kan forhindres.

Masteren avdekker flere kritiske sikkerhetshull i den digitale løsningen, og det finnes flere måter studenter kan jukse på som ikke ville vært mulig ved tradisjonelle papireksamener. Arbeidet med å forbedre digitale eksamener er derfor viktig, og det finnes flere forbedringer som kan gjøres. På tross av at dagens løsning har noen utfordringer vil dette bli bedre ved oppdateringer, samt utbedringer av ek-

samenoppsettet som brukes av NTNU. Over tid er det derfor fullt mulig at digitale eksamener vil kunne tilby bedre sikkerhet enn tradisjonelle papireksamener.

Acknowledgements

I would like to thank my supervisor Guttorm Sindre for feedback and information I've received during the research for and writing of this thesis, some of which has helped me figure out what to focus on and what to look for. I would also like to thank Inspera Assessment for the help they have provided me during the work done on this thesis. Much of that work could not have been done without their support.

Contents

List of Figures

Glossary

Digital exam A digital exam is an exam executed on a computer (or another digital device), as opposed to old style pen and paper exams.

Inspera Inspera Assessment is a norwegian company that provides a website in which it is possible to author and take exams.

Safe environment Safe Exam Browser creates what is known as a safe environment (and conversely when not running in SEB it is considered an unsafe environment). Safe environment indicates that students will only be allowed to access what the author of the exam intended the student to access.

Acronyms

MITM	Man-in-the-middle attack.
LMS	Learning management system. In the digital exam case of NTNU this is Inspera Assessment. The learning management system is used in combinations with SEB for the exams.
SEB	Safe Exam Browser.
NTNU	Norwegian University of Science and Technology - Norges teknisk-naturvitenskapelige universitet.
FEIDE	Felles Elektronisk IDEntitet is a common identity solution employed by the Norwegian department of education.
BYOD	Bring your own device. One of the forms of digital exams employed by universities and other educational institutions around the world.

1

Introduction

This chapter will provide an introduction to the topics discussed in this thesis, as well as provide the motivation and scope of the thesis.

1.1. Background

Digital exams in and of themselves is not a new concept. The fact that we do everything else on a computer today inevitably leads to the desire for the exams themselves to be done digitally. Digital exams comes with several advantages over pen and paper exams. Some of these advantages include but are not limited to [22][8]:

- People might be more used to writing on computers than with pen and paper.

- No need to use large amounts of paper for the exams . Both the questions and answers are provided digitally, instead of using what is probably thousands of pieces of paper per exam.
- Remove possibility of exam papers getting lost, wrong papers (like the answers instead of the questions) being given to the students.
- A lot of user errors might become impossible when a computer system can dictate what action can be done at what time.

However, just because digital exams can provide advantages over pen and paper exams does not mean that they are the be-all and end-all for exams, and that simply making the exams digital would solve all problems exams today faces. While it is true that a good amount of problems with pen and paper exams might be mitigated by making them digital, digital exams provide their own set of challenges and issues. Some of these are simply new logistical challenges like the fact that all students are suddenly required to have a laptop available for taking the exam (something that most people do have today, but nonetheless a few might be excluded). Another issue is that the software used for the exams (Safe Exam Browser) does not currently support Linux, meaning that people who do use Linux on their computers (which for university students is probably a higher percentage than for the general populous) might have problems taking the exams. Another issue is the fact that the exams can only be held in a room with enough power outlets for all the students. Other errors might include:

- Power outage. Exams might not be doable because of a power outage preventing people from using their computers. Today however this is highly unlikely, not to mention the fact that even though it is technically doable, one might not want to do pen and paper exams during a power outage either.

- Internet failure. Same as with power outage, if the internet is down for some reason then exams cannot be executed. This is slightly more likely than a power outage, but still highly improbably in Norway today. However, this is something which could make digital exams impossible to perform while having no impact at all on pen and paper exams.
- Software failure. Problems with the LMS (Inspera Assessment in the case of NTNU) can result in people being unable to take the exam, or worse displacement/mismanagement of the results. It could also result in people being given the wrong exams or no exam at all. And if the LMS had a security breach, a student might be able to alter his/her responses after the exam has ended (when said student has regained access to the course book and/or other learning material).

Another very real issue with digital exams is that they introduce new ways by which students might attempt to cheat at exams, which is mainly what I'll be looking at in this paper.

1.1.1. Different Ways to do Digital Exams

The solution currently employed by NTNU to facilitate digital exams is based on students bringing their own devices and then running software specified by the university to execute the exam. This is known as "bring your own device" or BYOD exams. An alternative to BYOD exams might be to build a computer lab and fill it with university owned computers or thin clients. Another different approach is simply having home exams. Each of these approaches has their own benefits and drawbacks, some of them being:

Cost

Building out a computer lab and filling it with screens, keyboards, and computers (even if they are thin clients) can be very costly. Not to mention the fact that there is a high upkeep in that computers needs to be kept up to date, and that thousands of students using them every year might result in a high degree of wear resulting in things having to be replaced often. Another less visible cost is the fact that in order to run a lab of computers, staff that knows how to set them up and manage them is required. As opposed to simply buying computers which is a one-time cost (if we ignore the need to ever replace any of them), having to pay several full-time employees to manage the computers might be just as expensive.

On the other end of the scale, you have home exams, which depending on the exam in question might be entirely automated. In the case of a multiple choice exam for instance, students could sign in from home during the right time window, answer the questions, and the computer could immediately let them know what grade they achieved. There would be no need to pay for invigilators, examiner or locale. BYOD exams are somewhere in the middle of the two.

In a BYOD exam, the university needs to provide a place to house the exam, and invigilators. This is the same cost as with traditional pen and paper exams though. However, with pen and paper exams, students don't typically need access to a power outlet. Another factor in the cost of a BYOD exam is that sometimes things go wrong, and there is a problem with a students' hardware and its compatibility with the software required to run the exam. In such cases, the university is likely expected to provide a small number of computers or thin clients to enable these students to also take the exam. However, this is still much cheaper than having to provide a separate computer for each student.

Another factor of the cost is who creates and manages the software used to perform the exams. Creating robust software is not cheap, and requires both time and money. It also needs to be maintained over time, and likely also evolved to fit new needs that arise along the way.

Required Expertise

If the university wants to run its own in-house service solution for digital exams it needs server administrators that know how to ensure that everything is running smoothly, and what to do when an error is encountered. In the case that the university wants to run its own computer lab, skilled technicians are required to manage the computers and the network required for all of them to be connected to the internet.

Even in the case where the home exam solution is opted for, a certain degree of expertise is needed. After all, exams still need to be made, users need to be given access the exams, and answers to the exams needs to be graded. While a large part of the responsibility can be outsourced to another company, someone at the university needs to know who things work. This is the case though, regardless if the solution is made in-house or purchased from a third party.

Security

Security can have several meanings depending on the context. When talking about an online exam solution, hacking is a security concern. If the exam solution is offline however, and the exams are either handled entirely on the clients, or on a closed intranet, the risk of external hacking can be entirely eliminated. Security in the context of digital exams can also mean how hard it is to cheat at the exams. With home exams for instance, it's notoriously hard to prevent certain kinds of cheating, such as collaborating with other people. It's next to impossible to verify that the student taking the exam is doing so alone without having some kind of surveillance, which to a large extent defeats the whole point of home exams. Collaboration is much harder to do during an exam where the students are placed in the same room with oversight from invigilators.

There are many more ways students can cheat during digital exams, several of which have been looked at by previous papers [23][24], and several others that will be examined in great detail in this thesis.

1.1.2. Bring Your Own Device (BYOD)

This thesis is primarily focused on the current solution to digital exams employed by NTNU, namely a BYOD solution using Safe Exam Browser and Inspera Assessment. Most of the issues and exploits discussed in this thesis are therefore only present during this form of examination. No assumption should be taken regarding the validity of any of the exploits discovered during this thesis using any other form of digital examination.

1.2. Motivation

Eventually, it's likely that most, if not all exams held by universities in Norway, and likely also around the world, will be entirely digital. This changes the well-known formula of paper and pen exams, and introduces new uncertainties to exams. While pen and paper exams have been tried and tested over the ages, digital exams are still a fairly new concept [8]. Not to mention the fact that while there are probably multiple ways one can do pen and paper exams, the paths one can take towards digital exams are fairly limitless. Do you make your own solution? Do you buy somebody else's? Or do you hire a contractor to figure the whole thing out? Each of these have advantages and disadvantages. NTNU has opted for a solution using Safe Exam Browser (SEB) to ensure that students cannot access other resources available on their own computers, and Inspera Assessment to do the presentation of exams and collection of exam results.

There are already several studies looking at how secure Safe Exam Browser is [23][24]. However, this is only one part of the solution selected by NTNU in their digital exam solution and there are other parts that need to be assessed as well.

Digital exams are also a challenge in and of themselves simply because they are new. When something has been done for ages everyone knows how it works. The students know what to expect, the invigilators know what to check and the teachers know how to write the exams. According to a study by Hiller [8], pen-and-paper exams might have become a self-reinforcing phenomenon in that students train themselves at pen-and-paper exams by using pen-and-paper to take notes, re-write notes for revision and do test exams. Considering how high-stakes exams can be, it's not weird that people want to keep the status quo.

With digital exams, the way to create, execute and grade the exam all changes. Just a simple thing like the fact that on a written exam, the students write their candidate number on the top of every exam paper, along with the page number, the course id, date, etc. Granted, most of these fields are to ensure that if an accident like someone dropping a stack of exams down a flight of stairs happen, they can be reassembled correctly, which is no longer an issue with digital exams. Yet the fact still remains that the invigilators know to check that the student signed the correct candidate number on each page. However, in a digital exam as employed by NTNU today, there is no place to write your candidate number, with the exception of on the paper that is brought by the invigilators when the exam starts. On the actual exam site, you sign in using FEIDE. New procedures, however small they might be, still has to be learned.

According to the same study by Hillier [8], even though a lot of the students who partook in the study choose to write the exam using pen and paper, amongst the ones who choose to do the exam on a computer they reported several advantages. Some of these included the ability to write faster, not having to get pains in the wrists, ability to edit text that has already been written and several others. Another advantage reported by the students is the fact that some of them have bad handwriting. When writing the exam digitally, the didn't have to worry about their grades being badly affected because the examiner couldn't read their handwriting.

Sindre and Vegendla [22] also states several strong points for digital exams. They list advantages for all stages of the exam process. For making the tests it would enable easier sharing of questions between learning institutions, to the ability to include multimedia and interactive material. For executing the test it would reduce cost due to less paper, and might make it easier to adapt to students with special needs. And for assessment, it would make it easier to get the answers to the graders as they could just be transmitted digitally, and easier reading than handwritten answers. Handwriting in pen-and-paper exams might also potentially be used to identify the student which could not happen in a digital exam. Lastly, in an earlier paper Hillier and Fluck argues that it's important to get a good computer-based way of conducting exams, else it might prevent other positive development that could occur from adopting digital systems in teaching and learning in general [9].

Another important motivational factor is the fact that preventing cheating at high-stakes exams is incredibly important. According to a paper on the issue, the more cheating is perceived by students as taking place, the more likely they are themselves to cheat because they don't want to be put at an unfair disadvantage [12]. Another study shows that as much as 1 in 5 self-reports having cheated on a test or examination [11]. According to Harpp and Hogan [7], one of the problems with academic cheating is the fact that it's hard to do anything when cheating is detected. Accusing students of cheating can be costly, and the evidence is often not solid. This leads to administrators being reluctant to move forward when potential cheating is discovered. Therefore, if doable, a better approach than catching cheaters is to make sure that cheating can't occur in the first place. Cheating might also adversely affect the image of company/software used to prevent cheating as they did not manage to achieve their sell-point. This might have monetary consequences because companies/software that is perceived as having weak cheating mitigation might not get customers.

1.3. Scope of the Thesis

In this paper I will look at possible security exploits within the exam setup used by NTNU. I will mostly be looking at Inspera Assessment and the interactions between the Inspera LMS and SEB. I will not be attempting to do any data break-in at Inspera or other disruptive actions that would potentially affect an exam like a Denial of Service attack (DoS) or the more common Distributed Denial of Service attack (DDoS). I will also not be attempting to get access to any resources from Inspera that I as a student taking an exam should not have access to, with the exception of attempting to access the exam itself from an unsafe environment.

I will look at both technical and non-technical solutions to cheat at digital exams as they are held today at NTNU primarily with the goal of assessing resources I should not have access to during the exam. The focus will primarily be on Bring Your Own Device digital exams as that is the format currently employed by NTNU to provide digital exams.

1.4. Research Questions

The goal of this master thesis is to look at different ways a student can practically cheat at a digital exam at NTNU. The main focus will be trying to access resources that are disallowed during exams, such as pre-made notes, online resources such as Wikipedia, or communication resources such as chat. I also consider it important that any exploit I discover is not only reliable but also hard to discover.

RQ1 What are technical cheating vulnerabilities of the digital exam solution employed by NTNU?

RQ2 How can the ability for students to cheat during digital exam solutions be reduced sufficiently so that digital exams are no more vulnerable to cheating than their pen-and-paper counterpart?

RQ1 relates to attempting to work around the security features in place by Safe Exam Browser and Inspera Assessment, attempting to get access to resources I should not have access to during the exam. With regards to **RQ2** it will to some extend depend on what is discovered during **RQ1**.

1.5. Research Method

I do not believe there is any single research method that could adequately answer the research questions stated in Section 1.4, thus several research methods described below were employed.

1.5.1. Penetration Testing

I will be attempting to penetrate the security mechanisms implemented by Inspera Assessment in their LMS to enforce that students utilize SEB when taking the exam, as well as getting access to resources I should not have access to while doing exams in SEB on the Inspera website. This consists of making multiple and varied attempts at forging information or modifying the code running in my browser to fool either the Inspera LMS or SEB into doing something that they were not intended for.

1.5.2. Literature Study

With regards to **RQ2**, and how to solve any issues discovered, I will look to other solutions to the digital exam problem. Digital exams, while not old are also not entirely new, so there are other cases to look at, and other educational institutions that have used different methods to give digital exams to their students. As such I will try to look at other solutions and see if they can be used to solve our problems.

1.6. Ethical Issues

As with all research pertaining with attempting to get around security systems, there is the chance that severe security issues might be discovered. Any such discovery carries the risk of the public being alerted to ways to mitigate said security systems, which in the case of Safe Exam Browser and Inspera Assessment is likely to mean students figuring out how to cheat at the exams. In spite of this however, it is still imperative that the research is done, because any security vulnerability that exists is likely to be discovered at some point regardless. As with what is the norm when doing security research, any security vulnerabilities that I've discovered were properly disclosed to the related parties before being publicized. As such, it should be possible to resolve the vulnerabilities before the information becomes available to the public. This leads to better safety, and a better overall product [19] [3].

At the same time, the role of NTNU in the relationship between NTNU, SEB and Inspera Assessment is that of a customer (whether it is paying to use the product or not). As such, NTNU should be privy to do security audits of the applications it has selected for its digital exam solution. In the case of SEB, it's open source, and anyone can do what they want with it, therefore there isn't much of a problem with any attacks used against it. However, Inspera is a hosted service that runs on servers not owned by NTNU. Attempting forceful break-ins against their service might, for instance, prevent other people elsewhere from taking their exams. This is obviously not desired, as such, I will not be performing any such actions without the explicit permission from Inspera Assessment itself.

Any exploits discovered in this thesis (including an early copy of the thesis itself) has been properly disclosed to both Inspera Assessment and one of the lead developers of Safe Exam Browser prior to the publishing of this thesis.

1.7. Report Outline

Chapter 2: Background Research and Related Work: In this chapter I'll give an introduction to previous work done on the topic as well as explain how the different technologies in use by NTNU for doing exams work.

Chapter 3: Attempted Exploits: In this chapter I'll give a overview of planned and executed attacks against Safe Exam Browser and Inspera Assessment. I will describe how the exploits are executed, and what issues occurred.

Chapter 4: Proposed Solutions: In this chapter I'll discuss some potential solutions to the problems discovered in the previous chapter. I'll try to look at what the actual problems is (and not just the perceived ones), and I'll also look at how others have implemented digital exam solutions.

Chapter 5: Conclusion and Future Work: In this chapter I'll conclude on the thesis, and look at some potential future work.

2

Background Research and Related Work

In this chapter I'll talk about the different technologies used in this report, previous research done on the subject as well as background research done.

2.1. Technologies

In this section I will describe the different technologies used in this report, how different parts of them work, and how they connect together.

2.1.1. Safe Exam Browser

Safe Exam Browser (SEB) is an application developed to provide what is known as "safe environments" or more accurately controlled environments to allow students to take the exam on their own devices. SEB is currently available for both Windows and Mac users, and it contains several technologies to prevent users from getting access to things they should not have access to. SEB is highly configurable, allowing the institution/test provider to create an encrypted configuration file which is used to start the exam. Some of the things that can be configured:

- Whether or not users should be allowed to exit the exam midway through.
- Whether or not users should be allowed to minimize SEB during the exam (access other parts of their computer).
- Whether or not users should be allowed access to the internet (and if so, which sites).
- Whether or not SEB should block keyboard shortcuts such as `Alt+F4`.
- A password that should be known only to invigilators allowing preemptive exiting of the exam.
- A list of valid root certificates to allow.
- And many many more.

Windows

The windows version of Safe Exam Browser is written in C# using the .NET framework. It contains a separate application for running SEB and creating configuration files. It uses XULRunner as its browser component.

Mac

The Mac version of Safe Exam Browser is written in Objective-C using the Cocoa framework. The configuration file editor is part of the same application that runs the browser (you have to start SEB first to edit/author configuration files). It uses WebKit as its browser component.

2.1.2. Browser Request Hash

Browser Request Hash is an HTTP header generated by SEB and sent with every request to any web server to validate that the user is using SEB with the right configuration and build. SEB generates this value by first generating what it calls a Browser Exam Key which is then concatenated with the URL of the request and then hashed using SHA256. The Browser Exam Key itself is generated by hashing all the files of the current running SEB instance (including its browser components) and then combining that with the SEB configuration file in question. The result is then used to generate a tag using HMAC with SHA256 as the hash function.

The Browser Exam Key is shared with any LMS with SEB support prior to the exam, and used by the LMS to validate that the user is in fact using SEB. This is done by taking the incoming request URL, concatenated it with the Browser Exam Key (just as SEB does on the client side) and then hashing it with SHA256. Then the generated request hash is checked against the request hash header transmitted from SEB, and if they do not match the request is denied.

2.1.3. SHA-256

SHA-256 is part of the SHA-2 family of cryptographic hashing algorithm. SHA stands for Secure Hashing Algorithm. All of the SHA algorithms are what is what is considered a one-way-function, that is, a function that only transforms data one way. This ensures that given any hash it's prohibitively expensive to compute the input value of any of the SHA functions given the resulting output value. Over the years however, some of the algorithms have been discovered to contain flaws that reduce their safety, however, the SHA-2 family of algorithms is still considered safe to use by NIST [4].

2.1.4. XULRunner

XULRunner is a deprecated [1][16] Mozilla runtime package [15] used to bootstrap feature-rich applications such as Firefox and Thunderbird. SEB for Windows uses XULRunner as its main browser component to create a chromeless (or custom chromed) browser window inside the kiosk mode frame that is SEB itself.

2.1.5. WebKit

WebKit is the browser engine used by the Safari browser, which is the default browser on both macOS and iOS. It's developed primarily by Apple and a fork of the WebKit project named Blink is also what powers Chromium based browsers such as Google Chrome. WebKit is easily integrated as a "browser component" into any Cocoa project. It is used by SEB to create the browser view on Macs.

2.1.6. Node.js

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine [18]. It is used to create all kinds of different applications, but was traditionally designed to enable the usage of JavaScript on the server. Node is used in this paper to be a reverse proxy server described in a later chapter. There is nothing specific about this proxy server that makes node a requirement, and it could just as easily have been written in Python or Java for instance, it was simply picked because it fulfills the requirements of being cross-platform, and I am more familiar with it than the alternatives.

2.2. Previous Works

This paper is built on previous research into SEB and digital exams already performed by other students at NTNU. Several attempts on working around Safe Exam Browser by way of tools such as remote desktops, virtual machines, etc. have been attempted [23][24]. Attempts have also been made at modifying Safe Exam Browser such that security measures could be toggled off, therefore I was initially not going to look at this. However, during my attempts at accessing Inspera Assessment without using Safe Exam Browser, I looked at the source code for Safe Exam Browser to understand how it generated the Browser Request Hash to figure out why my code wasn't working. Looking at the code I hypothesized that it would actually be doable to use a modified version of Safe Exam Browser and have it generate the same Browser Request Hash, and some quick testing later proved that to be the case.

2.2.1. Cheating in Digital Exams

Another thing to look at is how prevalent cheating is in digital exams as opposed to pen-and-paper exams. There have been multiple studies done on the topic, and the results are somewhat conflicting. A study on academic honesty and online learning found no significant differences in cheating behavior of students between digital and pen-and-paper exams [5]. Another study on distance learning the same year found that "most students cheat", and that they are more likely to cheat online [10]. However, this study covers student behavior in general, with questions about cheating during a course in general, not specifically at exams.

A newer study from 2009 found the opposite result, that students were more likely to cheat on traditional courses than online courses [25]. The study also provided the probability of students cheating on tests, which was 0.0013, as opposed to any form of academic cheating which was much higher.

A fourth study found only a small difference between cheating on live vs. online classes, with 32.1% cheating in live classes, and 32.7% cheating on online classes [28]. This study also conflates cheating on tests and assignments. Students were also asked whether they themselves thought that they and others would be more likely to cheat in online or live classes, in which an overwhelming majority of the students thought that they or others would be more likely to cheat in online classes. Since however, the data doesn't necessarily support this, it might be a conception issue, or it may just be that digital exams are too new so people haven't had the opportunity to cheat on them.

3 Attempted exploits

In this chapter I will go through the different exploits I've attempted and discovered. It will focus mostly on the technical side of things, for the most part ignoring the fact that the exam problem does not need a purely technical solution, and even if we can find a purely technical exploit, there may be non-technical reasons that make such an exploit unfeasible. As an example, if for some reason playing loud music while running Safe Exam Browser would allow you to access resources you should not be able to otherwise, this would obviously not be a practical exploit because the invigilators would shut you down immediately if you attempted to use it during an actual exam. However, in a purely technical bubble, it would still be a perfectly valid exploit. I will discuss this more in Section 4.1.

3.1. Injecting Code into the Safe Environment

One of the ways to cheat at a digital exam in which a safe environment is prohibiting you from accessing your files and the Internet, is to modify the code running inside the safe environment itself. Instead of trying to break out of the safe environment or access the exam without using the safe environment, we try to trick the safe environment into believing code we wrote is actually part of what was intended to be executed as part of the examination. If we are successful in doing so, it would effectively render the safe environment useless as we would be able to access any resource available on the Internet. Depending on how this code injection is done, it would also be possible to get access to local resources.

3.1.1. Man in the Middle Attack

One of the unique things about a browser (and working with the Internet in general) is the fact that the browser executes code written by strangers sent to us over a wire we have little to no control over. Traditionally, software came with the computer, delivered by a company that you trusted and handed over as a physical object (in the form of a CD or floppy disk, or as part of the computer itself). In order to breach that software and get your computer to execute code that you did not intend for it to execute, you would either have to get physical access to your computer, the shop where you bought your software, or the physical device used to transmit the software itself.

After we started using the Internet for distribution of software, a lot more points of failure was introduced. What happens when you download a program from a website like `example.com` is (simplified) the following steps:

1. Your computer makes a DNS request for the domain name `example.com`.
2. Your computer gets a response with the IP address for `example.com`.
3. Your computer attempts to open a connection to the IP returned by the DNS request.
4. The server which your computer believes to be `example.com` answers the request.
5. Your computer receives the file requested.

All of the steps above can be intercepted. For traditional software you download, this has bad implications, because it can cause you to download virus software thinking it was something else. However, binary applications you download can generally mitigate this problem by also giving out a checksum of the software itself (it does require some manual work on the part of the user to verify it though). On the other hand, websites is another form of software, transferred as text, and executed by the browser without any regard for whether or not it may be what the user intended. This means that if we can get the browser to believe that the answer to the request sent to `example.com` is something we made up (instead of what actually is the answer from the `example.com` server) we can get the browser to do more or less whatever we want it to. This means that even if we are using Safe Exam Browser (which is still a browser), and it prevents us from accessing files and folders we have on our computer, and also prevents us from accessing the Internet at large, by intercepting the request Safe Exam Browser makes to Inspera Assessment we can get it to do our bidding, instead of what it was intended to do.

Normally this attack is something that would be called as a Man in the Middle attack and is a well-known attack vector against users who try to access websites. The reason it is called Man in the Middle attack is that it would typically be performed by someone who has access to hardware that is between you and the

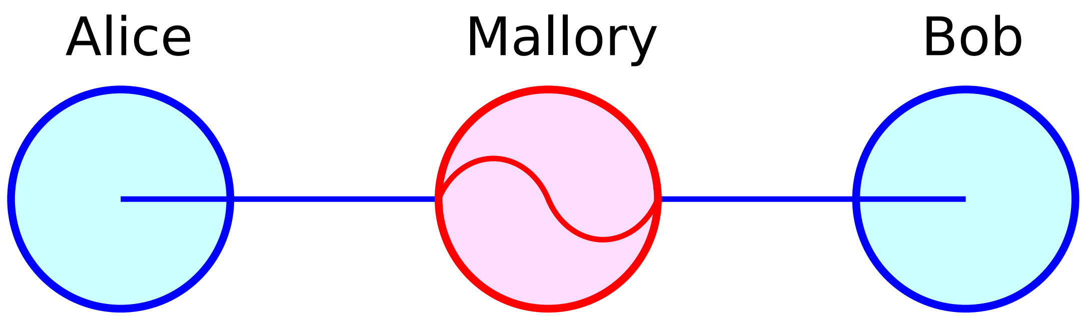


Figure 3.1: An illustration of a typical man in the middle attack. Made by Wikimedia Commons user Miraceti [13].

server you are trying to access. See Fig. 3.1 for example. For instance, in the request above to `example.com`, if the attacker had access to my router, he could instruct the router to reply to the DNS request for `example.com` with an IP address that he controls. Our browser would then happily connect to that IP address thinking it's connecting to `example.com` and take whatever it got sent its way as what the user wanted to execute. Alternatively, instead of modifying the DNS reply, he could listen for requests to the IP address that is `example.com`, and hijack the requests by not transmitting them to their final destination, instead replying with his own reply. Especially tricky is the fact that the attacker, in this case, knows the request you made to `example.com`, including any session data or username and passwords you sent with the request, meaning he could send the same request to `example.com`, get back the correct response, modify the response and then send it back to you. In this case you, would likely have little to no clue of the fact that you did not get what you asked for, because for all intents and purposes you did. You just got a little something extra as a bonus, normally not visible to you at all.

However, this technique is well documented. It has existed for years, and the solution to prevent Man in the Middle attacks is fairly simple. We use HTTPS. The solution here is two-fold. The first part is simply applying encryption. By using public key cryptography we are able to encrypt our requests in such a way that an attacker is not able to see what we are requesting, they can also not see the response. And although they would be able to modify both the request and the response, since both are encrypted, any modification would result in the request being invalid. If a weak encryption is used, it might still be able to modify the requests and responses, but most certificate providers nowdays recommend a key length of at least 2048 bits RSA encryption. This means that in practice, tampering with requests is not doable.

However, just encryption is not good enough. It would still be easily doable for an attacker to respond with an IP address they control to the DNS query for `example.com`. In such a case, if we only had encryption, our browser would connect to the server controlled by the attacker instead of the one we intended to connect to, and the result would be just as bad. However, HTTPS also provides a mechanism to ensure that the server you are connected to is actually the one you intended to connect to, which is achieved through certificate chains. When you connect to `example.com` over HTTPS, your browser request from the server proof that it is actually `example.com` (and not somebody else). The way this works is that the owner of `example.com` is issued a certificate, proving that he indeed does own `example.com`. This certificate is then installed on the server(s) that host `example.com`, so that they can prove that they are indeed responsible for hosting `example.com`. And the way that your browser knows to trust the certificate is because it has been signed by another certificate, which has been signed by another certificate all the way down to what is known as a root certificate, and that root certificate is installed on your computer. That is to say, in order for the attacker to hijack requests for `example.com`, he must both manage to give custom replies to DNS queries, and at the same time he must get someone who is trusted by the owner of a root certificate on your computer (or the owner of the root certificate itself) to validate the fact that he owns `example.com` (even though he doesn't). These things put together makes Man in the Middle attacks generally unfeasible as long as websites are using the secure HTTPS protocol.

3.1.2. Creating Your Own Certificate Authority

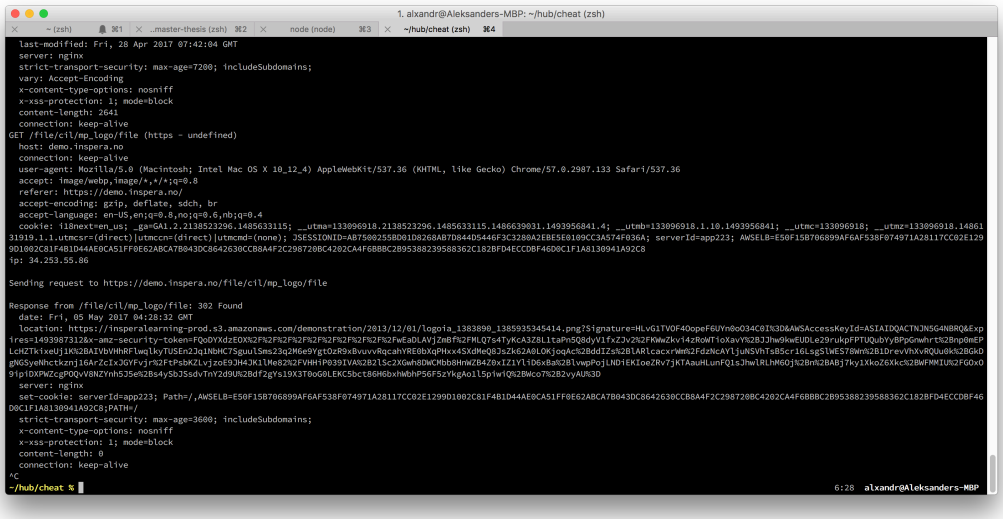
As explained above, HTTPS is generally considered safe against man in the middle attack because strong encryption and a trusted certificate chain ensure that when Safe Exam Browser connects to Inspera Assessment, it is indeed Inspera Assessment that replies. If this is not the case, Safe Exam Browser will throw an error message (which incidentally will freeze up your computer without the ability to exit safe exam browser or shut down normally). However, while that is normally the case, the security measures made to prevent Man in the Middle attacks were intended to stop people who have access to networking hardware between you and the server you're trying to access from being able to read and modify your requests and responses. They were however, not intended to prevent people who have physical access to your computer from so doing, and most certainly they were not indented to prevent you from attacking yourself.

The thing to remember is that while normally you would have an attacker trying to get access to data or trick you into running code you don't want to, in this case, we do want to run our custom code, therefore there should normally be no good reason for our computer to prevent us from so doing. Except that in this case we are running an exam using Safe Exam Browser, and running our custom code would be cheating. But still, these security measures have not been designed to prevent us from modifying or monitoring requests on our own machine, in fact, there is software written to enable just this. It turns out that being able to inspect network requests while developing things that sends network requests is quite useful. However, if those network requests are encrypted it would lose its usability. The way to get around this is to create our own server with its own certificate signed by our own certificate authority, and then use that as a proxy server for whatever secure resource we are attempting to access. All we then have to do is to install the certificate authority certificate (often called a root certificate) on our computer and we are all set.

The reason this works is that when we create and install a root certificate authority (henceforth known as `Cheat CA`), and use it to sign our own certificate for `example.com` (or any other website), when the browser connects to our proxy webserver and asks for proof that it is indeed `example.com`, the server responds with the certificate signed by `Cheat CA`. The browser then checks who signed it, and looks for the signer in the installed certificates, however since we installed `Cheat CA` on the computer previously the browser concludes that the certificate can be trusted (because we explicitly told it we trust `Cheat CA` by installing it, and by extension we trust anything that `Cheat CA` trust, ie. signs).

3.1.3. Reverse Code-Injecting Proxy

Once we have our own certificate authority up and running, setting up a reverse proxy server is pretty straight forward. And as explained earlier, one of the security implications of the fact that we are running our exam in a browser is the fact that we can just add code to the response as text, and the browser will happily execute it. In theory, I should be able to modify my hostfile (modifying my own DNS responses), and make `inspera.no` point to the loopback address `127.0.0.1`, while running a simple reverse proxy server I wrote in `node.js`. Starting the exam



```
  last-modified: Fri, 28 Apr 2017 07:42:04 GMT
  server: nginx
  strict-transport-security: max-age=7200; includeSubdomains;
  vary: Accept-Encoding
  x-content-type-options: nosniff
  x-xss-protection: 1; mode=block
  content-length: 2641
  connection: keep-alive
GET /file/cil/mp_logo/file (https - undefined)
  host: demo.inspera.no
  connection: keep-alive
  user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36
  accept: image/webp,image/*,*/*;q=0.8
  referer: https://demo.inspera.no/
  accept-encoding: gzip, deflate, sdch, br
  accept-language: en-US,en;q=0.8,no;q=0.6,nb;q=0.4
  cookie: i18next=en_us; _ga=GA1.2.2138523296.1485633115; __utma=133096918.2138523296.1485633115.1486639031.1493956841.4; __utmb=133096918.1.10.1493956841; __utmc=133096918; __utmz=133096918.14861
31919.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); JSESSIONID=AB7500255BD01D8268AB7D844D5446F3C3280A2EBE5E0109CC3A574F036A; serverId=app223; AWSELB=E50F15B706899AF6AF538F074971A28117CC02E129
9D1002C81F4B1D44AE0CA51FF0E62ABCA7B043DC8642630CCB8A4F2C298720BC4202CA4F6BBBC2B95388239588362C182BFD4ECCDBF46D0C1F1A8130941A92C8
ip: 34.253.55.86

Sending request to https://demo.inspera.no/file/cil/mp_logo/file

Response from /file/cil/mp_logo/file: 302 Found
  date: Fri, 05 May 2017 04:28:32 GMT
  location: https://insperalearning-prod.s3.amazonaws.com/demonstration/2013/12/01/logoia_1383890_1385935345414.png?Signature=HLvG1TVOF4OopeF6UYn0o034C0I%3D&AWSAccessKeyId=ASIAIDQACTNJN5G4NBRQ&Exp
ires=1493987312&x-amz-security-token=FQoDYXdzEOX%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaDLAVjZmBf%2FMLQ7s4TyKcA3Z8L1taPn5Q8dyV1fxZJv2%2FKWwZkvi4zRoWTioXavY%2BJJhw9kwEUDLe29rukpFPTUQubYyBPpGnwhrt%2Bnp0mEP
LcHZTkixeUj1K%2BAIVbVHhRFlwqlkyTUSEn2Jq1NbHC7SguulSms23q2M6e9YgtOzR9xBvuvvRqcahYRE0bXqPHxx4SXdMeQ8JsZk62A0LOKjoqAc%2BddIZs%2BlARlcacxrWm%2FdzNcAYljuNSVhTsB5cr16LsgSlWES78Wn%2B1DrevVhXvRQUu0k%2BGkD
gNGSyeNhctkznj16ArZcIxJGYFvjr%2FtPsbKZLvjzoE9JH4JK1lMe82%2FVHHiP039IVA%2B2lSc2XGwh8DWCMbb8HnWZB4Z0xIZ1YliD6xBa%2BlvwpPojLNDiEKIoeZRv7jKTAauHLunFQ1sJhwlRLhM6Oj%2Bn%2BABj7ky1XkoZ6Xkc%2BWFMMIU%2FGOxO
9ipiDXPWZcgPOQvV8NZYnh5J5e%2Bs4ySbJSsdvTnY2d9U%2Bdf2gYs19X3T0oG0LEKC5bct86H6bxhWbhP56F5zYkgAo1l5piwiQ%2BWco7%2B2vyAU%3D
  server: nginx
  set-cookie: serverId=app223; Path=/,AWSELB=E50F15B706899AF6AF538F074971A28117CC02E1299D1002C81F4B1D44AE0CA51FF0E62ABCA7B043DC8642630CCB8A4F2C298720BC4202CA4F6BBBC2B95388239588362C182BFD4ECCDBF46
D0C1F1A8130941A92C8;PATH=/
  strict-transport-security: max-age=3600; includeSubdomains;
  x-content-type-options: nosniff
  x-xss-protection: 1; mode=block
  content-length: 0
  connection: keep-alive
^C
~/hub/cheat % █                                                                                                         6:28  alxandr@Aleksanders-MBP
```

Figure 3.2: Sample output from a reverse proxy server with logging enabled. Shows requests and response headers and status codes.

with this configured should result in me being able to access the exam like normal, while injecting code into the exam website allowing me access to resources I should normally not have access to. See figure Fig. 3.2 for sample output from this reverse proxy server being set up against `demo.inspera.no` simply browsing the web-page in Chrome.

Modifying the requests going back and forth between Safe Exam Browser and Inspera Assessment should allow me to do more or less anything you would be able to do using either a browser or a console application running on your computer, which is basically anything. I could write the server in such a way that when I click a link on the exam website my computer shuts down for instance. Although doing so serves no practical purpose, it illustrates the fact that I'm able to do things to the computer that Safe Exam Browser is attempting to prevent me from doing. Another thing I could do is have it so that when I click a button, the content of a file on my harddrive is written into whatever input field is currently in focus, thus giving me access to local notes on my computer. More worryingly is the fact that I could write scripts that allow me to navigate to any website inside safe exam browser. This basically means that all bets are off with regards to keeping the exam environment "safe", and that I can do anything from searching for answers on Google, communicate with outsiders that could help me on the exam, collaborate with other exam takers, etc. Effectively I have the world at my fingertips, and while Safe Exam Browser is still preventing me from closing down my browser window to use other programs, I have complete control of what happens inside said browser window.

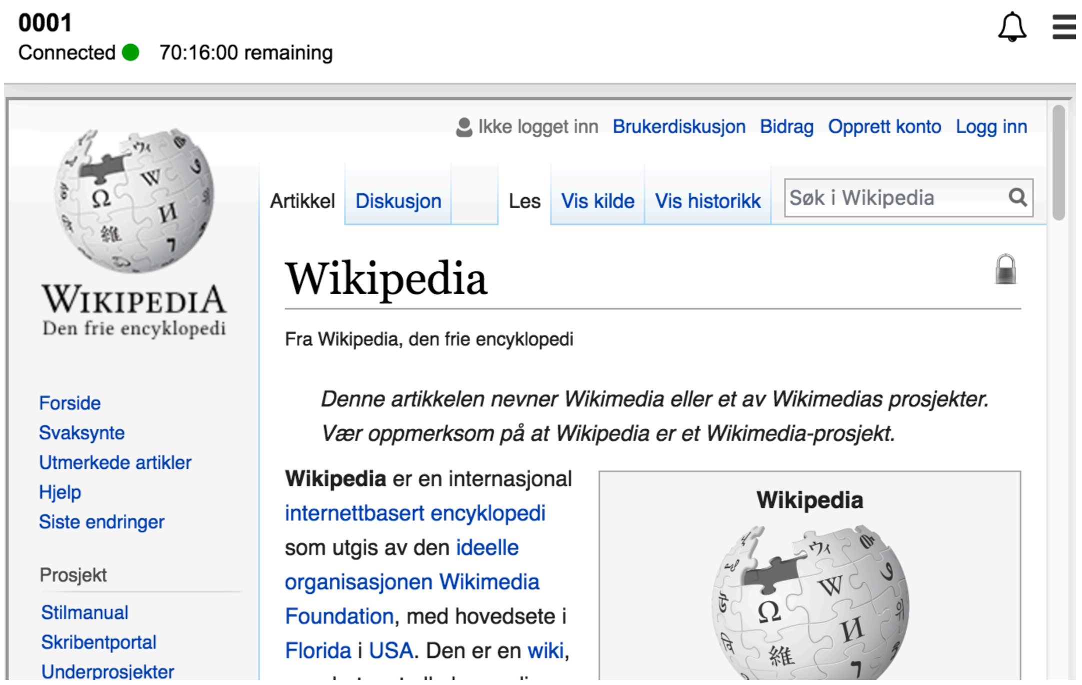
3.1.4. Testing

My initial test of using a reverse code-injecting proxy to take an exam using Safe Exam Browser did not work out as expected. I made several tests using different implementations of the reverse proxy server and using different certificates and certificate management software. But, no matter what I did I could not get Safe Exam Browser to work with my certificate, self-signed or not. Instead what would happen was that when I attempted to start the exam, I would get an error message telling me that a certificate error had been encountered, and at that point I was stuck. Safe Exam Browser does not help you at all in cases where something goes wrong, instead it just leaves a blank window that you're not able to close. Without having the reset code available (which you typically wouldn't) the only way to get out of this state is to forcefully reboot the machine by holding down the power button. This made testing annoying and time-consuming because any tiny change I did required a full reboot of the computer.

After having attempted several ways of getting Safe Exam Browser to accept my custom certificate I eventually tried to use the network monitoring tool Fiddler

[26]. I already knew Fiddler to be working, because I've used it many times when debugging websites, and I wanted to look at the data produced by Safe Exam Browser and the data produced by my reverse proxy and try to figure out what was different, however to my great surprise, I got the exact same error using Fiddler. This lead me to the conclusion that Safe Exam Browser was not using the global store of certificates, but instead had its own. This is also the reason why Firefox (which is traditionally based on XULRunner) needs to be specially configured for using Fiddler [27]. However, since I can't really configure XULRunner without modifying Safe Exam Browser, and modifying Safe Exam Browser would defeat the purpose of the reverse proxy server altogether, therefore the conclusion is that using a reverse proxy server for injecting code into Safe Exam Browser **if** the exam system runs over HTTPS on Windows is not possible. At least not easily possible. For more info on how it can possibly be done without changes to Safe Exam Browser see, Section 3.4.1.

Safe Exam Browser on macOS however, does not use XULRunner (even though XULRunner runs on all platforms). Instead, it uses a browser component based on the WebKit browser engine. In difference to the XULRunner browser engine used by Safe Exam Browser on Windows however, it does not have its own root certificate store. It instead (like most browsers) uses the operating system for dealing with what certificates to trust. This means that while it is not possible to use the reverse proxy server I wrote to modify requests and responses made by Safe Exam Browser on Windows, the same server does work on macOS.



0001
Connected ● 70:16:00 remaining

Ikke logget inn Brukerdiskusjon Bidrag Opprett konto Logg inn

Artikkel Diskusjon Les Vis kilde Vis historikk Søk i Wikipedia

WIKIPEDIA
Den frie encyklopedi

Forside
Svaksynte
Utmerkede artikler
Hjelp
Siste endringer

Prosjekt
Stilmanual
Skribentportal
Underprosjekter

Wikipedia

Fra Wikipedia, den frie encyklopedi

Denne artikkelen nevner Wikimedia eller et av Wikimedias prosjekter.
Vær oppmerksom på at Wikipedia er et Wikimedia-prosjekt.

Wikipedia er en internasjonal internettbasert encyklopedi som utgis av den ideelle organisasjonen Wikimedia Foundation, med hovedsete i Florida i USA. Den er en wiki,

Wikipedia

Figure 3.3: Accessing Wikipedia inside Inspera Assessment running in Safe Exam Browser.

3.1.5. Modifying the LMS

While being able to set up a reverse proxy server that you can use to inspect traffic going from Safe Exam Browser to Inspera Assessment servers and back is certainly helpful when doing other exploits, it is not really an exploit in and of itself. Only by modifying the responses coming from the LMS or by crafting custom requests going there are we able to gain access to restricted resources. If the LMS system has any security exploits like the client deciding whether or not an answer to a question is correct, this would enable us to forge requests going to the LMS just telling it that we answered all the questions correctly. This however, is not the case with Inspera Assessment. Instead, from the data I've seen going between Safe Exam Browser and Inspera Assessment for demo exams that were set up so I could do this testing, it seems that they simply store the answers to the questions, and figuring out whether or not they are correct is either done on the server, or at a later time by a human.

Instead, what I can do is modify the responses coming back from Inspera Assessment before it reaches Safe Exam Browser. This enables me to do all kinds of things that I would not normally be able to do during the exam, like accessing Wikipedia (see Fig. 3.3). More interestingly though is the fact that since you're running your code inside Inspera Assessment it becomes possible to make it look like whatever you're accessing is actually part of Inspera Assessment and not an external resource.

Fig. 3.4 shows a quick mockup of putting Bing search capability inside the exam software. Obviously this does not do a good job of masking the fact that you're using resources you should not be, because Bing with it's glaring colors stands out like a sore thumb, but the fact that I can make a button on the bottom bar (where you go from question to question) that gives you access to the Internet is still quite telling. If I instead of making a search button made a chat button, and designed a chat that used the same fonts and style as Inspera Assessment itself, it would look like just another question page. You can hide what you're doing in plain sight to the point where people would have to read the actual text to figure out that something is wrong.

Another option would be to make things invisible by default, such that if you click a specific set of key combinations the search window pops up, and when you let go it vanishes. This too would make it almost impossible for invigilators to figure out that you are cheating.

3.1.6. What You Can't Do with This Exploit

There are some things that does not work with the code injecting exploit for various reason, some of which are easy to solve and some of which are not. Amongst

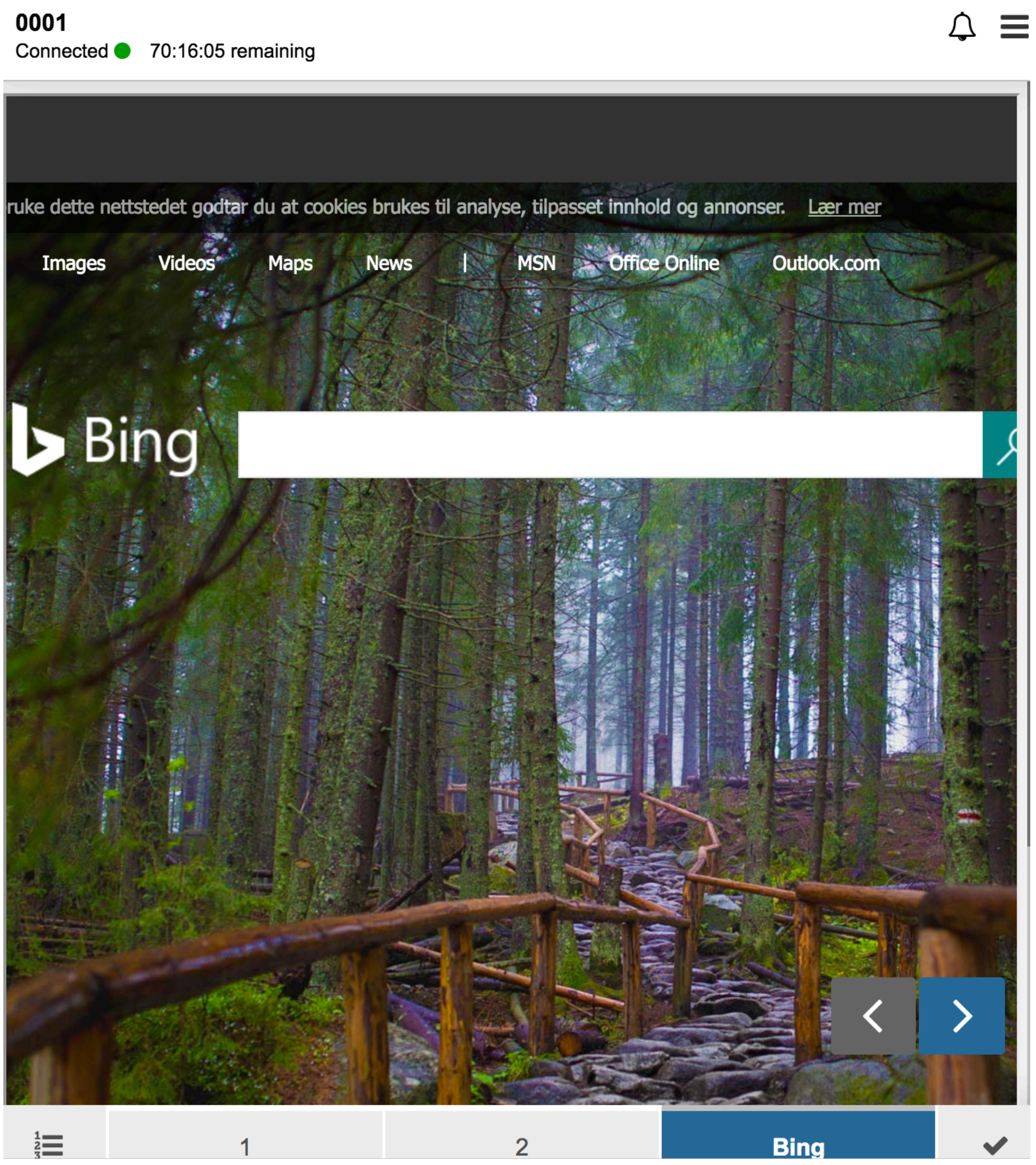

0001
Connected
70:16:05 remaining
ruke dette nettstedet godtar du at cookies brukes til analyse, tilpasset innhold og annonser. Lær mer
Images
Videos
Maps
News
MSN
Office Online
Outlook.com
Bing
1
2
Bing

Figure 3.4: Creating a Bing button inside the Inspera Assessment to make it look like it's part of the exam.

the things that do not work with the current implementation of the reverse proxy server is using Google instead of Bing as the search engine for instance. The reason Google does not work is that they provide a `content-policy` header that forbids the hosting of their websites inside `iframes` (which is how I've done it with Bing and Wikipedia). This is however rather easy to get around, and the solution is the same as what we're already using to inject code into Inspera Assessment, modifying the response from a server we don't own. So enabling Google would be as simple as generating a new certificate, modifying my reverse proxy server to remove a specific server, and making sure request to Google also go through my reverse proxy. The problem with this though is that while doing so for one or domains is not too much work, eventually, you would have to make a system that auto-generates certificates for you (the way fiddle does). Since you don't know which websites use this `content-policy` header, you would have to run everything through your reverse proxy server. For the simple demonstration required in this thesis, it was much easier to just go with Bing that just works as is.

Another thing that this solution does not really help you with is getting access to local resources. There are a few ways around that which generally involves writing (or installing) a local web-server that you can browse to access the files on your harddrive, yet this is cumbersome and would generally only allow you to access plain text files and not documents such as Word or PDF files. This too can be done in a browser, but it would require a lot of work. Instead, it would likely be much easier to just upload your local files to something like Google Docs, and just navigate to Google Docs during the exam. You would have to solve the content policy problem, but that would be much easier than figuring out how to render a PDF or a Word document in the browser.

3.1.7. Applying Outside BYOD

Getting this exploit to work on a computer you do not have full access to is definitely a lot more tricky than one you can modify as you wish. The man in the middle reverse proxy server itself is not hard at all, and can be put on a tiny system-on-chip computer similar to, but much smaller than a raspberry pi. This device could then be connected between the Internet connection of the client used to take the exam and the Internet cable. The problem though is getting the certificate validated. Also, in the case of thin clients that only run something like a remote desktop, unless you have access to install the reverse proxy server inside the remote desktop this exploit is completely mitigated.

3.2. Accessing the LMS Outside the Safe Environment

While being able to modify the code being run inside the safe environment does enable us to do quite a bit already, the fact that we're running in a safe environment puts a lot of restrictions on what we can do on the local machine. Safe Exam Browser (with the settings provided by Inspera Assessment) prevents you from starting programs, opening local files, using your computer for solving math (and other things the computer are quite good at), etc. If we could instead run the exam without having to bother with Safe Exam Browser at all, that would make things much easier. We could simply minimize the window when we wanted to do other things, and have open separate tabs for other websites, such as Google, Wikipedia and Wolfram Alpha.

In order to achieve this however, we need to trick Inspera Assessment into believing that we are using Safe Exam Browser when we in fact are using another browser altogether. There are a few ways we could go about doing this, one of which is to write a browser extension that would modify our requests before they are sent to Inspera Assessment so as to mimic Safe Exam Browser. Another option would be to do some simple modifications to the reverse proxy server I used in Section 3.1.3. Since most of the work were already complete for option two I opted for that.

3.2.1. Identifying Safe Exam Browser

In order to trick Inspera Assessment into believing we're using Safe Exam Browser when we're not, first we need to figure out what it uses to identify Safe Exam Browser in the first place. My contact at Inspera Assessment told me that in addition to the `x-safeexambrowser-requesthash` header which is used by any LMS with support for Safe Exam Browser they also validate the user agent string. Figuring out the user agent is straightforward, I just need to run my reverse proxy server and log it, then start an exam. The request hash is a lot more tricky, because it is generated for each request, and is different for each request. Therefore, I couldn't just look at the log output from my reverse proxy server and replicate it, I needed to figure out how it was validated.

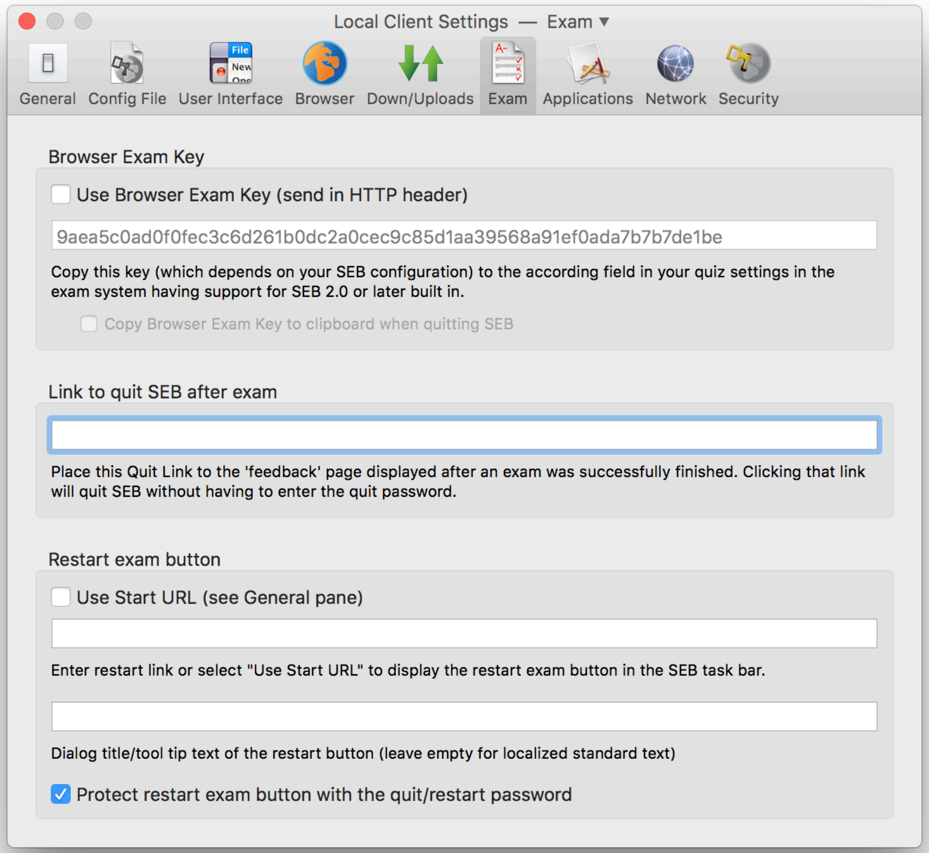
While Inspera Assessment is a closed source program, another LMS with support for Moodle [14] is an open source alternative with support for Safe Exam Browser. The module they use to validate requests from Safe Exam Browser is also open source. From looking at the source code for Moodle I found that a request from Safe Exam Browser is validated by taking a secret key (generated by Safe Exam Browser and configured in the LMS) and append it to the current request URL, and then run the resulting concatenated string through `sha256`.

Given that my requests are going to pass though a reverse proxy server, I already know the URL, and `sha256` is a standard cryptographic function that is available in most programming languages and environments, the problem is figuring out the secret key that Safe Exam Browser generates.

3.2.2. Browser Exam Key

Figuring out the browser exam key (as it is called by Safe Exam Browser) is the harder part of forging requests to make it look like they were made using Safe Exam Browser. It is also the entirety of Safe Exam Browsers security model with regards to preventing people from using other browsers at exams. If the Browser Exam Key could easily be forged, then effectively Safe Exam Browser is just an optional browser when taking digital exams. Therefore, the generation of the Browser Exam Key is rather complicated. It changes if you modify SEB, and it changes if you modify the config file, and it's different on Windows and macOS, so it's not trivial to compute. However, given the fact that the secret needs to be inserted into the LMS for the LMS to be able to validate a request from Safe Exam Browser that indicates that at some point the secure secret has to have been made available for humans to use. Otherwise, how would you get the secret into your LMS?

As it turns out, if you open the Safe Exam Browser config editor tool it will happily just give you the only secret all of its security hinges on. See Fig. 3.5 for example of how the config tool presents the Browser Exam Key to any user who knows how to use it. Not only do you get the secret here, you also get the user agent string (if you didn't grab that using a network logging tool), plus all of the settings used for this exam, like which website to go to start the exam etc. To Safe Exam Browsers' credit, the config file is rather heavily encrypted, and you need to enter the encryption password to be able to open the config file in the config editor, but at the same



Browser Exam Key

☐ Use Browser Exam Key (send in HTTP header)

9aea5c0ad0f0fec3c6d261b0dc2a0cec9c85d1aa39568a91ef0ada7b7b7de1be

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.

☐ Copy Browser Exam Key to clipboard when quitting SEB

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 pane)

Enter restart link or select "Use Start URL" to display the restart exam button in the SEB task bar.

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

✓ Protect restart exam button with the quit/restart password

Figure 3.5: Browser Exam Key computed and ready for copy from Safe Exam Browser config editor.

time the encryption password is handed out when the exam starts because you need the password in order to start the exam. That means you have a short while (depending on where you sit) to open the config editor, grab the Browser Exam Key, start the reverse proxy server, and then start the exam in another browser in fullscreen.

3.2.3. Brute Force the Password

Being able to get the Browser Exam Key from Safe Exam Browser itself is nice and all, but the fact that we have to wait for the password to be published makes it less than optimal. If we could get the Browser Exam Key a bit earlier, that would mean we could set up our computers before the invigilators start doing their rounds, and everything would be ready to go. Yet, as said, Safe Exam Browser uses an encryption that is effectively unbreakable. On my somewhat old laptop I managed to get about 800 key checks per second. If we use alphanumeric passwords of 6 character lengths that would mean it would take me on average a bit over 2 weeks. Considering most exams last about 4 hours that would not be any help at all.

Luckily, the passwords used by Inspera Assessment isn't alphanumeric, nor are they 6 characters long. Instead they are just numeric pins with 4 digits. This reduces the keyspace enormously, allowing me to search through all possible answers on my old laptop in roughly 12-15 minutes. This could easily be accomplished before the exam starts. I also tested the program on a faster laptop with more cores, where the entire process completed in less than 3 minutes. This means that you could very easily show up 15 minutes before the exam starts (just as you normally would anyway), decrypt the config file, and setup either a proxy server (like I've done) or a browser extension (if one was ever made) with the correct Browser Exam Key. Then you could start the exam in Chrome in fullscreen, and nobody looking at your screen would be the wiser.

3.2.4. Applying Outside BYOD

For the most part, using the exploits described here requires the same access as described in Section 3.3.5. It also requires a different browser than Safe Exam Browser to be available on the client. In the case where clients are regular remote desktop thin clients this is not much of a problem, but if the clients have been specifically designed with exams in mind, and they only run the examination software then doing the above would likely prove difficult. However, creating such clients would require a large amount of infrastructure to be in place.

3.3. Modifying Safe Exam Browser

Having looked at how the Browser Exam Key is used to identify Safe Exam Browser, I wanted to look at how it was generated, to see if I could sidestep Safe Exam Browser and not need to use its configuration tool. Since Safe Exam Browser is open source, the code is openly available for anyone to read and make changes to. However, the idea is that if you make changes to Safe Exam Browser, it will generate a different Browser Exam Key which would result in the LMS not validating the request. As we've seen in Section 3.2.1, the request hash is generated per request composed of the URL of the request (which is known), the `sha256` function (also known), and the Browser Exam Key which is the presumably secure part of the equation.

3.3.1. Generating the Browser Exam Key

Figuring out how the Browser Exam Key is generated is actually quite simple. The windows version of Safe Exam Browser is written in C#, which is one of my main programming languages, and is decently written, so tracking down where the Browser Exam Key was generated was not hard at all. The relevant parts is the methods named `ComputeBrowserExamKey` and `ComputeSEBComponentsHash` in the `SEBProtectionController` class. The `ComputeBrowserExamKey` method takes the config file content and concatenates it with a hash of all the "SEB Components" (acquired by calling `ComputeSEBComponentsHash`). It then runs the resulting string through a `HMAC` algorithm using `SHA256` as the hashing function.

Cryptographically this is as I understand very safe. Considering the cryptographic functions in use are all considered safe. The only thing that could potentially be a problem with the cryptography itself is how they are used together, however you'd need an expert in cryptography to figure that out. Regardless of whether or not there exist any exploits usable against the encryption however, does not really matter, because there are no secrets involved in computing the Browser Exam Key. All of the information above is information that we either know or can figure out. The config file is inherently available because we're working on modifying Safe Exam Browser itself, and it obviously already deals with the decrypting of the config file so we don't have to do that on our own. The only thing that's supposed to prevent us from modifying Safe Exam Browser and still be able to use it as if it wasn't modified is the `ComputeSEBComponentsHash` method.

3.3.2. Safe Exam Browser Components

So, what does the `ComputeSEBComponentsHash`method do? Quite simply, it uses the current running application to figure out where it is executing from (where Safe Exam Browser is installed), then it finds a bunch of files that are part of Safe Exam Browser itself and computes a hash of all these files. In general, the idea is fairly straight forward. If you modify any of the files required to run Safe Exam Browser, you will get a different hash value. However, getting around this security constraint is incredibly simple. Considering the fact that the `ComputeSEBComponentsHash` is constant (it will always return the same thing as long as you're running the same version of Safe Exam Browser), if you could figure this value out once you could simply replace the `ComputeSEBComponentsHash` method with a method that returns the expected string. This still leaves you with the issue of figuring out what the result of this method is, and I'll get back to that later.

Another way you can easily make `ComputeSEBComponentsHash` return the correct hash even when you're using a modified version of Safe Exam Browser is simply by replacing the line that gets the current running application directory with a path to an installed correct version of Safe Exam Browser. This is the route I used to make my own version of Safe Exam Browser generate the correct Browser Exam Key.

3.3.3. Disabling Security in Safe Exam Browser

Looking further into Safe Exam Browser, most of the configuration keys are available at the `SEBSettings` class. Examples such as `KeyEnableAltTab` and `KeyEnablePrintScreen` corresponds to settings that can be toggled on and off. Doing a quick search through the codebase for where the different keys are used, it's easy to disable almost all security provided by Safe Exam Browser. With only a few lines of code changed I was able to run a custom Safe Exam Browser that connected to the exam as if it was the original Safe Exam Browser while allowing me to alt-tab to other windows while I was doing the exam, and copy text in and out of Safe Exam Browser.

3.3.4. Linking Against Safe Exam Browser

Given that Safe Exam Browser for Windows is written as a .NET application in C#, I wanted to see if I could write another application that linked directly against the installed Safe Exam Browser, and it turns out it's perfectly doable. This is what allowed me to write my brute force program in Section 3.2.3. Simply open up Visual Studio, create a new C# project, and add the installed Safe Exam Browser as a dependency of your project. You can then call public methods in Safe Exam Browser (like `ComputeBrowserExamKey`), or using reflection you could also call private methods (like `ComputeSEBComponentsHash`).

3.3.5. Applying Outside BYOD

Depending on the client used by the university to perform exams, the exploits relating to modifying SEB or creating programs that extract information out of the SEB config file is likely to still be usable. The application I wrote to brute force the config file (see Section 3.2.3) does not require installing, nor administrator privileges to run. However, it does require that you have access to a Windows desktop environment and are able to run arbitrary applications (prior to exam start).

3.4. Other Potential Exploits

There are other potential problems with how Safe Exam Browser is built on Windows. I have not looked at how any of this works on macOS. These have not been categorized as their own exploits considering the fact that I've already managed to both run Inspera Assessment without using Safe Exam Browser, and using a custom version of Safe Exam Browser. However, there is one more inner working of Safe Exam Browser that is worth taking a short look at, and that's how it integrates with XULRunner.

3.4.1. Late Updates to XULRunner

When you launch Safe Exam Browser in its normal "browser" mode, what happens is that it reads the config file, then based on the settings set therein it registers some key handlers and modifies some register values, and does other things to effectively take control over your computer and limit you in what you can do. Then it takes a bunch of the settings and a generated Browser Exam Key and writes them to a file. It then launches XULRunner which reads this file and runs as the "browser component" of Safe Exam Browser (this is what the user sees). The browser integration is done through an addon written in javascript.

It stands to reason that a simple application could watch this file that Safe Exam Browser writes and copy its content before Safe Exam Browser deletes it again. This would be another way to get the Browser Exam Key automatically. You could then exit Safe Exam Browser before starting the exam (Inspera Assessment allows you to do this). Another thing to note is that once the XULRunner config file has been written, at that point the Browser Exam Key has already been generated and written to disk, so it would be possible to make changes to XULRunner at that time. This is unlikely to gain you much more than any of the other exploits explained in this thesis, but it is still an exploit.

4

Proposed solutions

In this chapter I will discuss some proposed solutions to the problems discovered with the digital exam solution employed today by NTNU. These solutions assume that the framework is still that students should bring their own computers.

4.1. Technical vs. Human Solution

The digital exam problem is not a purely technical problem, yet it's easy to look at it as one. If we could discover a purely technical solution that prevented any sort of digital cheating, that would mean that digital exams are *at least* as safe as pen-and-paper exams. However, a purely technical solution that prevents any sort of digital cheating is unlikely to exist, as such we cannot look at the digital exam problem in a bubble. For the most part, until now, what I've discussed has been purely technical solutions and problems. I've stated some statements about how something might help keep the fact that you're cheating hidden, but all in all, I've focused entirely on defeating the computer, not the invigilators.

As stated in Section 3.1.5 some uses of the exploits discussed in Chapter 3 is like asking to be discovered. If you for some reason for instance use Chrome to access the exam, and then at the same time go to YouTube to look at lectures on the subject while playing audio through the speakers you would easily be discovered as cheating. This is obviously an incredibly unlikely scenario though, as someone who manages to figure out how to get the Browser Exam Key out of Safe Exam Browser and into either a chrome plugin or a reverse proxy server is unlikely to make the foolish mistake of playing videos on full blast during the exam. Rather you would want to take any step possible to prevent others from finding out that you have cheated. For instance, a clever student could design a chat application that looks like Inspera Assessment.

4.1.1. Identifying Cheating Using Invigilators

One of the goals of having invigilators at exams is to help prevent cheating. They make sure that people didn't bring equipment they weren't allowed to, and that they do not communicate with one another. It stands to reason that invigilators could also be used to prevent digital cheating. The problem with relying on invigilators spotting digital cheating though is the fact that there may not be any audible or visual cues that the computer is doing anything. At the same time, it's also possible to make the computer put on a show by displaying on screen actions it's performing, while actually not doing anything.

For instance, if you know what you're doing it should not be all too hard to pretend to reboot a computer - while in fact not doing so. This can be done using a virtual machine that you actually reboot, or simply playing a video in fullscreen. Even, should it be decided that the exam software is to be put on a memory stick and students have to boot from it, once someone figures out what's contained on the memory stick it would be possible to write software that extracts any needed secrets from it and then again puts on a show of booting from the memory stick. And this is all still just in software.

If you were to look at specialized hardware, there is almost no limit to what people would be able to do. For instance, imagine someone builds a laptop with two computers inside it, and the ability to toggle between which one is connected to screen, keyboard and mice. This is not entirely far-fetched, as there already exists cheap tablet-like units that work like that. However, custom hardware is also a problem in pen-and-paper exams in that students can bring specialized hidden cameras and earpieces for instance, to communicate with people outside the exam. It also requires a rather substantial monetary investment.

4.1.2. Cheating Software

The problem with software that can be used to cheat is the fact that it can be
The problem with software that can be used to cheat is the fact that it can be
replicated indefinitely with next to no cost. Distributing software is fast, and there is no limit to a number of copies that can be made. It's also effortless for the person receiving the copy. If for instance, I made a custom version of Safe Exam Browser that had no security implemented but still produced the correct Browser Exam Key and uploaded it, students would just have to download my version of Safe Exam

Browser instead of the official one.

Because of these reasons what we need is a near-perfect technical solution, where invigilators can for the most part just fulfill the role they have always fulfilled. At the same time, we want to reduce the amount of resources one can get access to using cheating to be as low as possible. With the mindset that a perfect solution is not feasible, there are steps that can be taken to make cheating much harder, and at the same time also try to completely limit the access students has to online resources they should not be able to have access to unless they have custom specialized hardware.

4.2. Problem with Todays Solution

Before we can look at possible solutions to the digital exam problem, we have to take a closer look at the core of the problem, namely the fact that we have a server (Inspera Assessment) trying to validate what client software (Safe Exam Browser) is being used. The only way for two parties to have any form of validation (one or two way) is by way of either a shared secret, or public key cryptography. In either case, the party that is to be validated need to hold some form of a secret that is either transmitted as is, or some product of it is transmitted to the other party. In our case, that means that for Inspera Assessment to validate that we are indeed using Safe Exam Browser some secret needs to be transmitted to Inspera Assessment. This is effectively what happens with the Browser Exam Key.

The issue however is that we've (by installing Safe Exam Browser) effectively taken the secret that we use to validate the users, and given it to all the students. There is no way around this. No matter how well you hide the secret, at some point in time the secret needs to be available for Safe Exam Browser to send to Inspera Assessment, and as such at that point it will be available to the computer it is installed. Now, there are definitely ways it could be made to be harder to get hold of, but at the end of the day it is accessible if people have the knowhow to do so. And it only needs to be figured out once. If the person who figured it out spreads it, that's it. Game over. You have to start all over with a new way of hiding it. Effectively, any such measure is just security theater, or security by obscurity. Eventually someone will come along with the skills to find it, and it might as well not have been hidden in the first case.

What's usually used to validate clients is public key cryptography (typically certificates like you'd find on a server). However, this is used to validate a client, not a client application. It's to validate that I'm me, in other words there is no problem if I know the secret. The same techniques cannot be used to validate that the right client application is used.

4.3. Looking to Others

NTNU is not the only, nor the first university to attempt to use digital exams, and other universities across the world have done so using different solutions. The Lebanese French University in Kurdistan developed their own solution using normal web technologies and a custom browser implementation written in C# without the normal close/minimize/maximize buttons normally shown on windows [20]. They claim that by having randomized questions for each student (delivered from a question database) they can effectively reduce cheating to almost zero, but do not provide much data to back this claim up. While it's true that having random questions would prevent the typical "look at another screen" kind of cheating, it does nothing to mitigate users accessing resources they should not have access to, online or not. A paper on digital exams at the University of Southern Denmark (SDU) in Denmark [17] notes another potential advantage of having a digital question database in that such a system can provide statistics over the different questions, such that questions that everyone answers correctly or nobody answers correctly can be removed. One of the techniques SDU uses against cheating is anti-plagiarism software. They also let the students know beforehand that they are using such software, such that students are less likely to cheat because they know it's more likely that they will get caught.

In the paper "Challenges of Online Exam, Performances and problems for Online University Exam" they propose using biometric sensors to authenticate users as a means to prevent cheating. It also proposes having cameras that can view the entire lecture hall [21]. The biometric sensor might certainly help prevent students getting other people to take the exam for them, if that is indeed a problem, yet we already require student ID to be displayed before starting an exam. While having a camera in the exam room might prove helpful, it's unlikely to be much more helpful than having invigilators, and it does provide some privacy issues considering the fact that the footage may be recorded.

Swiss Federal Institute of Technology (ETH) Zurich has a rather interesting approach to preventing cheating at digital exams. While normally they would just go with a similar setup that NTNU is currently using, on some exams they would like the students to have access to other applications than a browser, such as CAD software or similar. Instead of running the LMS directly inside Safe Exam Browser, they run a remote desktop client which connect to a server they control and also runs Safe Exam Browser which then connects to the LMS and allows access to selected client applications on said server [6].

ETH Zurich also takes snapshots of the screen at regular intervals. However, these snapshots are never looked at unless requested by the student themselves. The students are assured that the snapshots will not be used to find cheaters, and are only there in case something goes wrong during the exam, so that most of the work can be recovered (in the form of images). This is a rather good failsafe solution as long as there are strict guarantees that it will not be used for surveillance.

4.4. Solvable Problems

It's important to take a step back and look at the overall problem we're trying to solve here. It's very easy to start asking questions such as "how can we make sure students are using Safe Exam Browser when connecting to Inspera Assessment?"(or any other LMS for that matter), but that's not really the issue at hand here. Another question to ask is "how do we prevent code injection into Safe Exam Browser?". This is actually a rather easy problem to solve as Safe Exam Browser supports specifying which certificates it should support. I have not been able to test this functionality because it would require me to modify the Safe Exam Browser config file which would result in a different Browser Exam Key being generated, but according to documentation it should work. But again, this is not really the right question to ask.

It's very easy to be blinded by what we have now, and only look for solutions to the immediately discoverable issues, and try to patch things to work as intended. That is not to say that we need to replace/rewrite everything, far from it, but the questions needs to be framed with the actual goal in mind, not what we think is the solution.

The original goal of the digital exam solution we have at NTNU is to facilitate digital exams on students own laptops, in which it should be hard to cheat. Preferably as hard as it is to cheat at a pen-and-paper exam. And in this case, by cheat we mean gain access to resources that has been deemed illegal to access during the exam.

4.4.1. Preventing Access to Online Resources

The most damaging resource students can get access to during exams, and the easiest resource to block is arguably the internet. By simply disconnecting the students from the internet a whole slew of resources would be unavailable for the students. They would be unable to search for answers online, they would be unable to communicate with eachother, or outsiders etc. As noted by Thea, doing so would prevent several classes of exploits [24, pp. 67].

The problem is that Inspera Assessment runs over the internet, so simply cutting internet access is not really a solution. Also, cutting internet access completely might actually make it easier to get around the restriction, than having restricted internet access. My suggestion is using a specialized router for the exams which has been set up in such a way that it can only access a whitelist of IP addresses (such as the LMS website). This way, it would be easy to say that students are allowed to access online resources that are needed for the exam, but students will not be allowed to access anything else.

The reason why this is better than simply cutting internet access and running everything locally is the fact that most people nowdays has smartphones that can be used as wireless access points. But not many people have computers capable of connecting to two different wireless networks at the same time. The other part of the puzzle is that by forcing all the students to connect through a single router, all of them can be given the same public IP address. This would mean that we can validate on the server that the students are in fact connecting through the restricted network (instead of just using the cell phones as a router).

Next, the LMS needs to be designed in such a way that it can notice when connection to the client is lost (this should be fairly easy to do). After the LMS notices that a client has disconnected for more than a few seconds, it should notify the invigilators and they can go check if the student is doing anything he shouldn't, or if he simply lost his internet access. While this may mean that a student with a bad wireless card may get visited by the invigilators a few times during an exam, this should be a rather small price to pay.

In terms of hardware/setup, this is also not a very costly solution. It's much cheaper than providing computers for each of the students during the exam. At the same time, it increases the difficulty of accessing online resources without being caught to the stage of needing custom hardware to do so (for instance a computer with two wireless network cards).

4.4.2. Preventing Access to Local Resources

It's likely not doable in any good way to prevent access to local resources. In all likelihood, Safe Exam Browser already does as good a job as can be done. You could arguably change Safe Exam Browser to some bootable examination system given to the students on a memory stick during the exam, but as explained in Section 4.1.1 at the end of the day it's a security theater. At some point students will figure out how to get around it, and it'll be impossible to create a new system for every exam. A student could for instance very easily copy the content of the flash drive before booting from it, in which case he would have ample time to analyze the content later and figure out how to work around it.

Invigilators would probably be able to spot if someone does something stupid, like open an easily recognizable application like Microsoft Word during the exam. Considering, however, that the student took the effort to figure out how to get access to local resources during the exam, it's unlikely they didn't also take the time to figure out what local resources they could access without raising any flags.

4.4.3. Anti-Plagiarism Software

The University of Southern Denmark has been successful in using anti-plagiarism software in order to catch students who have cheated on digital exams [17]. Anti-plagiarism software can be a useful and automated tool to flag potentially cheating students, after which a manual investigation can be started. This is most useful however, in exams where students are allowed access to learning resources, online or not. The software might also be able to flag if two or more students are collaborating in some way during the exam. The paper on digital assessments at the University of Southern Denmark also states that the anti-plagiarism software has a preventative effect on cheating in that the students know that there is a higher chance of getting caught. No data is provided to support this claim however.

4.5. Crossing from Analogue to Digital

One of the things exams today suffer from (either they are pen-and-paper or digital) is the fact that the world has changed, and is still changing. Yet the exams remain mostly the same, even if we put them on computers. The way we work over the years has changed greatly. Students of physics today write python programs to visualize their findings, students of cybernetics write MATLAB to run complex simulations, and of cause students of computer science write programs that do anything from compile other programs to run rudimentary artificial intelligence. When writing these, the Internet is used to lookup large amount of related research to use as references.

Exams are supposed to test your knowledge and readiness to enter the working force, but they do so in a bubble constrained by the fact that they are done on pen-and-paper. No self-respecting programmer would ever implement larger parts of their program using pen-and-paper, and if they did they likely wouldn't care if they got the names of a few framework methods wrong, because when the program is to be transferred to a computer, the computer will help you fix those trivial errors. No mathematician would ever constrain himself to only using pen-and-paper and a simple calculator when he needs to do several hundreds of thousands of calculations.

This is not to say that pen-and-paper is useless in any way. There are many things that are still way easier to do using pen-and-paper. Things like writing math equations, drawing, sketching outlines of system architecture etc. Computers and pen-and-paper have different strengths and weaknesses, and simply taking the exams we have today and slapping them on a computer is not necessarily serving anyone.

4.5.1. Tasks Suitable for the Platform

One of the things that become possible when doing digital exams is actually using the incredible resources that are at our fingertips. This is also in a lot of cases much closer to what people will experience when they enter the workforce. Digital exams offer a unique opportunity in that they can be used to conduct tests that are completely different from what can be achieved using pen-and-paper examinations. And in several cases, if the tasks are constructed right, they will of themselves completely remove most ways of cheating.

For instance, an in industrial design or architecture could have students making CAD drawings during the exam. If the system is set up correctly so that files on the students own computer cannot be transferred (even if they can be accessed) to the LMS then having access to local files or the Internet will be of limited help. If we design exams in such a way that having access to the Internet or local files does not provide any help, then there is no point in limiting them. That being said, regardless, what parts of the Internet can be accessed should probably be limited, because otherwise people could communicate with eachother and collaborate on the exam.

Switching from pen-and-paper exams to digital exams is a change in paradigm, and it's likely that in order for the change to be effective in any way or form (except in just saving paper) a paradigm change is also required in the tasks students are given during an exam. And while it's true that this will likely require a large amount of work in creating new tasks, and figuring out how to enable them during the exams, so would figuring out and setting up an exam environment in which cheating is virtually impossible too. If the selling point of digital exams is the fact that they are digital, then there are little point in them.

5 Conclusion and Future Work

In this chapter I draw my conclusion on this thesis. I will answer the original research questions as well as comment on some of the issues and solutions I've described in earlier chapters. I will also outline some potential future work.

5.1. Conclusion

In a society where more and more people are getting higher education, and at the same time more and more of our daily tasks are performed digitally, exams will likely have to adapt in order to stay relevant. While teaching methods have already started to technology in order to teach students the craft they need to know, given the fact that students are primarily graded based on their exam results this is also where many of them will put forth most of their efforts.

When people want to take a drivers license, they are tasked with driving trough traffic with an examiner, not write an essay on how engines work. Yet when a student wants to certify that he knows how to make webpages, he is asked to draw by hand how some code would look on screen. In especially bad cases this might lead to students focusing all of their effort towards the exam, and ending up with a certification that they possess a certain skill, while in reality they only know the theory behind it. School is supposed to prepare students for the work-life, yet if there is a large disconnect between the two that is obviously a problem.

Some of the reasons high-stakes exams are lagging behind when it comes to digitalization is the very fact that they are high-stakes. As such we cannot afford to make errors. In some cases, the outcome of a few exams can literally change the entire life of a student. Therefore changing the formula that has been tried and tested for generations is obviously somewhat scary, and has to be done with great care. And considering the breakneck evolution speed technology has these days it is not at all weird that high-stakes exams are lagging behind. Yet evolve it still must, or be faced with the possibility of becoming entirely irrelevant. If an exam grades you on nothing except that particular exam itself, then the exam has lost all purpose.

For the reasons mentioned above, it is both important that exams can be done digitally and that such digital exams are safe and resilient to students cheating. A goal with the digital exams has been to make sure that they are at least as hard to cheat in as their pen and paper counterparts. While this is definitely a tall order, striving towards that goal is definitely not a wasted effort.

RQ1 What are technical cheating vulnerabilities of the digital exam solution employed by NTNU?

The current digital exam solution employed by NTNU has several security issues outlined in both this thesis here, and previous papers looking into Safe Exam Browser. One of the bigger issues is the fact that another person outside the exam facilities might be taking the exam for you while you're only present at the actual exam to make it seem like you're doing it yourself. This has even recently been in Norwegian news [2]. Most of these issues can be and are being fixed however. While it's true that there will always be some way for students to cheat, updates to Safe Exam Browser is making it harder every update. In my conversations with Inspera Assessment, I was told that planned updates for SEB may very well prevent several of the exploits I discovered in this thesis.

It's quite obvious that there are issues in the current digital exam solution used by NTNU. There have been multiple issues discovered beforehand in other papers [23][24], and there are several discovered in this thesis. As it stands, currently what's stopping students from cheating at digital exams are mostly themselves. This is definitely not ideal and there are several large security concerns that need to be addressed as quickly as possible.

RQ2 How can the ability for students to cheat during digital exam solutions be reduced sufficiently so that digital exams are no more vulnerable to cheating than their pen-and-paper counterpart?

Several of the solutions proposed in Chapter 4 would go a long way in solving the worst issues. Particularly restricting the access to the internet from the students' computer would prevent any form of outside helping. If done right, it could also eliminate the possibility of anyone outside the exam facility doing the exam on behalf of a student.

Another option is to use thin clients owned by the university. As discussed in Section 1.1.1 this has both benefits and disadvantages. However, neither has been thoroughly analyzed in this paper, and thin clients might have other really bad security vulnerabilities that have not been considered. In all likelihood, there is no silver bullet. With exception of some low hanging fruits such as limiting internet access during the exams, improving security at digital exams will likely be an iterative process which gets better and better over time. This is not a process that NTNU is doing alone, there are large amounts of universities and other educational institutions in the world trying to make safe digital exams, several of which have been referenced in this thesis. All of them have resources allocated to improving the situation.

5.2. Future Work

As described out in my conclusion, digital exams are important and so is making them secure. This is not work that can be done overnight, and requires both time and resources. Both Safe Exam Browser and Inspera Assessment is being developed actively, and over time what exploits works against them will likely change. It's entirely possible that in closing one security vulnerability, another one opens that nobody discovered, therefore it's important to keep testing future versions of the software.

Another research inquiry that Inspera Assessment is looking into is applying AI techniques like big data classification to attempt to automatically detect when a student is engaged in suspicion activity. In the beginning, this is likely to cover simple things like if a user copies large amounts of text into the exam during a short time-frame that will be seen as suspicious, and might need further investigation. AIs have the ability to discover patterns and correlations that are completely invisible to humans, and if such a system could be created it could potentially lead to digital exams being more cheating resilient than traditional pen and paper exams.

Bibliography

[1] Benjamin Smedberg. *XULRunner future and ownership - Google Groups*. URL: https://groups.google.com/forum/?%7B%5C_%7Descaped%7B%5C_%7Dfragment%7B%5C_%7D=msg/mozilla.dev.platform/%7B%5C_%7DrFMunG2Bgw/C-4PcHj9IgAJ%7B%5C#%7D!msg/mozilla.dev.platform/%7B%5C_%7DrFMunG2Bgw/C-4PcHj9IgAJ (visited on 02/12/2017).

[2] Jon Bolstad and Simen Sundfjord. *Didrik avslørte at andre kan sitje heime og ta eksamenen hans - NRK Hordaland - Lokale nyheter, TV og radio*. May 2017. URL: https://www.nrk.no/hordaland/didrik-avslorte-at-andre-kan-sitje-heime-og-ta-eksamenen-hans-1.13528215.

[3] James Conrad. "Seeking help: the important role of ethical hackers". In: *Network Security* 2012.8 (2012), pp. 5–8. ISSN: 13534858. DOI: 10.1016/S1353-4858(12)70071-5. URL: http://www.sciencedirect.com/science/article/pii/S1353485812700715.

[4] Quynh Dang. *Recommendation for applications using approved hash algorithms*. Tech. rep. Gaithersburg, MD: National Institute of Standards and Technology, 2012. DOI: 10.6028/NIST.SP.800-107r1. URL: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-107r1.pdf.

[5] Therese C Grijalva, Joe Kerkvliet, and Clifford Nowell. “Academic Honesty and Online Courses”. In: *College Student Journal* (2006).

[6] Tobias Halbherr et al. “Making Examinations more Valid, Meaningful and Motivating: The Online Exams Service at ETH Zurich”. In: *Eunis Journal of Higher Education IT* 1 (2014), p. 14. ISSN: 2409-1340. DOI: 10.13140/2.1.4635.1044. URL: http://www.eunis.org/erai/2014-1/%7B%5C%%7D5Cnhttp://www.eunis.org/eunis2014/papers/%7B%5C%%7D5Cnhttp://www.eunis.org/download/2014/papers/eunis2014%7B%5C_%7Dsubmission%7B%5C_%7D69.pdf.

[7] David N. Harpp and James J. Hogan. “Crime in the classroom: Detection and prevention of cheating on multiple-choice exams”. In: *Journal of Chemical Education* 70.4 (1993), p. 306. ISSN: 0021-9584. DOI: 10.1021/ed070p306. URL: http://pubs.acs.org/doi/abs/10.1021/ed070p306.

[8] Mathew Hillier. “e-Exams with student owned devices : Student voices”. In: *Conference: International Mobile Learning Festival, At Hong Kong SAR China* December (2015), pp. 582–608.

[9] Mathew Hillier and Andrew Fluck. “Arguing again for e-exams in high stakes examinations”. In: *2013 Australian Society for Computers in Learning and Tertiary Education Conference* (2013), pp. 385–396. URL: http://ecite.utas.edu.au/87910/2/87910%20-%20Arguing%20again%20for%20e-exams%20in%20high%20stakes%20examinations.pdf%7B%5C%%7D5Cnhttp://transformingexams.com/files/hillier%7B%5C_%7Dfluck%7B%5C_%7D2013%7B%5C_%7Dascilite%7B%5C_%7Dfullpaper.pdf%7B%5C%%7D5Cnhttp://www.ascilite.org.au/conferences/sydney13/program/papers/Hillier.ph.

[10] Mark M Lanier. “Academic Integrity and Distance Learning”. In: *Journal of Criminal Justice Education* 17.2 (Oct. 2006), pp. 244–261. ISSN: 1051-1253. DOI: 10.

1080/10511250600866166. URL: http://dx.doi.org/10.1080/10511250600866166.

[11] Donald L McCabe. “Cheating among college and university students : A North American perspective”. In: *International Journal for Educational Integrity* 1.1 (2005), pp. 10–11. ISSN: 1833-2595. URL: http://ojs.ml.unisa.edu.au/index.php/IJEI/article/view/14.

[12] Donald L. McCabe, Linda Klebe Trevino, and Kenneth D. Butterfield. “Academic integrity as an institutional issue”. In: *Ethics & Behavior* 11.3 (2001), pp. 249–259. ISSN: 10508422. DOI: 10.1207/S15327019EB1103.

[13] Miraceti. *User:Miraceti - Wikimedia Commons*. URL: https://commons.wikimedia.org/wiki/User:Miraceti (visited on 05/12/2017).

[14] Moodle. *Moodle - Open-source learning platform | Moodle.org*. URL: https://moodle.org/ (visited on 05/07/2017).

[15] Mozilla. *XULRunner - Archive of obsolete content | MDN*. URL: https://developer.mozilla.org/en-US/docs/Archive/Mozilla/XULRunner (visited on 02/12/2017).

[16] Mozilla. *XULRunner deprecation notice*. URL: http://ftp.mozilla.org/pub/xulrunner/nightly/latest-mozilla-aurora/Deprecation%7B%5C_%7Dnotice.txt (visited on 02/12/2017).

[17] Kurt Gammelgaard Nielsen et al. “Evaluation of Digital Assessment”. In: *EUNIS konference 2014 European University Association*. 2014.

[18] Node.js. *Node.js*. URL: https://nodejs.org/en/ (visited on 02/13/2017).

[19] C. C. Palmer. “Ethical hacking”. In: *IBM Systems Journal* 40.3 (2001), pp. 769–780. ISSN: 0018-8670. DOI: 10.1147/sj.403.0769. URL: http://ieeexplore.ieee.org/document/5386933/.

[20] Mohammed Salim and Mazin S Al-hakeem. "Developing a New e-Exam Platform to Enhance the University Academic Examinations : the Case of Lebanese French University Developing a New e-Exam Platform to Enhance the University Academic Examinations : the Case of Lebanese French University". In: *International Journal of Modern Education and Computer Science* 5.5 (2017), pp. 9–16. DOI: 10.5815/ijmecs.2017.05.02.

[21] Mohammad Sarrayrih and Mohammed Ilyas. "Challenges of Online Exam, Performances and problems for Online University Exam". In: *International Journal of Computer Science* 10.1 (2013), pp. 439–443.

[22] Guttorm (IDI/NTNU) Sindre and Aparna (IDI/NTNU) Vegendla. *E-exams and exam process improvement*. Tech. rep. 2015. URL: http://ojs.bibsys.no/index.php/NIK/article/view/258/221.

[23] Thea Marie Søgaard. "Cheating Threats in Digital BYOD Exams: A Preliminary Investigation". Project thesis. Norwegian University of Science and Technology, 2015.

[24] Thea Marie Søgaard. "Mitigation of Cheating Threats in Digital BYOD exams". Master thesis. Norwegian University of Science and Technology, 2016.

[25] Donna Stuber-McEwen, Phillip Wiseley, and Susan Hoggatt. "Point, Click, and Cheat: Frequency and Type of Academic Dishonesty in the Virtual Classroom". In: *Online Journal of Distance Learning Administration* 12.3 (2009), pp. 1–9. ISSN: 15563847. URL: http://www.westga.edu/%7B~%7Ddistance/ojdla/fall123/stuber123.html.

[26] Telerik. *Fiddler - Free Web Debugging Proxy - Telerik*. URL: http://www.telerik.com/fiddler (visited on 05/06/2017).

[27] Telerik and Eric Lawrence. *Configuring Firefox for Fiddler*. URL: http://www.telerik.com/blogs/configuring-firefox-for-fiddler (visited on 05/06/2017).

[28] George R. Watson and James Sottile. “Cheating in the Digital Age: Do Students Cheat More in Online Courses?” In: *Online Journal of Distance Learning Administration* 13.1 (2010), pp. 798–803. ISSN: 15563847. URL: http://proxy.lib.odu.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true%7B%5C&%7Ddb=eric%7B%5C&%7DAN=EJ877536%7B%5C&%7Dsite=ehost-live%7B%5C&%7Dscope=site%7B%5C%%7D5Cnhttp://www.westga.edu/%7B~%7Ddistance/ojdla/spring131/watson131.html.