Penetration Testing of Web Applications in a Bug Bounty Program

Pascal Schulz

Faculty of Health, Science and Technology

Computer Science

15hp

Leonardo Martucci

Donald F. Ross

140604

Penetration Testing of Web Applications in a Bug Bounty Program

PASCAL SCHULZ
Department of Mathematics and Computer Science

Abstract

Web applications provide the basis for the use of the "World-Wide-Web", as people know it nowadays. These software solutions are programmed by a numerous amount of developers all over the world. For all this software, it is not possible to guarantee a 100 percent security. Therefore, it is desirable that every application should be evaluated using penetration tests. A new form of security testing platforms is provided by bug bounty programs, which encourage the community to help searching for security breaches. This work introduces the currently leading portal for bug bounties, called Bugcrowd Inc. In addition, web applications, which were part of the program, were tested in order to evaluate their security level. A comparison is made with statistics provided by leading penetration testing companies, showing the average web application security level. The submission process, to send information about vulnerabilities, has been evaluated. The average time it takes, to receive an answer regarding a submission has been reviewed. In the end, the findings are retested, to evaluate, if the bug bounty program is a useful opportunity to increase security and if website operators take submissions serious by patching the software flaws.

Keywords: Penetration Testing, Bug-Bounty Program, Web Application Analysis.

This thesis is submitted in partial fulfillment of the requirements for the Bachelor's degree in Computer Science. All material in this thesis which is not my own work has been identified and no material is included for which a degree has previously been conferred.

_____

Pascal Schulz

Approved,

_____

Supervisor: Leonardo Martucci

_____

Examiner: Donald F. Ross

Contents

1 Introduction

This chapter presents the main goals of the project, the motivation for the research and a short overview including the results, which have been evaluated during the execution of this work. Additionally, the dissertation layout is outlined briefly.

1.1 Project Goal

The main project goal of this thesis is, to execute multiple web application penetration tests on websites, which are part of a bug bounty program. With these security analyses, the question has to be answered, if security checks, made by the community, can help to improve the security level of a website. The associated platform that lists the websites, which are tested, is evaluated regarding their service and processing of the interaction between tester and website operators.

Concomitant with the use of these security programs, it is being evaluated, how convenient the submission process is and how the contact with the operator is designed. The vulnerabilities discovered on websites, listed on the bug bounty platform, are submitted not directly to the website owner, but through an online form available on the platform. Detailed information on the vulnerabilities discovered are provided, such as the way to exploit them, the security impact and patch solutions. The amount of time, how long it takes to inform the tester, if the finding is valid or not, is also being examined. After the reply of the bug bounty portal and an appropriate time to fix the flaw with a software patch, the website should be retested to check if the website operators take the submissions seriously.

1.2 Motivation

The existence and the success of bug bounty programs is connected with the effort given by security researchers all over the world. All of these people should be taken into account or at least have the feeling to help unsophisticated people in their field of expertise. A big motivation are people that are imprudent when using web applications due to their weak awareness regarding information technology security. Family and friends especially serve as good target groups who should be prevented from becoming a victim in online frauds and swindles. Therefore, security specialists should use as much energy as possible to ensure safe access to all the media and applications on the internet. Evidence of how many people became the target of attacks is given by many requests on respective forums or in newspapers asking for help.

Furthermore, the number of significant organised hacking incidents, in which companies with many customer data are the target, is rapidly increasing. Therefore, people start to get an insecure feeling, in case many sensitive data is demanded by certain web applications in order to work properly. One example that received a lot of attention was the big outage of the "Sony Playstation Network" in April 2011. "Criminal intruders stole personal information from all of the approximately 77 million PlayStation Network and Qriocity service accounts." [1] The company itself confirmed on their website that in this case they believe

"that an unauthorized person has obtained the following information [...]: name, address (city, state, zip), country, email address, birthdate, PlayStation Network/Qriocity password [...]". [2] With this development in online activities it is hard to keep a reassuring feeling about sharing all the personal data with many different companies all over the internet.

To eradicate all these drawbacks, more evaluations of computer systems and especially of web applications have to be executed, as they are used by numerous users.

1.3 Scope

First the selection, which bug bounty program should get analysed and used for searching websites to test, was made by using the currently biggest portal available instead of comparing multiple existing ones. Bugcrowd Inc., as the leading portal for bug bounties offers two kinds of lists to find websites to test. The one only visible for registered users has just a few entries, as website operators have to pay to get on this list but additionally get extra service. Only these paid bug bounties were taken into consideration for the security evaluations executed for this thesis. This is because of the fact that only paid ones get active support by Bugcrowd.

The scope for the penetration tests was set on evaluating web applications; hence, testing mobile applications or client applications was skipped. For the testing process itself, the main goal was to test the websites for the most common vulnerabilities (XSS, SQL-Injection, etc.). Instead of searching for a long time to find a specific kind of a software flaw, the focus was set on using a broad spectrum of different attacking techniques. Due to restricted time, the goal was just set to analyse three different web applications regarding their software vulnerabilities.

1.4 Method

The project was divided in several sub steps. First, research was conducted by reading literature about web application penetration testing. Several examples on how to exploit vulnerabilities have been executed, to make it easier to understand the attacking principles.

A testing environment was created on the local laptop, to have every useful application installed, which is necessary to execute penetration tests. The software was explored and the main functionality was documented.

Next, the relatively new concept of bug bounty programs, which are established for evaluating the web application security of websites, was introduced. After becoming acquainted with the basic concepts of these programs, the currently largest provider, which brings testers and websites operators together, named Bugcrowd Inc., was explained including some important background information. A short research was conducted on further possibilities to receive information about new and existing bug bounty programs.

A user account for the bug bounty portal was created and the backend of the web application

was screened. With the knowledge of how this program was working, the first website was picked for investigation. For each website, the initial step was to gather valuable information about the structure and the technology used by the website. This preliminary process was documented, to offer a better understanding about which information is important to know. After the information gathering process, specific tests were made. The main testing process was the same for each application. Nevertheless, it was important to regard special structures and functionality of the web applications. The most important ways and tests, how to search for vulnerabilities, were documented again. During the testing, a few vulnerabilities were found on the websites. These security flaws were reported to the website operator, using the provided form by Bugcrowd Inc. With this process, it was possible to submit vulnerabilities found in a safe manner; hence, nobody else could abuse the software flaw. The notification about the validation status was awaited. This was important to see if the found vulnerabilities were credited as valid, invalid or duplicate submissions.

With the statement of the website operator, the information was available if they patched the security flaw or not. Some responses did not contain information about the correction state. Therefore, the assumption was taken that these web applications will be patched as soon as possible. To check that and to know if the operators took the submissions serious, it was necessary to retest all submissions found.

1.5 Contributions

The main contribution made in this thesis, is the execution of security analyses of websites. Three different web applications have been tested. Every single one was designed for a different purpose. In total, six submissions were evaluated as valid and two more as duplicate findings, which means that somebody else had already found those security flaws. The valid findings were forwarded to the website operators by the bug bounty platform, that these have all the information needed, to fix the vulnerabilities.

The first web application tested was eBay Magento eCommerce, a shopping platform, which was bought by the online auction company. A few vulnerabilities have been exploited on this website. It was possible to send emails in form of a newsletter to an arbitrary amount of people all over the world. During the checkout process of the software, there was a security breach that let the attacker guess valid coupon numbers, in order to save money, as often as he wanted. Furthermore, it was possible to find some implementation weaknesses, which was not validated as severe enough to be credited. These findings allowed an attacker to guess valid user names as often as he wants, to guess the current password in the password change functionality, to list directories with data stored on the webserver and to have the opportunity to unlimitedly guess the password of an existing user during the login process.

The second application, which has been tested, was the bug bounty platform itself. In the backend of www.bugcrowd.com, it was possible to find three security flaws and another duplicate one. First, it was possible to change the password length to the length of only one character even though the website required the user to use at least eight characters for the password. This was possible by inserting a control character after the first letter in the pass-

word, which cut off the rest. The second finding was the password quality rule set, which allowed the user to just use, for example eight lowercase letters. A password like this would be too weak, nowadays it should contain at least lowercase and uppercase characters including a digit or a special character. Additionally, it was possible to use the defect password change algorithm to enter a password with which it was not possible anymore to successfully log in to the application. Evaluated as a duplicate finding, Bugcrowd Inc. sent a statement that the possibility to guess users by using a brute force attack had already been found by another security researcher before.

The third and last penetration test was executed on Heroku, which is a platform-as-a-service provider for developing code. This web application structure was more difficult to understand, only two submissions have been sent to the website operator. First, there was a password weakness, allowing an attacker to just use one security criteria for a new password, although the requirement stated that a valid password has to contain at least eight letters, numbers and symbols. This finding was evaluated as valid and credited with points for the internally managed skill ranking. The second finding was a duplicate one, which granted an attacker the opportunity to change the password length to zero characters.

To sum up, all these submissions led to 25 points in the internally managed skill ranking by www.bugcrowd.com, which gives testers the opportunity to get more highly sophisticated penetration test offers. Additionally, for three findings, there was a monetary reward obtained from the website operators.

1.6 Thesis Layout

This thesis is divided in five chapters including an appendix at the end of this work, providing the reader with a link to a compressed folder with more detailed information about the tests executed.

Chapter 1 presents the reader the main goals, which should have been reached during the writing of this thesis. The motivation, why it is important to deal with this topic is given, followed by a quick summary over all working steps which have been taken. Next, the general testing results are stated in a short way.

Chapter 2 explains the project background, the aim and importance of a penetration test and the legal basis of these security evaluations are explained briefly. To give the reader an idea about the severity of the web application security situation, there are statistics showing the development of web application vulnerabilities over the past years.

Chapter 3 contains the main information about the practical work executed for this thesis. First, the general project expectation is elucidated. Second, the program, which defines the scope for all the tests, is explained, followed by a description about the setup of the testing environment and which tools are used. Then basic information is provided for the whole testing process, beginning with the gathering of information, the attacks themselves and the submission process to inform the website operator. The last part introduces the three web-

sites, which have been tested during this thesis. A basic understanding of all functionality offered by these web applications is given.

In Chapter 4, the reader can find information about the vulnerabilities found. To make it easier to understand, some figures are provided showing the attacking process or the web front end.

In the final Chapter 5, a conclusion is drawn on the goals, expectations and the practical work. Further, a brief insight is given about problems which appeared during the thesis, and plans as to how this work can be used in the future for further research and which steps are already in preparation.

2 Background

The modern life, in which people are living in the year of 2014, is moulded by the use of the Internet. Beginning with reading current news or checking the weather, the reasons for using the internet are becoming more and more with every single day. During the past 25 years, the appearance of websites in the "World-Wide-Web" has changed from static text, giving the user specific information, to dynamic web applications, which are constantly connected to multiple servers, interacting with the user and delivering everybody a different experience in usage. Today, it is possible to transact money, get in touch with all the friends, share data with specific user groups or just spend time playing games. Concomitant with this constant improvement and evolution of web-services, the number of vulnerabilities of web applications has been rising as well. Due to this increase, the importance of developing new methods to secure this media also has been growing.

One of the most important aspects from a security point of view is to ensure the three main goals of security in information technology which are given by the terms *confidentiality*, *integrity* and *availability* [3]. To understand the principles of how to find vulnerabilities regarding the previous mentioned factors, it takes a long time of studying and practical experience in searching security flaws. Assuming that a person has a good technical understanding of web application security, it is still necessary to update the technical knowledge on a daily basis, because there are new exploits found by multiple security researchers all over the world. In order to harden a web application, it takes a long period of testing by security specialists, which is called penetration test in the technical jargon. The current situation is that only a fraction of all existing web applications in the web has been the target of such a security evaluation. The main reason is that these tests are expansive and not affordable for most website operators. There is also a lack of security awareness given, for example in the top level management of bigger companies, which leads to complete misunderstanding as to why they should pay high sums of money for tests, which are not bringing any improvements at first sight.

A relatively new alternative, which has been growing in the past years, is community help. For personal reasons, there have already existed many different possibilities in the web to ask other users for help, which have special knowledge in a specific field (for example, as in discussion forums). However, companies forwent to ask the community to help for a long time due to information disclosure reasons. Now more and more businesses understand that they can benefit from the widespread skills of numerous specialists around the world by allowing them to help. Concerning the security of websites, there have been created a few different platforms, which collect uniform resource locaters (URL[1]) of several websites including a short briefing of which parts are allowed to investigate and which operations should be avoided. One of this platforms is Bugcrowd (www.bugcrowd.com), which will be introduced in chapter 3.2.2 of this paper. The benefit for the community is that they have opportunities to try to improve their hacking skills in a real environment without being

[1] URL = A uniform resource locater "identifies a resource via a representation of its primary access mechanism (e.g., its network "location")" [4]

sued.

2.1 Importance of Study

Every single person is characterised by a lot of information, of which some is very sensitive. Web applications demand these sensitive details of their users, to grant them access to different kinds of existing web portals. As the majority of people have no idea about data protection and security flaws, the accountability to ensure a safe use bears in a small number of security experts all over the world. The process they can use to harden a software and increase the security is called penetration test and will be introduced in section 2.2.1. Until a short time ago, when community wide penetration tests had not been existing, companies specialised in executing these tests, charged much money for investigating software. As most of the website operators could not afford to pay for these examinations, the security level of their application stayed more or less untouched.

With bug bounty programs, which are being introduced in chapter 3.2.1, there is finally a new approach existing to widen the opportunity of receiving a penetration test. With this possibility, more and more operators can afford letting their applications be tested and can therefore increase their level of security. As bug bounty programs open this field of study to more people, it is important to conduct some research on this topic. Furthermore, the level of reason and the quality of this service should be examined.

In case bug bounty programs will take off and more and more operators realize that they can benefit by letting the community help, a big step could be taken into the direction of a more secure web. Subsequently, it would be interesting to think about other new models, which would make penetration tests even more accessible, but this topic will not be expanded on in this thesis.

2.2 Project Background

The process to evaluate the security level of websites is called penetration test. Sections 2.2.1 and 2.2.2 present a general overview.

2.2.1 What is a penetration test?

A penetration test describes an extensive evaluation of security mechanisms implemented in computer systems, which should protect users and sensible data stored on the servers. The "United States Department of Defense" has already published a short explanation of the term penetration testing in an early paper in 1987 regarding system evaluation with the following words:

"*The portion of security testing in which the penetrators attempt to circumvent the security features of a system. The penetrators may be assumed to use all system design and implementation documentation, which may include listings of system source code, manuals, and circuit diagrams. The penetrators work under no constraints other than those that would be applied to ordinary*

users. (italics added)" [5]

It is important to differentiate between penetration testers and people who attack websites in order to benefit from retrieving sensitive data. In the information security jargon, the terms "white-hat-hacker" and "black-hat-hacker" are often used to characterize the different motivations people have for starting a security analyses, whereby "white-hat-hacker" represents the indulgent person. [6] During the last years, more and more companies have been founded only concentrating on offering penetration tests. Consequently, the use for people with special education in this field of study has also been increasing.

A penetration test is a long-term process that needs extensive pre-agreements between the customer who is ordering the test and the company executing the security evaluation. The first and most important step is to declare a scope in which the client specifies the systems that are allowed to be tested and the mechanisms to be used. [7] An additional useful part of this scope can be a special focus on core functionalities of an application. Penetration tests usually have a short interval due to their high costs. Therefore, a special focus helps to keep track of evaluating the most critical software parts, which are specified by the customer. After the primary conditions for the test have been settled, the client has to support the security researchers with details as to how to connect to the application and/or to log on. Depending on the role of the user that the customer wants to be tested, he is able to decide whether he is only giving access rights with low privileges or full access to all single parts of the software.

With all previous mentioned information, it is possible to start with the technical part of the evaluation beginning with downloading necessary tools and setting up an appropriate testing environment. Normally, penetration testers have already installed most of the core tools they need for testing, hence, they just have to add programs in case of a special need for a given order. It is also possible to use pre-configured operating systems which only aim at providing a perfect environment for penetration testing (an example would be Kali Linux[2]). In the field of security analyses, there are well known vulnerabilities, which can show up in web applications. The typical approach is to start with these, while it is important to adapt these mechanisms in order to find flaws (further information will be given in Chapter 3 of this work). Overall, it can be said that there is no strategy that guarantees to find software bugs. The most common situations that security researchers are facing are trial and error scenarios in which testers approach a possible flaw systematically. During an analysis, it is important to document all steps, beginning with writing down findings and the way it is possible to exploit the software, ending with all other tests even if they failed. It is important to keep in mind which tests have already been executed to give possible team members an overview to check the tasks if they share the document. It is a good way to deal with your client if you support him with a final report, explaining all the findings in an understandable way, giving additional information how to fix and avoid the found problems in future. [8]

Another important aspect of penetration testing is retesting. Especially if the software is

[2]Kali Linux is a specially designed Linux distribution for penetration testers with pre-installed applications which are useful for security evaluations. The download of the distribution is free and can be found in the internet at http://www.kali.org/downloads/

updated and further components are installed, it is highly prudent to let the application be tested again. An additional good advice from the sight of the company developing the application is to change the researchers who execute the security evaluations after every test. The reason for this is that individual security specialists have different expertise and are usually using different approaches to find software flaws. This means that there is a high probability that different testers will find different vulnerabilities.

2.2.2 Why are penetration tests important?

All web applications people use in their daily life are a result of programming using different languages designed by programmers. In companies specialized on web-design, there is most of the time more than just one person working on one web application. In this initial situation, it is not always easy to cooperate in order to design web applications properly. Furthermore, not all employees have the same knowledge of programming that can lead to minor quality in certain parts of a web application. Additionally, there are many different technologies used in a state-of-the-art application to guarantee the user an excellent experience in usage and a proper design and behaviour.

All of these pre-mentioned factors form the underlying cause for the appearance of software bugs in applications, which can lead to severe security issues. Of course, people writing the code for an application try to do their best without making mistakes but it is not excludable to overlook bugs. On the contrary, most of the time web designers do not even know that they have built in security flaws. A good way to deal with these situations is to let written code be reviewed by another person with at least the same skills. This quality of service departments are implemented in bigger companies. The tests being executed are white box tests in most of the cases, which means that the testing person has the source code available and a general understanding of the communication flows of the software. [9] This is a first step to check the quality, but it is even better to let the software get checked again by another person using a black-box approach, which means that this person just has the final product without any further information. [10] In this way, the testing person has to find different ways to search bugs and may therefore find different weaknesses.

These processes have been describing the normal sequence of software developing without giving a special focus on security. The first step always has to be to check the core functionality. If the final application fulfils all given requirements, then there is finally the point when programmers should consider ordering a penetration test to check their software.

A lack of implemented security features can lead to severe consequences for the application owner. If sensitive data is leaked by hackers, the likelihood that there will be media scandals and indignant people who spread the word is quite high. The following listing shows a few aspects which could be influenced by such an incident:

Reputation

One of the most important economic factors for companies is their reputation. In modern economy, it is important to have good marketing that encourages people to pay for the products. Often, the products which have a better quality are not being sold as much as the inferior ones due to a lack of advertisement and buzz marketing. Web applications with minor security protection mechanisms have a higher risk of becoming the target of successful exploitation. In case that very sensitive information is stolen (e.g. credit-card-information, political attitude, etc.), the reputation of these companies is reduced, which leads to the following point.

Reduction of usage

Concomitant with a loss of reputation the amount of users making use of an application is decreasing, which leads to lower turnovers in the annual balance sheets. Besides, many applications build on revenues of advertisement, which are placed in banners in the graphical user interface of a software. These incomes decrease if more and more users decide not to use the program anymore.

Loss of money (trough fines)

Governmental fines could also be imposed to the owners of a website running an application that was hacked. Especially for small companies, the danger of registering as insolvent increases with the prospect of being fined.

2.3 Legal basis

The legal basis for the execution of penetration tests is very difficult and varies from country to country. This paragraph gives a short overview about the legal situation in Austria where the corresponding laws are given by the national criminal code.

Following sections of the law are given by the criminal code concerning criminal actions against computer systems:

- § 118a. Widerrechtlicher Zugriff auf ein Computersystem (Unlawful access to a computer system)
- § 119. Verletzung des Telekommunikationsgeheimnisses (Violation of telecommunication privacy)
- § 119a. Missbräuchliches Abfangen von Daten (Abusive interception of data)
- § 120. Missbrauch von Tonaufnahme- oder Abhörgeräten (Abuse of recording or observation systems)
- § 126a. Datenbeschädigung (Damage of data)

- § 126b. Störung der Funktionsfähigkeit eines Computersystems (Disturbance of functionality of a computer system)
- § 126c. Missbrauch von Computerprogrammen oder Zugangsdaten (Abuse of applications or access data)
- § 148a. Betrügerischer Datenverarbeitungsmissbrauch (Fraudulent information processing)

Although it seems that the criminal code contains several sections of the law for the abuse of computer systems, their description is not accurate and offers a legal grey area.

As an example, paragraph 118a. (1) states that "an access to a computer system is unlawful, if the person who accesses the system, makes us of the data found[3]". This declaration does not declare, if an action on a computer system that belongs to a different person, was law-abiding or not.

Private persons, who are only scanning the internet for web vulnerabilities in order to improve their skills in the field of penetration testing, normally do not have special intentions to sell or make use of received data. Nevertheless, in this case it is difficult to say, if these people get liable of prosecution. In a "Bug Bounty Program" (described in 3.2.1), the owners of web applications grant special rights to private penetration testers to ensure them that they are not getting sued for executing security evaluations using special mechanisms to find security flaws. However, this actually still means that the law gets numerous security researchers break the law in order to be able to help the application owners.

Companies that are selling penetration tests as their product sign contracts with their clients, in which the customer confirms that investigating activities on their computer systems are allowed. In this scenario, the employees of these companies again have to act against the law to fulfil their tasks.

As these security evaluations, trying to hack assets of other people, are relatively new, it will take more time to specify more appropriate sections of the law regarding this topic. So far, criminal codes only offer an inaccurate clue about the tasks allowed and the ones forbidden by law. Therefore, security researchers, especially in a private environment, should take care about themselves in order not to get in trouble with the law.

[3] Original Version in German: "Wer sich in der Absicht, sich oder einem anderen Unbefugten von in einem Computersystem gespeicherten und nicht für ihn bestimmten Daten Kenntnis zu verschaffen und dadurch, dass er die Daten selbst benützt, einem anderen, für den sie nicht bestimmt sind, zugänglich macht oder veröffentlicht, sich oder einem anderen einen Vermögensvorteil zuzuwenden oder einem anderen einen Nachteil zuzufügen, zu einem Computersystem, über das er nicht oder nicht allein verfügen darf, oder zu einem Teil eines solchen Zugang verschafft, indem er spezifische Sicherheitsvorkehrungen im Computersystem überwindet, ist mit Freiheitsstrafe bis zu sechs Monaten oder mit Geldstrafe bis zu 360 Tagessätzen zu bestrafen."

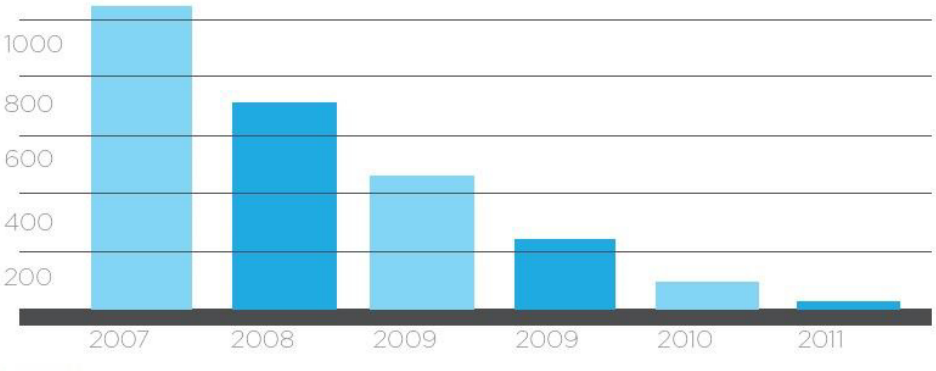

Figure 1: Vulnerability Historical Trend, The annual average number of serious vulnerabilities discovered per website per year. [11]

2.4 Usage and Statistics

There is no web application that can ensure a security level of 100 percent as there will always remain at least a slight probability of unexpected vulnerabilities. Therefore, it would be advantageous that every single software running on the internet should be evaluated before going online. Unfortunately, this situation cannot be summoned. It is also not possible to state the percentage of programs that have been checked, but the number is very small.

Nevertheless, it is possible to create various statistics with following questions:

- Which vulnerability class is the most common?
- Which sector of industry has the most vulnerable websites?
- What is the average duration to fix the flaws in this sector?

For this purpose, several well-known companies that are executing penetration tests as their daily business collect data and publish it in an anonymous way. It is possible to calculate numbers answering all the previous mentioned questions. One of the most important aspect, is the average count of single flaws per web application. Figure 1 shows the development from the year of 2007 until the year of 2011. [11]

One of the most interesting statistic is the distribution of vulnerability classes over all web applications regardless of their industry segment. Figure 2 shows an overview of detected vulnerabilities in the year of 2012 given by WhiteHat Security, an American company which publishes an acknowledged report on website security on a yearly basis. [11]

In general, it can be said that web-security is receiving more and more focus of developers and

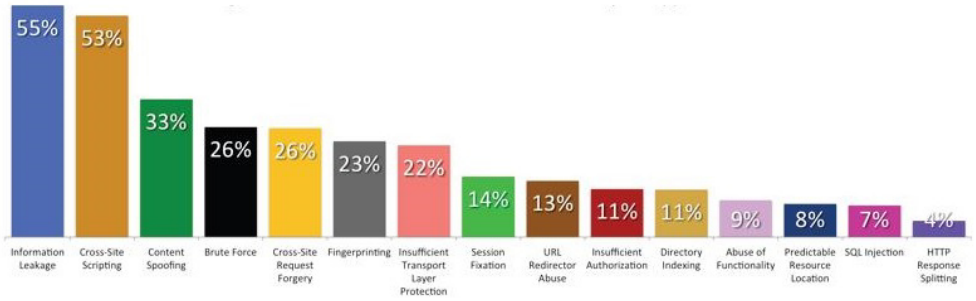

Figure 2: Percentage likelihood that at least one serious vulnerability will appear in a website. [11]

people who are using these web applications. This has led to a higher security of software, which is validated by statistics that have been made during the past years.

2.5 Chapter Summary

Society in 2014 is used to rely on the internet for searching various information and to communicate with other entities. Therefore, websites have changed over the past years, offering more and more services. With different requirements, it was important to develop new technologies to realize specific concepts. It takes a long time and hard work for software developers to get to know all these different technologies used. Nevertheless, it still stays important to develop further skills in this field of technology due to the rapidly changing technical background.

On the internet, millions of websites use numerous different web applications to guarantee their users a special experience. Many of these websites demand registering with partly sensitive information of the user. In order to keep this information secret, it is necessary to implement multiple security mechanisms. Unfortunately, the awareness for secure information systems is still not high enough, which leads to several web applications that are not safe.

A process to check and ensure a specific level of web application security is called penetration test. During this evaluation, most of the parts of a software is checked if they contain vulnerabilities that could be exploited by potential attackers. These tests cost considerable money and are not affordable by most of the website owners. Nevertheless, software quality has increased as computer security in general over the past years through various reasons. Modularization and the use of content management systems brought a higher level of security as many security researchers checked these systems.

The legal basis for executing penetration tests is not completely transparent in different countries. For companies that have contracts with their clients, there is no problem to execute penetration tests on the customer systems. People who are not working in security compa-

nies that offer these tests, only have little opportunities to learn the demanded techniques in order to improve their testing skills.

Overall, the topic of information security has become more popular in the past decade and will still become more important in the future. Companies specializing on testing web application are getting more and are searching for qualified employees. Prices for security evaluations will decrease, which leads to more orders, affecting the level of web application security in a positive way.

3 Project Design and Implementation

This chapter presents the essential part of the thesis, describing the environment and the approach taken to receive information about the security level of websites, which are taking part in a community wide evaluation. The program dealing with this security analysis is explained briefly, followed by the tools used to reveal security breaches. The testing routines that are used to detect vulnerabilities are described in a general way, accompanying by the web applications on which these techniques have been used.

3.1 Project Expectation

Web applications are the result of intense work by a single person or a group of software developers. The quality of these programs, especially if it comes to security mechanisms is varying greatly in size. Although nowadays the focus on security in these applications is higher as it has been years ago, there are still many security flaws to find, as it is impossible to guarantee a 100 percent security.

Hacking websites is an illegal activity that is executed by a numerous people spread all over the world. Regardless of the reason and purpose these people have, these activities have in common that security breaches are not being reported to the website operator. In case the owner of an application detects some kind of misuse going on, he still has to figure out, how it was possible to compromise the software. Others do not even know that they have been the target of an attack. In these scenarios, the operators suffer from hacking activities and cannot or hardly can improve the security mechanisms of the website.

Operators who decide to let their websites be tested by the community receive detailed instructions about the way it was possible to compromise their software. With this helpful information, the owner can quickly try to fix the bug and let it be tested again. This approach helps to increase the security of a website enormously.

In this thesis, only web applications are tested, which have been the target of an evaluation in bug bounty programs before. Therefore, the fundamental expectation is that it is hard to find security flaws. Nevertheless, as websites are going through changes of context and functionality over time, a certain probability is still given to find breaches. The assumption is that basic security flaws, as they can be found in unsophisticated software, are already eliminated. This would lead to the necessity to dig deeper into the software in order to find vulnerabilities.

With several attacking techniques, the level of security is being evaluated for different web applications. The process to find security flaws is documented and the operator has to get informed about possible findings. The reaction to the submission should be await, followed by a retest to make sure that operators take their website security seriously. In the end, a conclusion should be drawn regarding the significance of penetration tests in the context of bug bounty programs.

3.2 Project

This section gives a short overview of the whole bug bounty program and furthermore, information about different approaches to find bug bounties including the introduction of a framework for companies which want to get their websites tested.

3.2.1 Bug Bounty, General Information

A bug bounty program is an offer of web application operators to penetration testers all around the world, enabling them to test their websites restricted by certain limits legally and often rewarded with different kinds of awards and prizes.

Before bug bounty programs had been existing, people tested web applications and often published a proof of their hacking skills on certain forums and web platforms. Although this is not legal, these security problems were exploited by more people, which led to serious problems for the application owner. The interval, between a vulnerability was published and a software patch was implemented that fixed the flaw, was exceedingly long.

This is the point where bug bounty programs start to be a big benefit for operators of web applications. Under certain restrictions (e.g. target to be tested, vulnerabilities to be tested, etc.), penetration testers have the opportunity to put their skills in finding security flaws to the proof. The main difference between ordinary hacking of websites and bug bounties is that the testers are required to commit their findings to the operator before publishing information about the vulnerability. The submissions made have to describe in detail, how to exploit a software flaw and the impact which can occur. Additionally, it is a good way to add as many additional information as possible (e.g. HTTP-Requests, screenshots of the findings, etc.) to make it easier for the operator to reconstruct the exploit. As service in return, the owner of an application should show his appreciation. Certain procedures are common, beginning with just mentioning the person on the website in a hall-of-fame, till rewarding the testers with gifts or even money according to the severity of a finding (rewards range from $50 to more than $5000).

Although this procedure between the operator and the testers seems to be fair for both sides, bug bounty programs can also bring many problems. If the team behind an application receives too many submissions, it may be possible that they cannot handle the number of findings in an appropriate time, which encourages testers to publish their findings early as they do not receive any feedback from the operator. There is also another problem with duplicate and false submissions. The employees of the company running the application have to check every single finding handed in by testers, which is very time-consuming and costs indirectly considerable money as they cannot work on productive objectives during this time. [12]

3.2.2 Bugcrowd.com

Bugcrowd.com is a web application itself that offers a framework to operators of websites to let their software be tested. It acts as platform to find and receive information about actual penetration tests, which are currently running and even offers support for evaluating the

submissions by built-in analysis tools.

General Information

Bugcrowd Inc. is a company that is focused on offering services on the internet concerning application penetration tests. The company was founded in 2012 and became the most recognized name offering help with bug bounty programs. According to their website, they have currently, more than 8900 registered people who test websites for security flaws. [13] Companies that decide to let their websites be tested by the community can contact Bugcrowd for support. The objective for a test can either be a website or a mobile, tablet or desktop application. Additionally, it is possible to only inform a specific range of penetration testers (e.g. country-based, skill-based, etc.). Bugcrowd offers a control platform as the test is running which informs the owner of the tested software about the parts of the software currently being tested and the number of submissions that already have been made. This procedure gives a brief overview about lines of code that may be affected by security flaws. To save time for the operators, Bugcrowd also offers the possibility to find and remove duplicate submissions, in case different testers submitted the same vulnerability.

Different incentives and offer of bounties

Bugcrowd Inc. demands registration of testers before they receive all the information needed to search security flaws. As soon as a tester is registered, he is able to login and view the currently running penetration tests shown which are available for his country and skills. At the same time, it is visible for how long the test is running. Bugcrowd offers two kinds of timeframes in which the penetration tests can be executed. First, there are ongoing tests telling the user no information about the duration of the test. Second, there are tests limited in time, which are just open to penetrate for a short period. Next to the separation concerning time, there is also a difference in rewards offered by Bugcrowd. Kudos points are one kind of reward a tester can get. These points are used to show the skill level in finding bugs in an internally managed list. They also serve as the only unit to see the reputation of a tester. With many points, high qualification is proved. With a high rank, information for special penetration tests that are not visible for other users can be obtained. A second possibility to be rewarded is with money. Depending on the company tendering the test, the amount of money for different kinds of vulnerabilities fluctuates. In case a finding is getting a monetary reward, the user also gets Kudos points.

3.2.3 Different Approaches

Without using the special service of Bugcrowd Inc., there are other possibilities to execute penetration tests on a bug bounty basis. A good way to search for bounties is to use a web search engine, searching for the terms "bug bounty" or "bounty program". Nevertheless, one of the first results will be Bugcrowd again because they offer a list with many entries of companies that are offering bounties independently (This list can be found at https://bugcrowd.com/list-of-bug-bounty-programs/). A further example of a list of bounties is Bugsheet (http://www.bugsheet.com/bug-bounties). Additionally, there are a lot of

blogs and websites operated by security researchers also offering some information about bounties currently running.

All these companies offering bounties that can be found by reading these lists have in common that they manage their vulnerability disclosure programs by themselves. Therefore, it is important to read the conditions carefully as they vary from one bounty program to another.

3.3 Environment and Tools

This paragraph gives a short overview about the tools and the applications used to perform the penetration tests.

3.3.1 Kali Linux

Kali Linux is an operating system based on the Linux Kernel and the successor of the BackTrack operating system. This system was designed to deliver all the tools needed for security analyses. The image to install the operating system is being further developed every day and is downloadable free at http://www.kali.org/downloads/.

Kali Linux offers several pre-installed tools covering following sections:

- Information Gathering
- Vulnerability Analysis
- Web-Applications
- Password-Attacks
- Wireless-Attacks
- Exploitation-Tools
- Sniffing and Spoofing
- Establishing Access
- Reverse Engineering
- Fuzzing
- Hardware-Hacking
- Forensics
- Reporting

Next to this functionality, the system offers even more tools that modern operating systems normally have included (e.g. data processing, multimedia, etc.), but these are not relevant for this thesis.

3.3.2 Windows

The Microsoft Windows operating system is used to host the Linux as virtual operating system and to run different web browsers that are not executable on Kali Linux.

3.3.3 Web Browser

The web applications that are analysed in this work run in web browsers. These applications display the content and deliver the functionality of web applications to a user. Multiple browsers are available which differentiate themselves in design aspects, functionality, expandability and more. For penetration tests, it is important to know that web applications can produce different output in different browsers. Therefore, a possible vulnerability can still exist although it was checked in a specific browser. Hence, it is important for security analyses to check possible flaws in multiple browsers. A good way to select the browsers being used is to research about the distribution of usage. Regarding that aspect, following browsers are in use for this work (Statistics are provided by w3schools.com [14]):

- Google Chrome: With approximately 55 percent, Google Chrome is the most used Web Browser in January 2014 and since the launch of this application, the number of users has been continuously rising. This browser is pre-installed on all new Android devices.
- Mozilla Firefox: This web browser has been the most used from 2009 to the beginning of 2012 and has still a usage of 27 percent.
- Microsoft Internet Explorer: As Microsoft Windows operating system is the most used, most internet users have Internet Explorer installed because this application is pre-installed on every Windows distribution. Nevertheless, only 10 percent have been using this application to browse in January 2014.
- Iceweasel: Without any significant usage, Iceweasel is in use for this thesis because it is the browser that comes with Kali Linux.

3.3.4 Burp Proxy

"*Burp Suite is an integrated platform for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploiting security vulnerabilities. Burp gives you full control, letting you combine advanced manual techniques with state-of-the-art automation, to make your work faster, more effective, and more fun.* (italics added)" [15]

The Burp web-intercepting proxy is the main tool used for the execution of penetration tests in the context of this thesis. It is available in a free and in a premium edition, whereas the free edition is sufficient for the most tasks of a security analysis. The professional edition has some helpful features implemented like an automated vulnerability scanner, an extensive search function, prepared lists with injection patterns and many more extensions that are useful. For this thesis, the free version of Burp was used due to cost constraints.

The following list provides an overview of the main functionality of Burp:

- Target Tab – Lists all the directly and indirectly visited websites in a tree view and offers the possibility to define a specific scope to intercept only requests from desired domains.
- Proxy Tab – The main functionality is offered by this tab, which is the possibility to intercept outgoing HTTP requests in order to be able to easily spot injection points. These can be manipulated to get different behaviour and responses by the web application. Additionally, it offers a history functionality that saves all intercepted requests to be able to reconstruct sent data and received responses at a future date.
- Spider Tab – This implemented functionality offers a way to crawl a selected website for all pages available, which can be used to quickly see the scale of a web application.
- Scanner Tab – Only in the professional edition available, the scanner is an automated tool searching for vulnerabilities by inserting well know patterns to test various security flaws in web applications. It is also possible to get an edited document as output, describing the findings by the scanner.
- Intruder Tab – This tab enables a tester to use a request and mark specific spots to insert payloads (e.g. typical XSS-attacking-vectors) to automate the search for vulnerabilities.
- Repeater Tab – Similar to the intruder tab, it is possible to send one single request repeatedly with different parameters. The repeater is useful to penetrate the web server with handcrafted payloads manually.
- Sequencer Tab – This feature makes it possible to analyse the randomness of tokens and cookies. The sequencer collects tokens out of responses and compares them with all the previous collected ones.
- Decoder Tab – This small extension gives a tester the opportunity to quickly and especially offline convert data using different en- and decoding.
- Comparer Tab – A useful tool to compare requests or responses concerning their content as it is sometimes not easy to see differences immediately.

Additionally to all these implemented features, Burp offers a lot of options to change the behaviour of the tool regarding HTTP and HTTPs traffic, sessions and the display of the tool itself.

3.3.5 Helping Tools

Although Burp can test almost everything, there are still some scenarios in which other tools are more convenient to use and deliver better results or in which Burp is not able to execute the test. In this case, it is better to make use of other applications.

Dirbuster

Dirbuster is an application that searches for webpages that are not directly linked from the origin website itself. Often web applications have pages which are still in development or should not be used anymore, but which are available on the webserver. In order to find these webpages, this application offers the possibility to send requests using dictionaries with common internet terminology that are added to the original URL. The response code of the webserver that is contacted delivers information if the requested link is available or not. With this information, it is possible to find incomplete webpages that have a higher probability to have security flaws.

Qualys SSL Labs

Modern web applications should make use of Transport Layer Security (TLS) [16] while transferring sensible data. Since the introduction of this technology, there have been different versions with different encryption mechanisms. Several combinations are deprecated and should not be used anymore. Qualys SSL Labs is a web application that produces a detailed report of supported cipher suites by a given website and informs the user if the website is secure or not. The report can be created free and the web application can be found at https://www.ssllabs.com/ssltest/index.html.

3.4 Project Execution

The penetration tests executed for this thesis were performed in several steps. Every single test was following the same procedure. The three main parts (information gathering, testing, submission of findings) are described in the paragraphs 3.4.1 to 3.4.3. Before the actual security analysis was executed, the bug bounty, which should be analysed, had to be chosen. The selection was mainly based on the date the bounty was released. Only new bounties have been chosen and mobile, tablet and client applications were excluded. A closer look at the web applications can be found in section 3.5.

3.4.1 Information Gathering

Information Gathering is important before beginning the testing as it gives the tester the first impression of what he is dealing with. In both, a manual and an automated way, it is helpful to explore the application before using tools to find security flaws. Although it is not part of the gathering, it could be already possible to find vulnerabilities. At the end of this step, the security researcher should be aware of the general functionality of the website and the techniques used to program the software. The tester should already have first ideas of possible attacking vectors. [17]

The first step in an analysis was to browse through the web application by simply clicking from one link to another to obtain the following information:

- Purpose – What does the application aim for? What is the target group to use the software?
- Scale – How many different pages does the application approximately have? Are there many injection points?
- Registering – In most cases an account is necessary to test the full functionality. Therefore, it is important to create as many dummy accounts as necessary in the beginning.

The second step was using the automated Spider functionality of Burp to get a full overview of all available links that are reachable. Burp, thereby can start from the index page of a web application and tries to open all references available on a website. A major drawback of this automated procedure is that not all web links can be found if the website uses many forms as these often just can be submitted if all values are provided in the right format. Therefore, it was still necessary to click through all the forms available manually to enable the spider to continue its search.

Web applications often use hidden form fields as well to transmit important content. During the penetration tests for this thesis, a functionality of Burp was used to unmask hidden form fields in order to find additional injection points. While intercepting requests sent by the application, it was also possible to detect and manipulate available hidden form fields.

As not all available web pages on the webserver are directly linked, it is also necessary to search for unlinked ones. A good way to find these is to use special dictionaries that are used in combination with the tested domain. An application called Dirbuster was used for this purpose, which is described in section 3.3.5.

In the time the automated search for unlinked pages is running, it is also a good alternative to manually search for unlinked sites via web search engines with special patterns that these engines offer with their extended search.

Additionally to these different approaches, it was beneficial to check the application again with Javascript being disabled and without allowing cookies. A reason therefore is that the browser cannot display the software in its origin state, which may lead to different output. In this different representation, it is again necessary to check with common penetration testing techniques for software vulnerabilities.

Normally, an important check is to look at the infrastructure (e.g. software of the webserver) to get names and version numbers in order to check if there are well-known exploits and vulnerabilities available. In the sites tested in the context of a bug bounty program, it was excluded from the scope; hence, no steps were taken to check the infrastructure.

3.4.2 Testing

The process of a penetration test contains several different tests to find web application vulnerabilities. In the following points, the most important checks are described including a short introduction, how to search for these flaws. (Note: the following description is based on the execution of testing for specific vulnerabilities used in this thesis. There are other possibilities to search for security flaws.)

Cross-Site-Scripting (XSS)

"*Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.*

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. (italics added)" [18]

To search for XSS vulnerabilities, the main tool used is the Burp Intruder, which was described earlier. There are different kinds of XSS vulnerabilities. The first possibility is stored XSS, whereby the attacker places some script code in the request and this input gets shown at a specific spot in the application without being correctly sanitized. After the information gathering process, a tester should already know where this vulnerability could occur. With prepared lists of attack patterns and the intruder, it is possible to run the test automatically and watch afterwards if there was a pattern that was interpreted as a script.

A similar vulnerability is reflected XSS, where the input is not shown anywhere in the web application. Nevertheless, it could be possible that some script codes transmitted by the tester are correctly interpreted but only shown in pop-up windows. [17] To find this kind of XSS flaws, it takes more investigation as the automated intruder does not render the output during the screening. Therefore, it is necessary to discover the server responses manually. There are still ways to make it easier to find these flaws by extracting specific parts of the response and displaying them immediately to see how the input was interpreted. Subsequently, the pattern that seemed to be successful should be repeated using the browser instead of Burp to check if the vulnerability is valid.

Additionally, there have been a few manual checks made on every single input field trying to circumvent the sanitation mechanisms of the web application. Hereby, the process is to start with several initial attack patterns and look at the way different characters are being processed. The aim is to adapt the input string in order to be interpreted as script code by

the webserver.

There are further kinds of Cross-Site-Scripting vulnerabilities (e.g. DOM-based XSS) which are not as common as the two mentioned and therefore, the web applications in this thesis have not been checked for those due to time constraints.

SQL-Injection

Typically, web applications run different kinds of databases in the background to store important information, which is necessary for a dynamic operation with a user. Databases are used, for example to display content according to specific search patterns (e.g. web shops) or to store new information given by a user to remember the data for later use. Although these data storages are highly necessary, it has to be ensured to provide a secure implementation if a regular user must have the opportunity to operate with a given database. In this case, it always could be possible that the user is an attacker aiming to get more and highly sensitive information than the originally displayed one. Databases are using a syntax, titled "Structured Query Language". Different database providers use different syntaxes, which means that an attacker has to be aware of the technology used in order to use the right syntax for the attacks. These different behaviours can easily be understood by installing various databases on a local storage to study the syntax. Additionally, in case of not knowing the database in use, there is still a high probability of guessing the right one as most website operators rely on the most common ones (e.g. Oracle, MS-SQL, MySQL). [ ]

The approach to find SQL vulnerabilities used was to search first for requests to the webserver, which triggered database queries. After spotting them, a guess was made about the probable data type (e.g. string, numeric, etc.). After determining the possible value format, several tries to create a different behaviour then the usual one were made (e.g. " ' or $1 = 1$ - -" to break out of a string and leave the rest of the query as a comment, using the + operator to calculate the numeric value instead of directly typing it, etc.). In case a functional query was found or error messages appeared, further investigation was necessary to check if a valid injection vulnerability was available. At this point, it was helpful again to use the Burp Intruder with a list of well-known SQL-Injection patterns to check for a possible exploitation of the database query. As these tests were running, further manual checks were made.

To ease this process, another tool is downloadable for free that has extensive functionality for checking SQL-Injection flaws which is named SQL-Map (The tool can be found at http://sqlmap.org/). Using Kali Linux, this application is pre-installed, allows to footprint the database and execute various operations on it with the possibility to select the desired queries precisely used in the automated tests.

Caution: As a penetration tester does not want to harm the operator of a web application, it is especially important to be prudent with functional SQL-Injection vulnerabilities. In case a production system is tested, it is not advisable to use queries that can delete or modify production data in a certain manner.

Session Cookie Strength

Cookies are used in web applications to track a user due to various reasons. A necessary cookie is the session cookie that enables a user to stay logged in while browsing from link to link. As it would not be user-friendly to log in for every single page, the webserver returns a value that is transmitted with every single request to confirm the validity of the user. If this returned value is not strong enough or easy to guess (e.g. cookie includes obvious patterns like hashed birthday, name, etc.), an attacker could acquire possession of a valid cookie of users being logged in. A personification of a victim would be possible. [17]

To check the randomness and the length of a cookie, Burp Sequencer offers an easy way to obtain numerous cookies in order to compare them. The effective entropy is analysed and shown with graphs for every single bit. The check for specific patterns in session cookies however is not easy. To find these, it takes a long time of manually thinking about the details that could have been integrated in the cookie structure and many attempts to reconstruct a similar looking session identifier. A proper support for this purpose does not exist using Burp.

Leveraging Client-Side Checks

Browsing through the internet and using web applications always needs a consistent communication between a client and a webserver. The reason for outsourcing information processing and validation of input is, for example to accelerate the processing of data to enable the user a better experience. Different implementations are available, such as storing data in cookies and headers, using browser extensions (e.g. Java, Flash, Silverlight, etc.), script code (e.g. Javascript) and many more. These technologies can also be used to secure a website by using them to check client input. A common mistake of applications is to trust in the data being received by a user. This leads to big opportunities on the side of attackers. [17]

The following overview provides information about client-side controls that have been checked for flaws during the penetration tests for this thesis: [17]

- Hidden Form Fields – Web applications often store important information in hidden fields, which are not visible using the web browser. Nevertheless, these fields can be manipulated by using an intercepting proxy or by using browser extensions that automatically delete the string "type=hidden" in form fields from receiving HTTP responses.
- Disabled Form Fields – Fields that are visible but greyed out are disabled, which means that they are not sent by submitting the form. By changing the HTTP response, it is possible to enable them through deleting the "disabled" attribute. Thus, an attacker obtains a possible chance to exploit the software.
- HTTP Cookies – Important information can be stored in cookies. Cookies again can be manipulated. A good way to prevent fraudulent use is not to transmit them in plaintext. Further, in case of hashed cookies, it should not be obvious which information

was hashed to prevent from replication with crafted values.

- URL Parameters – Using HTTP Get requests, all information sent to a webserver is displayed in the URL bar of a browser. Sometimes (e.g. pop-up windows), the bar is suppressed which leads to ignorance of security. These parameters can still be influenced by using an intercepting proxy.
- Form Restrictions – While submitting forms; it often happens that the web application does not allow submitting it in its present appearance due to various reasons (e.g. length-limits, disallowed characters, etc.). These validations can be leveraged by submitting a valid form, manipulating the data afterwards using an intercepting proxy again.

These mentioned breaches give a good example, why it is still necessary to check every single request, incoming from a user, on the server side to be sure that the application is not compromised.

Authentication

Authentication mechanisms are highly important for the use of web applications. In order to grant every single user his own individual profile with sensitive information, the software has to ensure that only the registered person himself can enter the account. The most common procedure is a login mask demanding valid user credentials to continue. Several tests were made to analyse the software regarding authentication security:

- Login Bruteforce – A bruteforce attack is a technique, where you try to guess either the username and password or just the password if you have a valid username. The attack is automated using either random words with a specified length and charset or dictionaries with common passwords and mutations of these, which is the better approach. The main test is to check if or after how many tries the account is locked. To get valid credentials is just optional.
- Password Quality – An important role to exacerbate brute force attacks is to use strong passwords, which have to be enforced by the web application. Passwords should be checked if they fulfil following requirements: at least eight characters and 3 out of 4 security criteria (upper/lowercase characters, digits, special characters).
- Login Error Messages – During the login, an application should not provide information about which part of the credentials have been wrong in order not to disclose data (e.g. existing usernames). In case of a failure, the application should respond telling that the login failed without disclosing reasons. The same procedure should be used for giving information about password resets.
- Forgotten Password – When a user forgets his password, most web applications offer a possibility to reset it. Here, it must be checked, how the software sends a new password. Bad implementations like sending a new one in plaintext, should be avoided and replaced by sending links with tokens, only valid for a short amount of time.

These previous described checks are suggestive for every single web application. Nevertheless, there are a lot more possibilities to attack authentication, which depends on the specific implementation of the software. Hence, further attack scenarios are not described.

Further Tests

All previous explained attacking vectors are just an extract of all checks made during the penetration tests. Additionally there have been tests made focusing the application logic of each application. Further tests have taken checking the access controls to various functionality of the software. Additional architecture components only have been tested for known flaws if any version information was available. Apart from that, no more checks have been made due to limited time.

3.4.3 Submission of Findings

All the submissions made, have been for penetration tests that have been part of the testing program operated by Bugcrowd. After registering and login, a tester receives the opportunity to submit a finding by clicking on the submissions link in the menu. Bugcrowd demands a lot of information for each individual finding in order to be validated as shown below:

- Bounty [Required] – Drop-down list with all currently running bounties is given. The tester has to choose the one for which he claims to get a reward.
- Caption [Required] – A short name for the finding is required to identify the submission.
- Sort [Required] – A radio button list offers a few categories in order to roughly classify the finding (e.g. XSS, CSRF, SQL-Injection, etc.).
- URL – The correct URL has to be stated in its full length to make it easier to understand for the validator at which point the flaw was detected.
- Affected Users [Required] – A tester has to inform if the flaw affects all users or just authenticated ones.
- Description – A general statement about the software vulnerability should be given to make it more understandable.
- Trace Dump/HTTP request – Tracing information occurring because of programming exceptions or HTTP requests, which got intercepted while finding the software flaw, should get submitted. This helps to prove and reconstruct the finding.
- Affected Parameters – In case a specific parameter in a request is affected and causing a vulnerability, it should be indicated.
- Steps to Replicate [Required] – A detailed description, how it was possible to exploit a given flaw should be written down in order to make it easier for the person who has

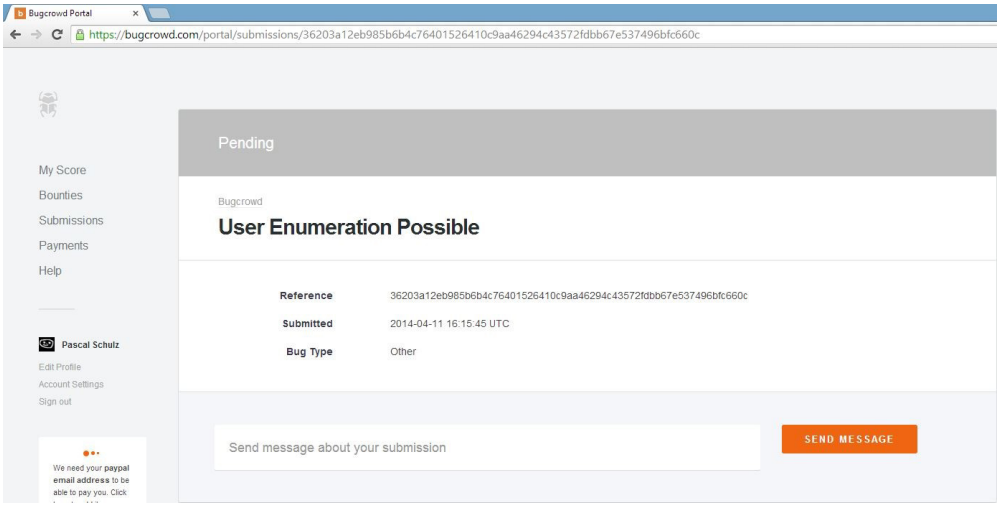

Figure 3: State of submission in user backend

to validate the finding. Here, it is better to give as much information as possible as this is the main part describing the whole finding.

- Additional Information – Any kind of helpful information can be written down in this section.
- How was the bug found – A short declaration should be given if the flaw was found either using a specific tool or by manually researching on the application.
- Tool – In case a tool was used that was helpful to find the software bug, it should be stated.
- Attach files – To reinforce the proof of a finding, it helps to make screenshots of the found vulnerability (e.g. a screenshot showing a pop-up window which was caused by a XSS-attack).

After successful submissions, Bugcrowd sends a confirmation mail to the user with a given reference number for each individual submission. After this point, the submission is not accessible anymore. The user backend is just listing all the findings and provides information about the actual state as shown in figure 3.

In case a penetration tester has some extra information or finds an additional flaw belonging to the same finding, it is possible to send further messages about a specific submission. The average time for an answer, if a submission is valid, during the entire penetration tests executed in this thesis, was between 20 and 30 days. Bugcrowd informs the user via email about the date when all submissions are validated. In case the bug bounty was offering money for submitting findings, Bugcrowd transfers the amount to the PayPal account of a user.

3.5 Penetration Tests

In the context of this paper, three penetration tests on different web applications have been executed. In this section, the purpose of these websites is shortly presented including an overview of the functionality of these applications. The result of the penetration tests can be found in Chapter 4.

3.5.1 Magento eCommerce

"*Build your business with the eCommerce platform that puts you in control.*

Magento offers flexible, scalable eCommerce solutions designed to help you grow and succeed online. Our cost-effective technology platform makes it possible for you to control the content, functionality, look and feel of your online store.

We also offer support, services and training to help ensure your success. Our network of partners and certified developers offer expertise and resources to help design, build and host your store. And partners also offer thousands of apps and extensions to help you add custom features and functionality. (italics added)" [19]

As already stated in the official statement about the company, Magento is a software implementing online shops that can be used by website operators to easily setup their own market place. The company was bought by Ebay Inc., which is known for managing auction sales online. The particular version tested for this thesis, was the Magento Go software that is frequently used by more than 200.000 users. [19]

For the bug bounty, there was a special setup, which was available at https://hackme.io. At this uniform resource locater, there was a trial shop installed with a few trial products in different categories. After the bug bounty, the website was erased. The bounty briefing requested that any found bug had to be able to be replicated on previous mentioned website. The trial shop was structured in an area for regular users and in an area for registered users. For a user without valid login credentials, it just was possible to view all the different goods in different categories in the store. A search function was available to discover the contents for specific keywords and a sitemap to get a quick overview.

After registration, there was following functionality available:

- Account Dashboard – Overview of recent activity and account information
- Account Information – Display of all user information with the possibility to change user parameters and password
- Address Book – Management of billing and shipping addresses
- My Orders – Review of ordered products
- Billing Agreements – Payment arrangements are shown

- Recurring Profiles – Establishment of default profile
- My Product Reviews – A listing is shown of all reviews made to specific products after the operator has checked them
- My Wishlist – Registered users can put products on a wish list to remember them for a delayed purchase
- My Applications – Reclamations that have been sent can be checked here
- Newsletter Subscriptions – Display of ordered newsletters
- My Downloadable Products – Possibility to download non-physical products (e.g. songs, movies, etc.)

3.5.2 Bugcrowd.com

The company Bugcrowd and its business model has already been described in previous sections. In the list of bounties offered for registered users, they also have their own web application offered to be reviewed by all of their users. The frontend just has several pages of static text. Hence, the focus was set on testing the user backend, which has the following sections:

- My Score – This page lists multiple rankings that are internally administrated by Bugcrowd, including various achievements.
- Bounties – The list of all bounties and the administrative description of Bugcrowd are given on this page.
- Submissions – The submission status of all findings ever handed-in including their validation status can be found here.
- Payments – The payment history, how much money has been rewarded for finding bugs, can be found in this section.
- Help – A direct link to the email address "support@bugcrowd.com" is given for any questions according the portal.
- Edit Profile – Personal details can be changed on this page, including an avatar, contact details and biography.
- Account Settings – Password settings, email address and paypal payment information can be adapted here.

The penetration test was executed using the real user account that is used for bugcrowd.com in order to get information about the bounties.

3.5.3 Heroku

Heroku is a Platform-as-a-Service (PaaS) provider, specialised on the following points as stated by their website:

"*Heroku (pronounced her-OH-koo) is a cloud application platform – a new way of building and deploying web apps.*

Our service lets app developers spend their time on their application code, not managing servers, deployment, ongoing operations, or scaling. [...]

Heroku was founded in 2007 by Orion Henry, James Lindenbaum, and Adam Wiggins. (italics added)" [20]

The scope of the penetration test included several subdomains of heroku.com, with the following main functionality regarding the backend:

- Dashboard: Overview of created apps with a search functionality
- Databases: Overview of all existing databases with the possibility to create new ones
- Add-ons: Listing of all existing extensions which can simply be added to the application in progress
- Docs: Extensive documentation page, giving information about building, deployment and management of the web applications.
- Support: Help-page of Heroku in case of any questions
- Account: Management of the account details is possible

3.6 Chapter Summary

Web applications often have many security vulnerabilities as software developers are often not aware enough of implementing sufficient mechanisms preventing from misuse by attackers. Therefore, a new approach to get websites tested is existing by letting the community analyse the software. These tests are well known by the term of a "bug bounty". An American company that is specialized on offering bug bounty services is Bugcrowd Inc. This company has engaged a large community of penetration testers all around the world, who receive information of new bounties through the web platform, including special incentives for executing these tests. As motivation, all the testers have monetary rewards in prospect or the possibility to earn reputation to obtain special invitations to bug bounties.

For the tests executed for this thesis, a testing environment was set up consisting of a Linux and a Windows operating system. The main tool used to perform all the tests was Burp, which is mainly an HTTP request intercepting proxy, giving the tester the possibility to manipulate the information simply that is sent to a webserver. As different web browsers

display web applications not in the same way, the most common used browsers have been used to analyse the applications. Additionally, further security analysis tools have been used to evaluate the website security of the tested bounties.

In order to understand how penetration tests are executed and which steps are taken to disclose software breaches, an overview of the most important hacking techniques is given. Beginning with Cross-Site-Scripting attacks, where it is possible to let the webserver execute user created script code, to attacks aiming at the authentication procedures, an insight is shown. Furthermore, the process, how findings are being submitted with the use of Bugcrowd, is stated in detail, listing all the necessary information.

Last, all the tested websites are briefly introduced, offering an overview of their main functionality and a short statement covering their purpose and their target group.

4 Results

For the thesis, three different websites have been the target of a thorough security analysis. In the following paragraphs, the findings are described including the statement given by Bugcrowd Inc. about the validity and the progress with the submitted information. Additional explanation and figures, especially about techniques used, are given in the appendix.

4.1 Findings

The following findings are briefly described with information given about how it was possible to exploit a specific vulnerability. Additionally, a short statement is given, how the flaw could be avoided and which impact could appear due to the existing security breach. Bugcrowd gives the following validation states (including their meaning):

- valid: The finding gets credited and the software flaw will be patched.
- invalid: The finding is either not severe enough, not qualified according to the terms and conditions or is not correct.
- duplicate: The finding has already been submitted by another security researcher and gets only credited with the minimum amount of points.

4.1.1 Magento eCommerce

For Magento eCommerce, which is owned by eBay Inc., six findings have been submitted. Two submissions were validated as correct findings that were rewarded by Bugcrowd Inc.

Newsletter Flooding (status: valid)

The website offered a form field, with which it was possible to subscribe for a newsletter. After probing the field with an email address that is in possession of the tester, it was possible to see that the newsletter was effectively arriving. There was no check made, if the person who sent the request, is really the owner of the email address. Without having a user registered, it was possible to subscribe various email addresses to this newsletter, as it can be seen in figure 4.

This security flaw was exploitable for all users and could have multiple impacts. First, it is possible to spam the newsletter to arbitrary people all over the world. Second, with this possibility it is possible to cause a denial of service of the mailserver. A good approach to patch this flaw would be to enable just registered users to subscribe to the newsletter with their own email address used for the registration mechanism. This finding was rewarded with a monetary bounty and 5 points for the internally managed skill ranking.

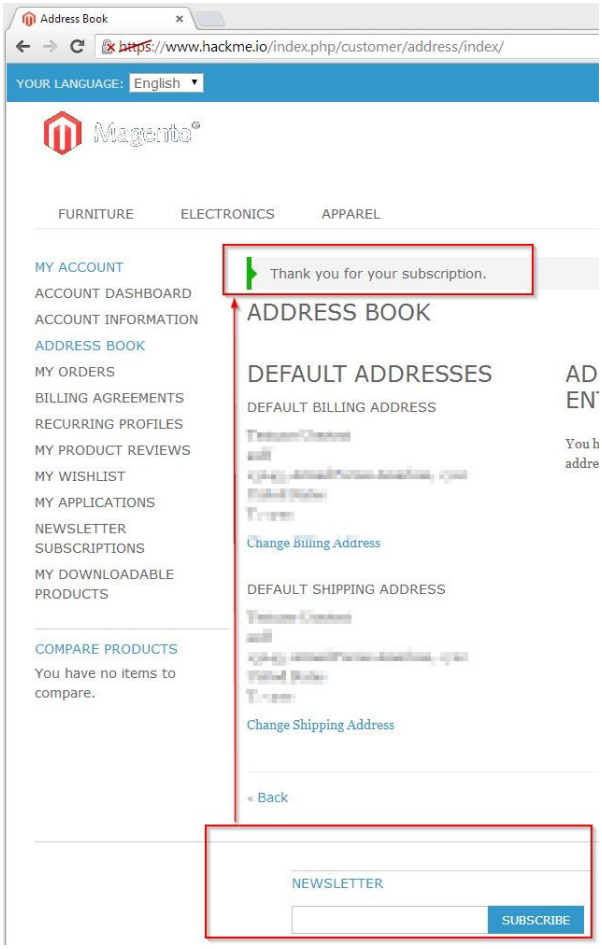

Figure 4: Newsletter flooding possible

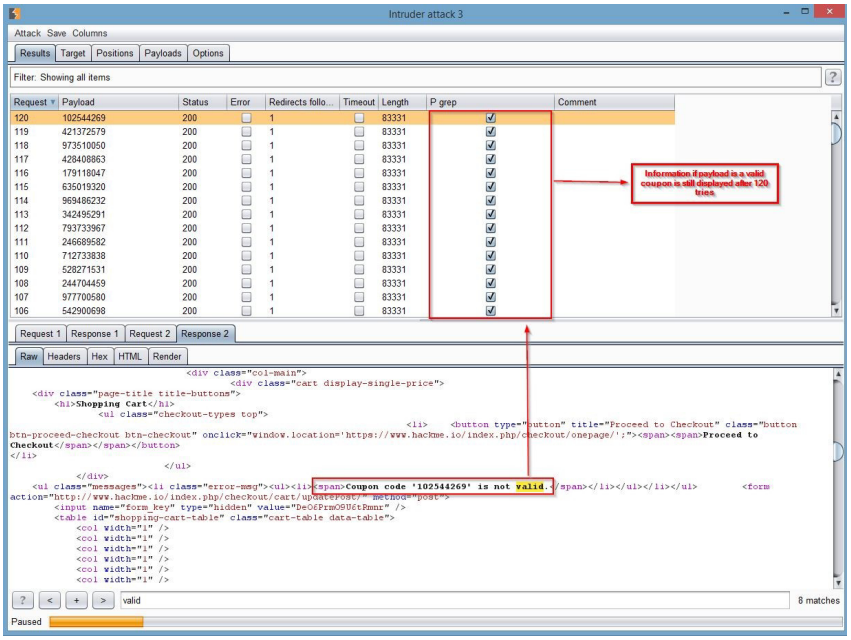

Figure 5: Guessing valid coupon codes

Coupon Brute Force (status: valid)

The market software offered the possibility to enter a discount coupon during the checkout in order to save money. When a false coupon code was transmitted, the webserver answered with a message telling the user that the code was not valid. As there was no limitation, how often a user could try codes, it was possible to use a tool that automated these requests in order to find a valid coupon number. An improvement of this technique would be to obtain a valid code (e.g. by buying a credit to use it as a present) to get to know about the structure and the length of these codes. The brute force progress is shown in figure 5.

If an attacker is able to find valid coupon codes, the company loses a lot of money. Additionally, this code normally belongs to somebody else, who has the code through various reasons. When this user later tries to use this specific code, the coupon will not be valid anymore, which leads to many complaints of regular users. A good way to avoid this attack would be to throttle the possibility to enter coupon codes as normally a user does not need more than about five tries to enter the code correctly. This finding was rewarded with a monetary bounty and 5 points for the internally managed skill ranking.

User Enumeration Possible (status: invalid)

The account edit functionality was offering an opportunity to change the email address of the user with an unlimited quantity. With every change, the web application was telling the user if this address was existing. With this information, it would reduce the work to find valid login credentials. Nevertheless, this finding was not credited, receiving following statement by Bugcrowd Inc.: "*At this stage we do not consider this attack a security threat to our application, as outlined in the bounty terms and conditions under the Rules->exclusions section. Exclusion: Descriptive error messages (unless it can be demonstrated in an exploit/chained-exploit) Unfortunately, at this stage your issue has not qualified for a bounty reward, […]*".

Unrestricted Guesses of "Current Password" field is allowed (status: invalid)

The password change functionality required the user to enter the current one to change it, which is a good implementation. Nevertheless, it was possible to guess the current password any number of times, which enabled a brute force attack (for example, if a user is logged in and forgets to lock his screen). This finding was not credited, giving the same statement as in the previous finding.

Directory Listings (status: invalid)

Several listings of directories stored on the webserver were found containing multiple data entries. An attack could not be produced using this data, which was the reason why the finding was declined with following words: "*At this stage we do not consider this attack a security threat to our application. Not a security issue. Unfortunately, at this stage your issue has not qualified for a bounty reward, […]*". Nevertheless, it is not necessary to offer directory listings to regular users.

Password Bruteforce Possible (status: invalid)

At the log in mask, it was possible to try any number of times to guess a password for a given username. Normally, a good way would be to throttle the number of tries to a small number and lock the account for a short amount of time after several wrong tries. Nevertheless, this finding was declined with following statement: "*At this stage we do not consider this attack a security threat to our application, as outlined in the bounty terms and conditions under the Rules->exclusions section. Exclusion: Descriptive error messages (unless it can be demonstrated in an exploit/chained-exploit) Exclusion: Account lockout attacks (unless it can be clearly demonstrated on a dummy account. Mass account lockout constitutes a Denial of Service attack and is strictly forbidden, as is using a real account that does not belong to you or is not specified as permitted under 'targets') Unfortunately, at this stage your issue has not qualified for a bounty reward, […]*".

4.1.2 Bugcrowd.com

For bugcrowd.com that is also the platform offering the bug bounties, four findings have been submitted. All of these were validated as correct findings, which were rewarded by Bugcrowd Inc., albeit one submission was a duplicate as another security researcher had found the same vulnerability before.

Circumvent Password Rules (status: valid)

During the registration of a new user or the password change functionality for an existing user, it was necessary to type in at least eight characters to fulfil the password requirements. However, it was possible to reduce the effective length of a password to just one character by submitting a password existing of at least eight characters, but including the NUL control character as second letter. After logging out and try to log in again with just one character, it was possible to see that the change was successfully made. The necessary request is shown in figure 6, whereas Burp displays the control character as a rectangle.

Although the user was just able to reduce his own password quality, this security flaw was dangerous as a password with a length of one character can be guessed within one minute, which could have led to an impersonation of this user. The webserver should sanitize special characters that are not valid for usage in a password to avoid a security flaw like this. This finding was rewarded with a monetary bounty and 5 points for the internally managed skill ranking.

Password Quality (status: valid)

The required password quality for bugcrowd.com was given by 8 characters, allowing a user to just use lowercase characters. State-of-the-art passwords should contain at least 3 security criteria out of following:

- Lowercase characters (a,b,c, …)
- Uppercase characters (A,B,C, …)
- Digits (1,2,3, …)
- Special characters (!,",§, …)

Regular users tend to choose a password with the minimum requirement, which leads to a high number of people having weak passwords. These passwords can be brute forced relatively easy compared with a password, fulfilling the above stated security criteria. This finding was accounted as valid, but Bugcrowd Inc. stated that this weak requirement set is on purpose. The submission was still rewarded with 2 points for the internally managed skill ranking.

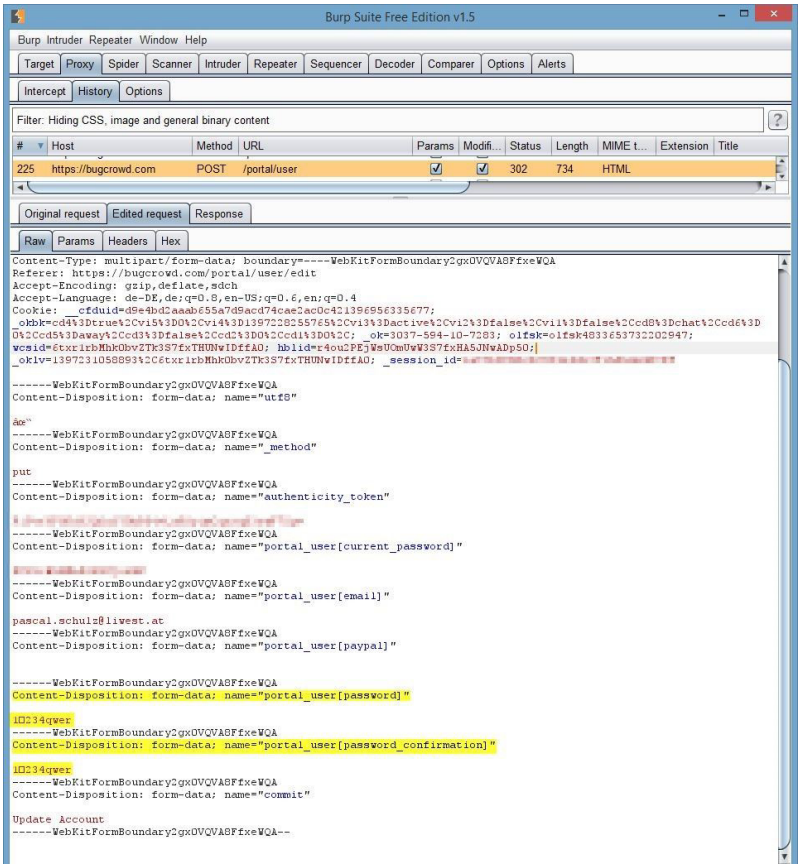

Figure 6: Tricking the password change functionality

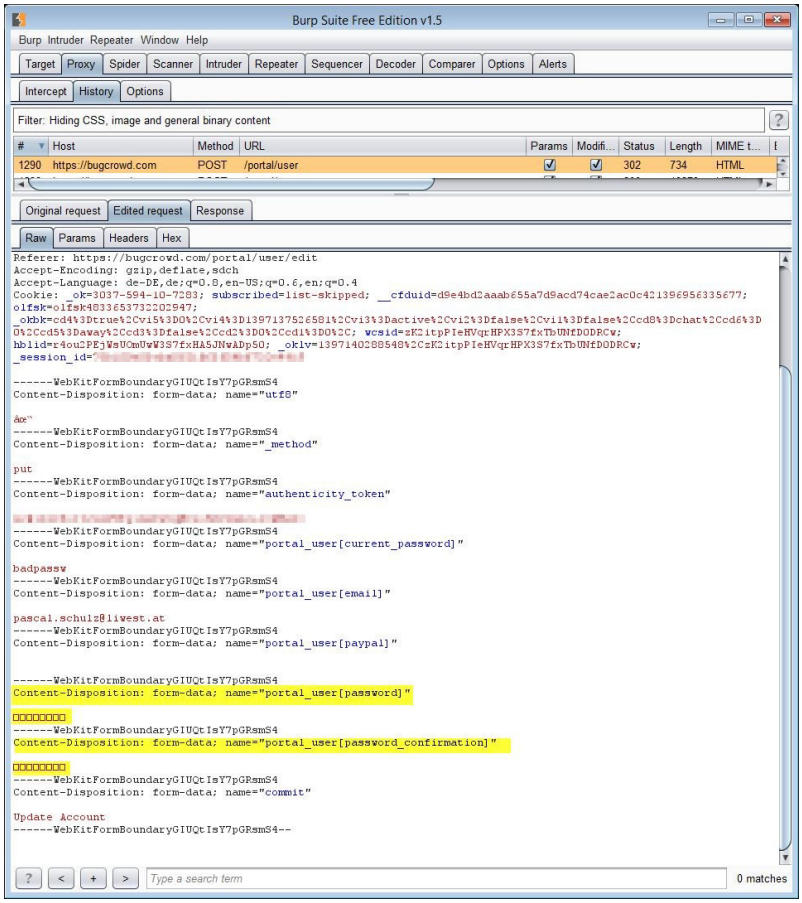

Figure 7: Attempt to log in with 8 control characters

Password Lockout (status: valid)

During the registration of a new user or the password change functionality for an existing user, it was necessary to type in at least eight characters to fulfil the password requirements. However, it was possible to enter eight control characters (e.g. NUL control character) to fulfil the requirements. After a new try to login with these control characters as new password, it was not possible anymore to log in. The request to log in is shown in figure 7.

The only way for a user to log in again was given by using the password reset functionality to obtain a new one given by the webserver. The finding was rewarded with 2 points for the internally managed skill ranking with the following statement: "*Wow, this is pretty technical. Given a user would have to go out of their way to do this to themselves we won't be fixing this*

bug. Plus they could always do a password reset. Other than that this is an excellent write up, given that it is reproducible but won't be fixed we are still awarding you 2 points. We'd love to see some bugs that could be used to comprise/attack the system. [...]"

User Enumeration Possible (status: duplicate)

During the registration process, an email address had to be provided which was used as username and for sending an activation link to prevent numerous fake accounts. During the submission of the form, the webserver responded telling the user if the mail address had already been in use. With this behaviour, it was possible for an attacker to enumerate existing users by sending a list of prepared addresses to the server, waiting for the response. With a valid username, it is becoming easier to brute force the login procedure. This finding was rewarded with 2 points for the internally managed skill ranking but was stated as duplicate as somebody else had found this flaw before.

4.1.3 Heroku

For Heroku, which is a platform-as-a-service provider, two findings have been submitted. One submission was evaluated as valid and the other finding as duplicate.

Password Flaws (status: valid)

The Heroku password requirement stated that a valid password had to exist out of at least eight letters, numbers and symbols. During several attempts to change the password into various weak combinations, it could be seen that this password quality rule was not enforced at all. It was only checked that the password consists of eight characters. Additionally, the application allowed a user to use the old password as his new password again, which should be prohibited. A good approach would be to disallow the user to use the last five chosen passwords. This finding was rewarded with 2 points for the internally managed skill ranking.

Login without Password Possible (status: duplicate)

The password change functionality of the account management required a user to enter a minimum of eight letters, numbers and symbols. While intercepting this change request, it was possible to put a NUL control character in front of the completely new password line. This led to a valid new password, where everything was cut off, beginning with the control character. In the test, it has been proven that with this security flaw, it was possible to create an account that has no password at all. The login request is shown is figure 8.

Providing the possibility for a user to have an existing account without having a password set, is a high risk for the user himself. If an attacker finds a valid username that uses no password to authenticate, the impersonation of this user is already done. During the change process of the password, the web application should check the newly set password to fulfil the original password quality requirements. If this is not the case, the attempt should be declined, informing the user that something went wrong. This finding was rewarded with 2

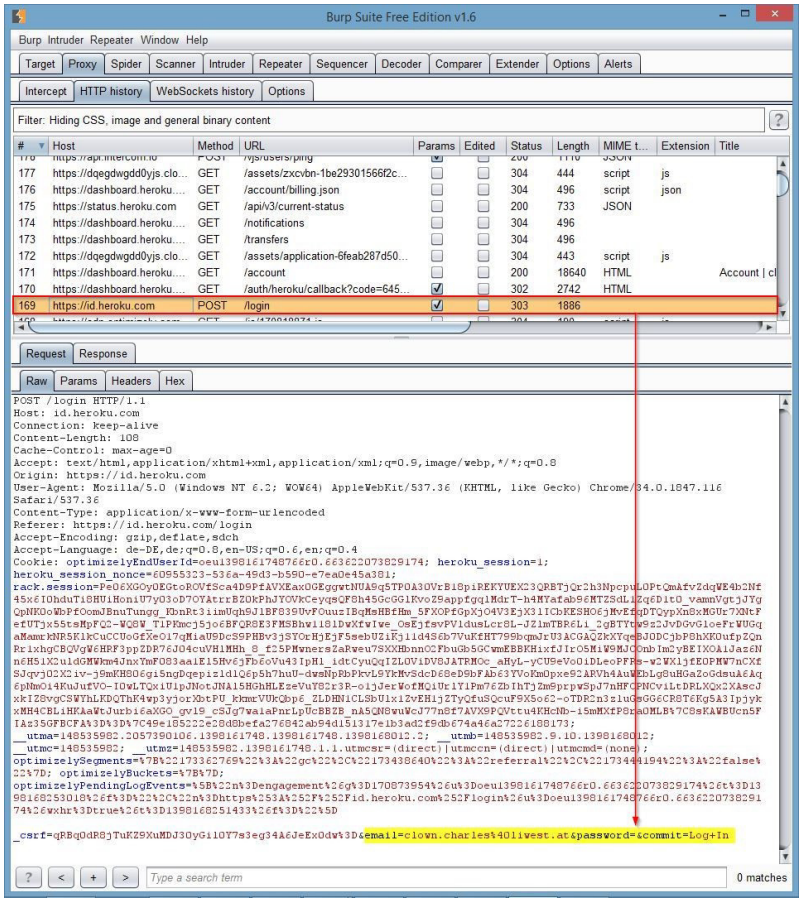

Figure 8: Logging in into the application without a password

points for the internally managed skill ranking, but was stated as duplicate as somebody else had found this flaw before.

4.2 Chapter Summary

This chapter presents all findings from the three penetration tests executed on eBay Magento eCommerce, Bugcrowd Inc. and Heroku, which have been done consecutively in the period between February 17th 2014 and April 25th 2014.

A short description about all the security flaws found during the penetration test is provided. Additionally, some figures are shown, which illustrate the possibility to exploit these breaches in order to make it easier to understand. Some examples are given, how an attacker could make use of these vulnerabilities. In the end of every valid submission, a short statement is given, how it would be possible to patch and avoid these flaws. The result of Bugcrowd Inc. is included, providing information about the validity of the submission and the reward earned for every single finding.

5 Conclusion

Three websites have been evaluated for the purpose of this thesis to receive some results about their web application security. Compared to most of the standard applications, which can be found in the World Wide Web, these websites have been the target of security analyses before. Accordingly, the assumption was given, that it is hardly possible to find new vulnerabilities. A conclusion on this initial thought is given in this chapter.

5.1 Project Evaluation

The project goals stated in chapter 1.1 and the project expectation stated in Chapter 3.1 give information about all the tasks which should be evaluated in this thesis. In the following sections, the results are given, including a conclusion.

5.1.1 Security Level of Tested Web Applications

The overall security level of all evaluated websites was high as the vulnerabilities with the highest appearance probability, according to the statistics in chapter 2.4, were not found. A sole exception are possible brute force attacks, which could be found on every single website. This originally lies in the fact that the operators are not changing these implemented algorithms because they do not see high risks in being compromised. This information has been received, for example by Bugcrowd Inc., informing the tester that the finding is valid but the operator will not change anything.

The vulnerabilities found in these applications should still be taken seriously and be patched as soon as possible. The impact of these findings was relatively low, which means that with these breaches, it would not be possible to harm other users or retrieve sensitive information about them.

The high level of security demonstrates that community help is helpful to secure a website. With a huge amount of security researchers helping, having different skills, the amount of security flaws can be reduced to an acceptable low number.

5.1.2 Submission Process

The submission system offered by www.bugcrowd.com is created in an easy way. Every bounty description, which can be opened through clicking on a link in the main bounty overview, has an integrated button that directly links to the submission form. For a penetration tester, it is therefore possible to reach this form without clicking through multiple web links. With nine different input boxes, two decisions to make through radio buttons and a possibility to upload pictures, the process to submit a finding is quick and easy. The information that is necessary is limited and should be already available after detecting a vulnerability. After submitting the form, the finding with its entire description cannot be read anymore. The submission just can be found in the submission section on the website, where it is displayed by showing only the caption and a reference number. Consequently, it is hard

to understand a previous finding, which was submitted several days before. The only opportunity a user has, is to use the save functionality (e.g. save as HTML) offered by the used browser in order to check the information later. This implementation is not user-friendly and could be improved.

The response to successful submissions is made by sending emails to the user. With every single submission, an email is sent, telling the user first, that the finding is being reviewed. In a second stage, Bugcrowd Inc. informs their testers about the validity of an already checked submission. This information via email is nice to have as normally emails are being checked more often than the bugrowd.com tester portal. At the time an email is received, it can also be checked in the user portal, how the submission was processed. Bugcrowd additionally gives short statements to every single submitted finding, clarifying the testers, why a vulnerability was declared valid, invalid or duplicate. This process chain is implemented well and gives the tester all the information needed in time.

For every single finding, there are also rewards that can be earned by submitting vulnerabilities. This information comes together with the validity status. The points for the internally managed skill ranking are distributed within one day and change the rank in the "myscore" tab of bugcrowd.com. Monetary rewards are shown immediately in the "payments" tab, being displayed as pending payments. The amount of time it takes to receive the money, being transferred only to PayPal accounts (Internet payment service, can be found at www.paypal.com), was between 7 days and an ongoing waiting time of more than 30 days. This latency is quite annoying and hardly understandable as the website ensures their users the monetary reward with above-mentioned email.

5.1.3 Retesting of Vulnerabilities

The purpose of letting a website be tested by penetration testers is to receive vulnerability reports that can be used to increase the security. Therefore, it is assumed, that all reported findings would be patched in the original web application as soon as possible after the submission was validated. To check if the website owner patches the security flaws, a retest of all detected vulnerabilities has to be made. The following listing is a short overview of all findings considering their patch level: (The values in brackets show the submission dates and the times since the findings were validated at May 12th 2014)

- EBay Magento eCommerce: The findings cannot be retested, as the website, which was available for testing, was removed from the original link.
- Bugcrowd Inc.: User Enumeration Possible (2014-04-11; 21): This security flaw is still existing in the web application.

  Circumvent Password Rules (2014-04-11; 21): The security breach has been patched and now tells, "Password may only contain printable UTF8 characters".

  Password Quality (2014-04-10; 21): The password requirements have not been changed by Bugcrowd Inc. and are still weak.

Password Lockout (2014-04-10; 21): The security breach has been patched and now tells, "Password may only contain printable UTF8 characters".

- Heroku:

  Circumvent Passwords (2014-04-22; 17): This finding is still not patched although the submission has been validated as duplicate.

  Password Flaws (2014-04-22; 12): This problem is still existing, as Heroku stated that this flaw would not get fixed immediately due to a not high enough risk.

5.1.4 The Bug Bounty Program

Traditional penetration tests, which are executed by companies being specialized on these tests, cost a lot of money. The bug bounty program offers a good alternative for website operators, who cannot afford to spend a huge amount of money on their website security. Another beneficial effect of these programs is, that way more people are testing the same website, which results in a better coverage regarding hacking skills, as different testers focus on different techniques. This may be one of the main reasons, that even big companies, which can easily afford a professional penetration test, let their applications being tested by the community.

The program offered by www.bugcrowd.com was easy to use. After a registration process to get a tester account, the backend has been available. The main tabs are limited to just a few important menus. The application logic is straightforward which enables a new tester to start using the bounty application easily without reading a manual. The number of available bug bounties is currently low, as the website has not been existing for a long time. During the writing of this work, the average number of available bounties increased from five to fifteen.

To receive the monetary rewards, in case a finding was submitted whereby monetary refund was promised, a tester has to have a PayPal-account. Here it would be more convenient to offer more than just one possibility to receive the money. Nevertheless, PayPal is a secure way to transfer money online. It is easy to use and free to register an account.

To sum up, the bug bounty web application of Bugcrowd Inc. is already very sophisticated and fulfils the main demands (e.g. finding submission form). During the writing of this thesis, the website was redesigned which led to a more clean and modern look. The main backend functionality stayed the same.

5.2 Limitations and Room for Improvement

During the practical part of this thesis, which was mainly characterised by the security analyses made for bugcrowd.com, occurred some problems that leave space for improvement of the work.

5.2.1 Time Constraints

Penetration tests are a very time consuming activity. It is never possible to check every feature, a web application is offering, in a way to guarantee perfect security. The schedule for the testing is normally designed in a way that it is checked first if the most common vulnerabilities are available. As it was necessary to check multiple websites to gain comparative values, this approach was taken to check the three websites. At the end, approximately 56 hours have been spent on each individual web application. According to 2-sec.com, "*the industry "norm" seems to be 4 days of infrastructure/web application testing, plus 1 day report writing* (italics added)". Additionally, it is stated that time is not enough to do an extensive test. [21] Assuming a normal working time of 40 hours a week, the amount of time used in this thesis is just slightly above the minimum time normally spent on a web application penetration test.

5.2.2 Cost Constraints

For this work, only open source and free versions of commercial tools have been used to execute the penetration tests. The reason is that there was no budget available for purchasing software licences. With these prerequisites, many important tools, which are helpful, could not be used. As an example, the Nessus Vulnerability Scanner by Tenable network security, which has "more than 20000 customers worldwide". [22] "*The versatile Nessus® vulnerability scanner provides patch, configuration, and compliance auditing; mobile, malware, and botnet discovery; sensitive data identification; and many other features.*

With a continuously updated library of more than 60,000 plugins and the support of Tenable's expert vulnerability research team, Nessus delivers accuracy to the marketplace. Nessus provides multi-scanner support, scales to serve the largest organizations, [...] (italics added)". [23] The most important tool used in the context of this work was Burp. This software, which was introduced in chapter 3.3.4, was also just used in a free version which lead to a more time consuming usage, as main functionality was throttled or even not available at all.

For a follow-up research, it would be highly recommended to contact companies, which have their core knowledge in information security, to get software licence contributions.

5.3 Future Work

The achieved knowledge will be used for executing further penetration tests, which are managed by Bugcrowd Inc. The skill points (Kudos points) received during the thesis are highly important to receive special invitations to more exclusive penetration tests. Apart from just executing web application penetration tests, the scope should also be placed on testing mobile applications (primarily Android applications) and client applications for Windows operating systems. In order to know how these tests can be executed, additional literature has to be read to understand the basic principles that are necessary to start the security analysis. After a while, advanced techniques should be learned to increase the probability to find security breaches by having a highly skilled knowledge base.

As additional important step, an already registered domain should be used to set up a website in the World Wide Web to make the information and the findings accessible to interested people and other security researchers. After the agreed non-disclosure period, a detailed description of all findings including the way to exploit them, the security impact and an approach how to remove the flaws should be demonstrated.

The website should also serve as a personal profile in the field of IT-Security, giving an overview of actual work in progress and a history of activities. With more content and interesting posts, attention of people should be drawn on the effort taken to receive job opportunities outside of the bounty programs.

Out of interest, creating tools to ease the penetration testing process should take into consideration. In case a useful helping tool would have been created, the source code should be made available on various software distributing websites free under the GNU General Puclic Licencse (https://www.gnu.org/licenses/gpl-3.0.en.html).

In general, the interest in the field of penetration testing should be kept alive by dealing with this topic on a weekly basis.

5.4 Chapter Summary

In this chapter a conclusion was drawn on several points which have been formulated in the project expectation in chapter 3.1. The web application security level of all tested websites had a high standard. The submission process by www.bugcrowd.com offered an easy design, to make it convenient for testers to submit their findings. To check if these submissions are taken serious, a retest of all findings was necessary, which brought some problems. Some findings could not be tested anymore as the website operator used a test instance on a domain, which was deleted afterwards. Other operators stated that the security severity is not high enough to patch the vulnerability. Nevertheless, the bug bounty program is a good step to improve the website security of applications that are taking part. This development is credited as more and more operators decide to let their website be tested by the community.

For better results, it would have been necessary to invest more time in every single penetration test executed. The tools used for searching vulnerabilities should be available as commercial edition instead of using free versions, which do not offer the full functionality.

The work on this topic will contain further research on different kinds of applications in the future. Additionally, a platform should be set up to enable other security researchers to read about findings and interesting ways about how to exploit vulnerabilities.

6 References

[1] K. Hirai, "Sony's response to the u.s. house of representatives." letter URL: https://www.flickr.com/photos/playstationblog/5686965323/in/set-72157626521862165/ (retrieved, May 17th 2014, 14:13), May 2011.

[2] P. Seybold, "Update on playstation network and qriocity." URL: http://blog.us.playstation.com/2011/04/26/update-on-playstation-network-and-qriocity/comment-page-2/#comments (retrieved, May 17th 2014, 14:48), April 2011.

[3] C. R. Pfleeger and S. L. Pfleeger, *Security in Computing* -. New Jersey: Prentice Hall Professional, 3rd revised us ed ed., 2003.

[4] M. Mealling and R. Denenberg, "Uniform resource identifiers (URIs), URLs, and uniform resource names; (URNs): Clarifications and recommendations," August 2002. RFC 3305.

[5] U. S. DoD, *Department of Defense trusted computer system evaluation criteria* -. Dept. of Defense, 1987.

[6] T. Wilhelm, *Professional Penetration Testing - Volume 1: Creating and Learning in a Hacking Lab*. Burlington, MA: Syngress, 1. ed., 2009.

[7] P. Engebretson, *The Basics of Hacking and Penetration Testing - Ethical Hacking and Penetration Testing Made Easy*. Amsterdam: Elsevier, 2. ed., 2013.

[8] Course Technology, *Penetration Testing: Procedures & Methodologies* -. Clifton Park, NY: Cengage Learning, 2010.

[9] S. Desikan and R. Gopalaswamy, *Software Testing - Principles and Practice*. Delhi: Pearson Education India, 2006.

[10] B. B. Agarwal, S. P. Tayal, and M. Gupta, *Software Engineering and Testing* -. Sudbury, Massachuetts: Jones & Bartlett Learning, har/cdr ed., 2010.

[11] Unknown, "Website security statistics report may 2013," tech. rep., WhiteHat Security, May 2013.

[12] Bugcrowd, "What is a bug bounty?." URL: https://bugcrowd.com/resources/what-is-a-bug-bounty (retrieved, March 31st 2014, 14:15).

[13] Bugcrowd, "Bugcrowd titlepage." URL: https://bugcrowd.com/ (retrieved, May 19th 2014, 00:02).

[14] w3schools, "Browser statistics." URL: http://www.w3schools.com/browsers/browsers_stats.asp (retrieved, March 25th 2014, 17:35).

[15] Portswigger, "About burp." URL: http://portswigger.net/burp/ (retrieved, March 25th 2014, 19:28).

[16] T. Dierks and C. Allen, "The TLS protocol; version 1.0," January 1999. RFC 2246.

[17] D. Stuttard and M. Pinto, *The Web Application Hacker's Handbook - Discovering and Exploiting Security Flaws*. New York: John Wiley & Sons, 2011.

[18] OWASP, "Cross-site-scripting (XSS)." URL: https://www.owasp.org/index.php/XSS (retrieved, April 7th 2014, 11:43).

[19] Magento, "About Magento." URL: http://magento.com/company/overview (retrieved, April 4th 2014, 19:22).

[20] Heroku, "About Heroku." URL: https://www.heroku.com/about (retrieved, April 23rd 2014, 14:34).

[21] 2-sec, "Penetration testing – building the business case." URL: http://www.2-sec.com/2013/10/31/penetration-testing-building-business-case/ (retrieved, April 29th 2014, 14:15).

[22] Tenable, "Products." URL: http://www.tenable.com/products (retrieved, April 29th 2014, 14:28).

[23] Tenable, "Nessus vulnerability scanner." URL: http://www.tenable.com/products/nessus (retrieved, April 29th 2014, 14:31).

7 Appendix

Additional information concerning this thesis can be obtained from following URL:

www.pascalschulz.at/bachelorthesis/appendix.zip

This compressed file is encrypted with a password chosen by the author of this work. To get access and be able to decompress the folder successfully, contact the author.

The folder contains following data:

- Project Documents: Project Timeplan/Milestones/Specification
- Screenshots: For each penetration test which has been executed, there are additional screenshots of the testing process available
- Sources: All sources that have been used and which were available online are saved as PDF-files
- Submissions: All submissions made to bugcrowd.com are available as HTML-files