Abstract
Technology keeps on developing with very fast rates nowadays, behind every piece of it there is software. People use it in their everyday lives for entertainment, work and a very big variety of other services. That is why software needs to work flawlessly and the way to ensure this is by testing it. This report provides a close look into the work and thought process of the whole Testing Software for Trustworthiness project. It starts by a detailed explanation of what is the project is about followed by the approach taken to solving the problem being faced, namely how software can be tested for trustworthiness. Deep research into the area had to be made in the beginning. It includes finding information about other products that already exist and finding a way to compete with them which is done by shifting the focus of the project to create an application that is educational. This was then followed by more specific and technical exploration of tools and methods that can be used for testing software. The whole research is covered in detail in this report. The next part of the report explains the thought process behind pointing the requirements for the project in the first place and then specifying some of the most important functionalities of the application that make it different from other existing tools. The technologies used to create the final Test and Learn program and the working process are shown in the Implementation section of the report. Finally, testing software also needs to be tested. The process of doing it is described at the end of this paper followed by the problems faced during the development of the project and the future work that can be done.
Acknowledgments

This project would not have been completed without my supervisor Dr. Daniel Dresner. His guidance helped me find solutions to a lot of problems that I faced while working on this project. The meetings I had with him were crucial to determining most of the functionality of the application I produced. He also helped me a lot to plan my work better and pushed me when I was behind schedule.

I would also like to thank my second marker Jock McNaught for his feedback on my two presentations which was very helpful in the development afterwards.

Last but not least, I would like to thank my friends who were always willing to give any advice on my project and tell me what they think of it honestly so that I can improve it if I needed to.
Table of Contents

Abstract ................................................................................................................................................. 2

Acknowledgments ................................................................................................................................. 3

1. Introduction ......................................................................................................................................... 7
   1.1 Project Overview ............................................................................................................................. 7
       1.1.1 What is Software Trustworthiness ......................................................................................... 7
   1.2 Project Approach ............................................................................................................................ 8
   1.3 What else has been done in the area ............................................................................................. 9
       1.3.1 Acunetix Web Vulnerability Scanner ..................................................................................... 9
       1.3.2 Sqlmap ................................................................................................................................... 12
   1.4 What the final application and why is it needed ........................................................................... 12

2. Development ......................................................................................................................................... 13
   2.1 Research ......................................................................................................................................... 13
       2.1.1 SQL Injection (SQLi) .............................................................................................................. 13
       2.1.2 Cross Site Scripting (XSS) .................................................................................................... 14
   2.2 Requirements .................................................................................................................................. 15
       2.2.1 Functional Requirements ....................................................................................................... 15
       2.2.2 Non-Functional Requirements ............................................................................................... 16
   2.2 Design ............................................................................................................................................. 16
       2.2.1 User Interface ............................................................................................................................ 17
   2.3 Logic behind the functionality of the application ........................................................................... 20
       2.3.1 Separating the test for SQL Injection and Cross Site Scripting ............................................. 20
       2.3.2 Manual instructions for running each test .............................................................................. 20
       2.3.3 the “How to Prevent” Buttons ............................................................................................... 20

3. Implementation ..................................................................................................................................... 21
   3.1 Technologies used .......................................................................................................................... 21
       3.1.1 Back End ................................................................................................................................. 21
       3.1.2 Graphical User Interface (GUI) ............................................................................................... 22
       3.1.3 Test running .............................................................................................................................. 23
   3.2 Working Process ............................................................................................................................. 23

4. Testing and Evaluation ......................................................................................................................... 24

5. Conclusion ............................................................................................................................................ 26
   5.1 Problems Faced ............................................................................................................................... 26
5.2 Future Work ................................................................................................................................. 26
5.3 Reflection .................................................................................................................................. 26
References ......................................................................................................................................... 26
# Table of Figures

**Figure 1:** The five attributes of trustworthiness \[^{3}\] ............................................................................................... 8

**Figure 2:** Bowtie Approach ........................................................................................................................................ 9

**Figure 3:** Acunetix WVS scan results (video screenshot) \[^{5}\] .................................................................................. 10

**Figure 4:** Acunetix Login Sequence Recorder (video screenshot) \[^{5}\] ................................................................. 11

**Figure 5:** Sqlmap demonstration (video screenshot) \[^{7}\] .................................................................................. 12

**Figure 6:** Search field input displayed on the webpage \[^{10}\] ................................................................. 14

**Figure 7:** YouTube Comment Section \[^{11}\] ........................................................................................................ 15

**Figure 8:** Test and Learn - UI ...................................................................................................................................... 17

**Figure 9:** Test and Learn - Testing XSS screen ........................................................................................................ 18

**Figure 10:** Results page ............................................................................................................................................... 19

**Figure 11:** SceneBuilder ............................................................................................................................................ 22

**Figure 12:** Part of the Initial Plan for the project ..................................................................................................... 23

**Figure 13:** Acunetix's vulnerable website .................................................................................................................. 24

**Figure 14:** Initial screen - Wrong URL format error .................................................................................................. 25
1. Introduction
Nowadays, technology is a part of almost every business area and a lot of human labor has been replaced by machines or algorithms. Moreover, we use computers, mobile phones and tablets all the time in our everyday lives. Most people also use a lot of online services such as banking and shopping. In order for all of the above mentioned to work properly there is a lot of hard work to be done and a major part of it is making sure that the software that all technologies use is trustworthy. The best way to do that is by testing that software against all real live scenarios it can possibly face. There are many examples of disasters that happened because of very tiny flaws in the software of the system left there because of the lack of sufficient testing. In 1990 AT&T network lines went dead for 9 hours resulting in big losses for the company. After analysis it turned out that a very small piece of code was the reason for the network failure. A mechanical problem in one of the 114 switching center caused it to shut down and when it was turned back on due the software error it sent a message to other switching centers and in the end the entire AT&T network was brought down [1]. Another example is the Ariane 5 disaster. The rocket was destroyed on purpose less than 40 seconds after its launch because the rocket started moving strangely. It turned out that the reason for this was a conversion from 64-bit number to 16-bit number. The designers assumed that the velocity of the rocket would not reach such a big number that will result in an overflow but it did. The total cost of the rocket and its cargo that were destroyed is 500 million dollars [2]. These examples are just two of the many big problems that occurred because of faulty software. That is why nothing should be assumed and everything needs to be thoroughly tested when developing software.

1.1 Project Overview
The aim of this project is to produce a series of tests and use them to evaluate whether a piece of software is trustworthy or not. Conditions for the software to have passed the tests must be determined as well. Although this seems like a project with a very well pointed and structured way of completion, its scope is actually very wide and allows the student to choose for themselves what exactly to focus on from a variety of possibilities. This gets clearer once the meaning of software trustworthiness, which is explained in the next paragraph, is understood.

1.1.1 What is Software Trustworthiness
In order to put a software into practical use you have to be sure it is working properly or in other words it is trustworthy. Thorough testing must be conducted before this can be verified because this is a complex concept that consists of different simpler ones. For a software to be declared trustworthy it must possess five key attributes (Figure 1):

1.1.1.1 Safety
The system must not enter any harmful states. This means that it should not present any physical or intellectual threat.

1.1.1.2 Reliability
The system must deliver its services as specified and cover all functional and non-functional requirements.
1.1.3 Availability
The system must deliver its services as specified at any point in time and any conditions without failure.

1.1.4 Resilience
The system must be able to recover, renew or transform in any way quickly in response to events such as updates, crashes or delays.

1.1.5 Security
The system must be protected against all kinds of attack whether they are deliberate or not. It must not have any vulnerabilities that can be exploited to steal data.

![Diagram showing the five attributes of trustworthiness](source: The British Standards Institution 2014)

Figure 1: The five attributes of trustworthiness

Every single one of these attributes must be tested before a software can be released.

1.2 Project Approach
After doing an in-depth research on what software trustworthiness is and the different attributes it consists of, it turned out to be much more extensive than initially thought and an application that can test in so much details all different types of software cannot be created by a single student in such a limited time. That is why after a few meetings with Dr. Daniel Dresner – the supervisor of the project, a decision was made to focus on only one of the five attributes of trustworthiness and a specific type of software. This plan allowed for producing more in-depth tests for this specific topic until the deadline. After the end of the project, the application can be expanded to cover either more types of software or other attributes of trustworthiness. We called this the bowtie approach.
I believe that security is the biggest problem with the software coming out nowadays because software developers seem to cope better with the other four attributes of trustworthiness. Many companies spend a lot of money to increase the security of their product. However there are still many security breaches and information leaked because of them each year. That is why a choice was made to concentrate specifically on this attribute. Developing web applications is something that I have previous experience in and that is why I decided to create a program testing the security of this type of software.

1.3 What else has been done in the area
As mentioned above security is probably the biggest problem with software nowadays. That is why many programs for testing web applications have been developed. These are just two of the more popular website vulnerability scanners found on the Internet. Both of them are professional tools used from many website developers to find vulnerabilities in their product.

1.3.1 Acunetix Web Vulnerability Scanner
Acunetix WVS runs automatic tests on a chosen web application to check for a variety of web vulnerabilities. It detects SQL Injections, XSS (cross site scripting), low password strength and others. As shown on Figure 3, this scanner provides very detailed information about what threats it finds and all tests it runs.
This tool's biggest advantage is that it is very fast, compared to other scanners, but it can also do very extensive scans in order to find all dangerous vulnerabilities in a web application. Acunetix WVS includes many different tools such as the Login Sequence Recorder which allows one to open a website inside
Acunetix and ‘show’ the program how to login (Figure 4). This allows for better scanning for liabilities in the login page which is where a lot of the hackers’ attacks happen.

![Acunetix Login Sequence Recorder (video screenshot)](image)

**Figure 4: Acunetix Login Sequence Recorder (video screenshot)** [5]

Last but not least, it is able to pinpoint the lines in the source code that create vulnerabilities [4]. This is achieved by the combination of black box testing (tests done without accessing the source code of the application that is being tested) and information from sensors placed in the source code during execution. Acunetix WVS uses graphical user interface and is available for all major computer operating systems.
1.3.2 Sqlmap
Sqlmap is different type of testing tool from Acunetix in many ways. It concentrates only on SQL injection scanning and it uses black box testing only. Moreover, it does not have a GUI and is running in the command line [6].

![Sqlmap demonstration (video screenshot)](image)

This tool can not only point the SQLi (SQL Injection) vulnerabilities but it can also exploit them and unveil information from the database of the website. That is why the creators of the application provide a disclaimer stating that it is illegal to use sqlmap on a website without the permission of its owner. Due to the lack of graphical user interface and the way tests are done sqlmap requires from its users the ability to work with CLI (command line interface) and some basic database structure knowledge.

1.4 What the final application and why is it needed
As previously mentioned, there are many web vulnerabilities scanners and the best ones are always developed by professional teams for a long period of time. They all provide very accurate and mainly very extensive tests, covering wide area of possible vulnerabilities. After some researching, it was clear that it would not be possible to create an application that can compete with the already existing professional tools for the limited amount of time given to complete this project. After a few meetings with the supervisor of this project - Dr. Dresner on which this problem was discussed, a conclusion was reached that instead of creating something that nobody will use because of all the better professional solutions that are already there, it is better to shift the focus of the application a little bit so that it provides its users something that does not already exist in other website vulnerability testing tools. Something that other scanners have in common is that they just list the possible threats and the user needs to find his own solution to the problems. That is why a decision was made that the final product of this project needs to not only find a vulnerability but also to show the user why it is a vulnerability and how it can be eliminated. That is why it was called it Test and Learn. Test and Learn is an application that provides automatic as well as manual tests that check if there are any SQLi or XSS vulnerabilities in a website and then gives an explanation of the problem and a possible solution. The target users of this software are mostly unexperienced web application developers who are new to online security and that
is why it does not require any previous knowledge in the area. It has a very simple graphical user interface that makes it very easy to use. It also provides step by step instructions on how to run the tests, but the most important feature is that for every test it provides an explanation on what vulnerability is being tested, how the vulnerability may have occurred and last but not least, how to prevent it.

2. Development

2.1 Research

At the start of the project my supervisor sent me a paper describing what the specification of software trustworthiness is [3]. Once I read it and understood what the project is about I started researching ways of testing software. I came across many tutorials and automated tools. However, none of them were the same, they were all testing different attributes of a software as well as different types of software. In fact, I could not find an application that can fully check whether a piece of software is trustworthy. That is why the initial idea was to combine the information I find to create a program that can do this. However, some tests were too specific for certain software, others were too detailed and time-consuming to do and combining all of them in one application, given the time constraint I have, would have been impossible or in the best case would have resulted in very subtle tests that do not at all guarantee the trustworthiness of a piece of software. That is why, as mentioned in a previous section, I made a decision to focus only on security testing for web applications mainly because this is something that I had some experience in since I have developed websites before this project. Having clarified what to concentrate on, I had to research in depth how this kind of software is usually secured and what are the threats to it. After visiting a lot websites and forums I noticed that the two most commonly mentioned attacks to websites were SQL Injection and Cross Site Scripting and they the most used tools by hackers to exploit web vulnerabilities [8].

2.1.1 SQL Injection (SQLi)

SQL stands for Structured Query Language. It is a language specifically designed to manage the data in relational databases. SQL Injection is an attack that uses this language to send malicious queries to the database and doing so reach the information in it. This means it is a server-side attack. It enables hackers to retrieve, insert, alter or delete data from the database. Unauthorized access to this information is extremely dangerous, especially if it includes personal information about the users of the website. The way SQL Injection is executed is by finding user input fields and type malicious commands there because often this user input is used in SQL queries to the database. For example, when someone tries to login in a website they type their username and password and when they click the login button the following (or similar) query is sent:

```
SELECT * FROM `users` WHERE `username`='john94' AND `password`='123456';
```

This command will not be a problem with this username and password; however, the user input may contain characters such as a single quote (‘) which will result in a database error because the query will have an odd number of single quotes that will not match.
This error will output information about the structure of the database and that is what hackers are trying to exploit. This is called error-based SQL Injection. The way the application works is by executing it on the website the same way a real attacker would. There is another type of SQL Injection called Blind SQL Injection which is used when there is a vulnerability but the website does not display any errors on the screen. This is very time-consuming and involves a lot of guessing. That is why the application does not provide any tests simulating this attack; however, this feature can be added in the future if the work on this project continues.

2.1.2 Cross Site Scripting (XSS)
Unlike SQL Injection, XSS is client-side attack. It still uses user input fields but this time one that are outputted directly on the webpage. An example for this are search fields: once the user types what they want to search for and click the search button, it is often displayed on the webpage.

If the input is unfiltered any HTML tags typed in will be executed as code. That is exactly what XSS is looking to exploit. It uses the `<script>` </script> tags indicating that everything between them is a Javascript code. This is a very powerful programming language that can be used from the attackers to do a lot of things such as stealing user’s cookie (HTTP cookies are a small piece of data that is sent from the website to the user’s web browser usually as a way of knowing that the user is logged in) allowing the hacker to use the website pretending to be that user. There are two types of XSS:

2.1.2.1 Reflected (non-persistent) XSS
In the reflected XSS attack the malicious code inserted in the webpage is directly executed. The hacker can then send an URL to this webpage and trick someone to click it (there are many different ways of doing this but it is outside the scope of the project). The code will run in the victim’s browser and it can send their cookie to the attacker.
2.1.2.2 Persistent XSS
In the persistent XSS the malicious code is inserted into a field that stores information in the database. Good example for this is webpages in which the user is allowed to leave a comment on.

Figure 7: YouTube Comment Section [11]

Figure 7 shows the comment section on a YouTube video where each comment is stored and then displayed on the webpage. Once this is done the code will run in the browser of everyone who loads this page [12].

2.2 Requirements
Before starting the actual implementing of any software it is very important to make sure that there clearly stated requirements for what the final application needs to be able to do. It is crucial that these are specified strictly according to what the goal of the application is and what the target users are. In this section the thought process when identifying the functional and non-functional requirements is explained.

2.2.1 Functional Requirements
As the name shows, functional requirements are the ones that determine the functionality of the software. In other words – given an input what should the output be. Those have been gathered by taking into consideration the deliverables needed for the project and the idea of what achievements are
targeted with this application, which is mentioned in a previous section. Since the goal here is to test a website’s security, the number one requirement for this software is to be able to identify web vulnerabilities that can be subject to SQL Injection or Cross Site Scripting. The way to this is by providing tests for each of the possible vulnerabilities. The other crucial requirement is what makes this application different from the others, namely its ability to teach the user how to prevent attacks on his websites from happening or in other words, how to fix the vulnerabilities and also where they can occur and how are they exploited.

2.2.2 Non-Functional Requirements
The non-functional requirements describe how the system should work. They are tailored according to the functional requirements, the targeted users and the aim of the application. This software needs to be designed in such a way that it is suitable for use from people with very little experience in web development. That is why one of the most important non-functional requirement for it is that it should be very easy and straightforward to use and no technical knowledge is needed for someone to use it. One of the biggest disadvantages of the most website security testing software that already exists is that it does not provide any information about the tests it uses and because of that the user only sees some results but does not actually understand what is being tested. That is why another important non-functional requirement for the application is to provide enough clarity about the tests done so that an unexperienced web developer can understand what vulnerabilities is the program checking for.

2.2 Design
In terms of what the final application should be, there are three choices: mobile, desktop or web application. The first type has become very popular in the recent years because of the incredibly fast rates at which the smartphone market expands. However, since the purpose of the project is to create a program that can be used by web developers to test websites, it would be very impractical to develop such an application for a mobile platform because of the limited functionality that it offers and the inconvenience of using the small screen of a smartphone for programming or testing purposes. This leaves only two possible options. The most notable advantage of creating a web application for this project, rather than a desktop one, is the easiness of access. It allows people to just enter an URL in their favorite browser and start using it. It also allows them to do it from anywhere on any device. However, desktop applications can offer more functionality because some of the more complex tests cannot be done from a website and that is why all web vulnerabilities testing programs are desktop applications. Moreover, I already had some web development experience before this project and I saw it as an extra personal challenge, which will also allow me to earn extremely valuable skills for a future career, to try and create a desktop application.
2.2.1 User Interface

The user interface for an application like this has to be very intuitive and responsive without overcomplicating it with too many items on the screen. This will allow the user to concentrate on what they actually opened the program for in the first place, namely testing their website for vulnerabilities and learning how to prevent them. The background in all different screens of the application is a bright color and all buttons and labels are in the grade gamma. This allows for a pleasant simplistic design that makes the information in the program easy to read. Something else that is hidden in this choice of colors is that it symbolizes the white hat and grey hat hacking also called ethical hacking. This is the practice of attempting to penetrate the security of an application without any malicious intent with the sole purpose of finding any vulnerabilities in it [13]. This is exactly what this project aims to do.

![Figure 8: Test and Learn - UI](image)

Figure 8 illustrates the initial screen that opens when the application is ran and the screen which allows the user to choose which vulnerability to test for. It is a demonstration of the simplicity of the design that does not have any redundant information. Everything is intended to guide the user straight to the point of testing.
Testing Software for Trustworthiness

**Figure 9: Test and Learn - Testing XSS screen**
Figure 9 represents the testing screen where the user will spend most of the time on. At the top there is brief general information about the attack. On the right hand side there are buttons for all the available tests. In the middle of the screen there is a text area with a description of the specific test and the instructions on how to do the test. For all of them there is a “How to Prevent” button which shows a pop-up window with information about the prevention of the particular attack (Figure 10). For each test the user has to manually answer if his application passes or fails. The logic behind this functionality is explained in Subsection 2.3 “Logic behind the functionality of the application”. For the tests that have an automated part there is also a “Run Test” button on the screen (first screenshot in Figure 9). Although on the testing screens of the application there are more items and more information it is structured in a very clean way that is designed to be easy to follow. After the completion of all tests in a section, a pop-up window with the tests results is shown. It follows the same minimalistic design. It also displays the final result of the test with a picture which is intended to draw the attention to it since it is the most important part of the screen (Figure 10).

![Results Page](image)

*Figure 10: Results page*
2.3 Logic behind the functionality of the application
This subsection explains the thought process behind the choices made for some of the more interesting functionalities of the program. It is all based on the idea that it is a testing but also educational application.

2.3.1 Separating the test for SQL Injection and Cross Site Scripting
The main reason for this decision is that SQLi and XSS are completely different attacks exploiting different vulnerabilities. If there is a single screen with both types of tests it is much more likely that the user gets confused with what each test is checking for. On the other hand, separating the tests in sections that are formed according to which type of attacks they cover contributes to the better understanding of the different vulnerabilities by the user. Moreover, some websites may exposed to SQL injection but at the same time completely protected against XSS for example. That is why it is better to for the application to be able to give a separate result for each of them. Another reason for splitting each type of attacks, probably the most important one, is that it keeps the application more generic meaning that in the future many more tests, concerning different types of vulnerabilities, can be added. Considering a program with one hundred tests, for example, it will be almost impossible to follow what is being checked by each one of them if they are not separated in clusters.

2.3.2 Manual instructions for running each test
Some of the tests have an automatic part done by the program (first screenshot in figure 9); however, because of the educational purposes of this application, no tests are completely automated. This is solely based on the idea of “active learning”, which requires the student to take actions and participate in the learning process instead of just listening or reading. Many studies show that this helps for much better understanding of what is being taught\(^{14}\). The unique selling point of this application is to teach while checking for vulnerabilities. That is why every single test is constructed of step by step instructions which helps the user learn how everything is being done in the smallest details. There are existing applications in the area that do all the testing automatically and then provides some information about them but this does not help an unexperienced web developer to learn the basis of the attacks and the way to protect his products from them.

2.3.3 The “How to Prevent” Buttons
This is a very big part of the learning process in the application. There are “How to Prevent” buttons below each test and for the failed tests on the results page. Using a single click immediately after doing a test to open a new window with information about how the attack is done and how to prevent it is the fastest way to learn more about it. The reason because of which these buttons are placed on each test screen is that users should be allowed to see the explanation for a vulnerability even if their website does not have it because the user may still be unsure why and how it may occur.
3. Implementation

The first part of this section contains a description of the technologies used for the project and the reason for choosing them. It also shows the working process throughout the whole project and compares the actual implementation with the initial plan submitted at the beginning.

3.1 Technologies used

3.1.1 Back End

As mentioned in a previous section, a decision was made that for the purposes of this project a desktop application is the most suitable. Since Java is studied in the first year of the Computer Science course in the University of Manchester it became the choice of language for the implementation of the Test and Learn program. Java is an object oriented programming language that is platform independent (it can be run on any platform that supports Java without the need of recompilation) that is possible because Java applications are typically compiled to byte code which allows them to run on any Java Virtual Machine on any computer architecture. It is very widely used and is very valuable to have a good practical knowledge of it since a lot of companies are looking for specialists in Java. Moreover, it is very suitable for this specific application because it can provide all the functionality needed in it.
3.1.2 Graphical User Interface (GUI)

For the user interface there were two possibilities – Java Swing and JavaFX. Both are very good for developing GUI using Java for the back end. Swing was released in 1997 and since then it has been the standard tool for creating GUI with Java \[^{16}\]. On the other Hand JavaFX is released 2007 designed to fully replace Swing. It still has not quite accomplished this but it offers a bigger variety of possible designs because it can be used in a combination with XML (Extensible Markup Language) and CSS (Cascading Style Sheets) to create more diverse styles of applications. The way this was made possible by Oracle is by creating Java FXML which combines XML and JavaFx and allows the style of the application to be modified with CSS. With this technology, for every single scene in the application that is being created, two files are necessary – one with the design of the scene (written in XML) and one java file that provides the functionality of the scenes (e.g. what happens when a button is clicked). Java FXML is an amazing, fast developing, way of creating GUI that is predicted to be very widely used in the near future. Because of this it was chosen over Java Swing for creating the user interface of this application. Another tool used for this part of program was SceneBuilder (Figure 11). It is a tool with GUI that works by dragging and dropping items (buttons, labels, text fields, etc.) onto a scene. It automatically generates XML code and this is extremely helpful because it allows for much quicker creation of the design without sacrificing on functionality.

\[\text{Figure 11: SceneBuilder}\]
3.1.3 Test running
The main idea for how to run the tests is to try and recreate actual attacks that hackers actually use in real life to exploit the vulnerabilities in different websites. The two methods that the application is using for this are SQL Injection and Cross Site Scripting (XSS). The former, as explained in a previous section of this report, uses SQL language that is used to make queries to the database of a website. In order to do the attacks a good knowledge of the language is required because actual complex queries were written. The XSS attacks are done using JavaScript. Since this application is not intended to be able to exploit vulnerabilities but just to identify them, the complexity of the JavaScript code needed was not high compared to the previous attack. However, some basic knowledge was still needed to find a proof for whether a website is vulnerable to XSS or not.

3.2 Working Process
In the initial plan for the project, made at the end of September, the beginning of the implementation was scheduled for late November (Figure 12). However, due to the extensiveness of the project, a lot of research was required in order to form a good idea of what the final product needs to be. Moreover, further studies were needed in order to understand how vulnerabilities can occur and how can they be exploited by attackers. Another requirement before starting to write the code of the application was to have a very good knowledge of the technologies described above. The combination of these factors resulted in the actual implementation starting a little bit later than initially planned.

<table>
<thead>
<tr>
<th>October</th>
<th>November</th>
<th>December</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct 12</td>
<td>Oct 19</td>
<td>Oct 26</td>
</tr>
<tr>
<td>Oct 26</td>
<td>Nov 2</td>
<td>Nov 9</td>
</tr>
<tr>
<td>Nov 9</td>
<td>Nov 16</td>
<td>Nov 23</td>
</tr>
<tr>
<td>Nov 16</td>
<td>Nov 23</td>
<td>Nov 30</td>
</tr>
<tr>
<td>Nov 30</td>
<td>Dec 7</td>
<td>Dec 14</td>
</tr>
<tr>
<td>Dec 7</td>
<td>Dec 14</td>
<td>Dec 21</td>
</tr>
<tr>
<td>Dec 14</td>
<td>Dec 21</td>
<td>Dec 28</td>
</tr>
<tr>
<td>Dec 28</td>
<td>Jan 4</td>
<td>Jan 11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>What is the project about</th>
<th>Plan my approach</th>
<th>Research other testing tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>Implementation</td>
<td>Functionality</td>
</tr>
<tr>
<td>Design</td>
<td>Testing</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 12: Part of the Initial Plan for the project**

A very important part of the application, namely the vulnerability tests, were to be created before the program because it needed to be tailored according to them. Writing those tests started in the beginning of January. After finishing with them, the implementation of the Test and Learning started at the end of January. Despite this late start, there was no time pressure when creating the application because the very thorough research beforehand and the good planning of the implementation process allowed for a problem-free execution. The application was in its final stage by the time the project
demonstration had to be done and was completely finished by the submission deadline. The testing was completed during the implementation and more information about it follows in the next section.

4. Testing and Evaluation

Every piece of software created has to be tested thoroughly before it is released. This is one of the reasons that this project is so important. Even though the application created for it is one that tests other software it still needs to be checked if it is correctly working itself. This section shows the tests that were done during the development of the project and the results of them. Since this application was made according to the time constraints of the project a lot of thinking has been done on how it can be improved or expanded in the future, before it was implemented.

As mentioned above, web attacks had to be written before the application. That is why its testing is considered to have started before the beginning of its actual implementation. Acunetix - one of the companies that already produced a web vulnerability detecting application also created a website that was initially created to test their program but is also free to use by anyone else.

![Acunetix website](testphp.vulnweb.com)

**Figure 13: Acunetix’s vulnerable website**

This website is developed in a way that makes it vulnerable to many possible attacks and that is why it was considered to be the best testing subject for this project. A set of attacks, all of which managed to break through the security of Acunetix’s website, were created and used in the final application. Another important part of the testing for the project was done on the Test and Learn program. It does not have a database with any information that users must not see so security tests designed to prevent hackers’ attacks were not needed. However, all input fields are checked so that the user provides the right type of information (Figure).
Figure 14: Initial screen - Wrong URL format error

Figure 14 is an illustration a wrong input error shown to the user. This particular input field is protected by the following function:

```java
class TestAndLearn {
    public boolean isValidURL(String urlStr) {
        try {
            URL url = new URL(urlStr);
            return true;
        } catch (MalformedURLException e) {
            return false;
        }
    }
}
```

Which uses the URL class provided by Java developers that can check whether an URL is in the correct format.

Use cases were created that had a predicted results of actions that the user can take. These were compared with the actual results that the program returns to make sure that all of the functionality of the application is working as intended. Since this testing was done during the process of developing and not after it, it allowed for many mistakes in the code and logic of the program to be corrected in a timely manner.
5. Conclusion

5.1 Problems Faced
The most difficult part of the project was definitely the beginning. Figuring out what a successful product would look like in the end was very difficult and required a lot of research and meetings with the supervisor of the project. Once the direction to go for was chosen another big obstacle came out. Since SQL Injection and XSS are very dangerous attacks which are most used ways of penetrating website’s security there is not a lot of information available for free access on the internet. That is why this was when most of the effort was made. A way of only testing for the vulnerabilities without actually exploiting them had to be found which required deep understanding of the attacks. An additional challenge was to learn how Java FXML works and a lot of practice was required before starting the implementation of the final application.

5.2 Future Work
This application was made with great consideration for its future after the submission of the project. At the moment it contains only six tests that after a lot of research were considered crucial to identifying the most common and dangerous vulnerabilities. However, there still are other attacks that can harm a website in different ways. That is way this application was created in generic way (tests are separated in clusters according to the vulnerabilities they are designed to identify and they are done by the user using step by step instructions from the program). This allows the programmer to add more tests to the program very easily while keeping the initial idea of the application being educational. Moreover, the approach to this project, as shown in Figure 2, was to narrow it down to security for web applications and once this is fully covered, to create tests for other types of software, such as desktop applications or software that controls different machines. It can also be expanded so that it covers all five of the attributes of Software Trustworthiness (Safety, Reliability, Availability, Resilience and Security).

5.3 Reflection
Creating an application in an area that so many companies invest a lot of money in (security) was very difficult. However, the idea of adding an educational part in it is innovative and makes very well employable. This was a very challenging project that allowed for a great self-development in terms of it being a very good practical experience that will definitely be very useful in the future. New technologies were learned, which was one of main the reasons for choosing to use them. Anyone who wants to make a career as a Software Engineer needs to be aware of the security risks that exists for every application. That is why the most important part of the project was that it was about testing software. Having to create an application from the beginning to the end and facing all the different challenges throughout the development was a great preparation for a real working environment.

References
Available from: <https://around.com/ariane.html>
[cited 10/04/2016]

[cited 10/04/2016]

Available from: <http://www.acunetix.com/vulnerability-scanner/>
[cited 11/04/2016]

Available from: <https://www.youtube.com/watch?v=uM6X4zXRoE>
[cited 11/04/2016]

Available from: <http://sqlmap.org/>
[cited 11/04/2016]

Available from: <https://www.youtube.com/watch?v=RsQ52eCtI4>
[cited 11/04/2016]

Available from: <https://www.port80software.com/support/articles/most-dangerous-web-vulnerabilities>
[cited 12/04/2016]

Available from: <https://www.port80software.com/support/articles/most-dangerous-web-vulnerabilities>
[cited 12/04/2016]

[cited 12/04/2016]

Available from: <https://www.youtube.com/watch?v=F3e1zGh3sro>
[cited 12/04/2016]

Available from: <https://www.youtube.com/watch?v=M_nllcKTxGk&list=PL1A2CSdiySIGRec2pvDMkYNi3iRO89Zot>
[cited 12/04/2016]

Testing Software for Trustworthiness

