TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Science

ITV70LT

Liivi Volt 111504 IVCMM

ANALYSIS OF VULNERABILITIES IN ESTONIAN WEB APPLICATIONS

Masters thesis

Elar Lang
MSc
Penetration tester/Trainer

Rain Ottis
PhD
Associate Professor

Tallinn 2014

Declaration

I hereby declare that this Masters Thesis is the result of my own research and investigation except as cited in the references. The work is original and has not been previously submitted for a degree or diploma at any university.

Author:
Liivi Volt

…… ……

(signature) (date)

Abstract

The aim of this thesis is to provide information about the current state of security of Estonian web applications. The goal is reached by conducting a thorough analysis using the data compiled from statistics based on Clarified Security OÜ penetration test results. The topic is relevant as Estonia is considered to be one of the worlds leading e-countries, yet the status of vulnerability statistics in local web applications has not been researched and is thus largely unknown.

The thesis is divided into seven chapters. The thesis begins with an introductory chapter, followed firstly by theoretical aspects and then by a brief introduction of Clarified Security OÜ, who provided the data on which the analysis in this thesis is based on. The fourth chapter explains the used methodology which is then followed by the description of the empirical data used in the thesis. The subsequent chapter conducts an analysis of the data and, based on the results, also provides additional information about the most common vulnerabilities. The chapter ends with a conclusion of the analysis and recommendations for further research. The thesis is then finished with a general summary.

The planned outcome of the thesis is to obtain knowledge on which are the typical vulnerabilities that have been found in Estonian web applications and how do they compare to foreign statistics. The thesis will provide information on what areas should be more emphasized when educating IT specialists for the Estonian market and which topics should be under accentuated interest during not only development, but already planning and ordering stages for new applications.

The thesis is written in English and contains 68 pages of text, 7 chapters, 13 figures, 5 tables.

Annotatsioon

Eesti veebirakenduste turvavigade analüüs.

Antud lõputöö eesmärgiks on anda ülevaade Eestis kasutusel olevate veebirakenduste üldisest turvatasemest. Töö eesmärgi saavutamiseks viiakse läbi põhjalik analüüs, mis põhineb Clarified Security OÜ poolt teostatud turvatestide raportite alusel koostatud andmetel. Antud teema käsitlemine on oluline, kuna kuigi Eestit peetakse maailmas üheks juhtivaks e-riigiks, siis kohalike veebirakenduste turvalisuse hetkeolukorrale pole senini, põhiliselt täpsema informatsiooni puudumise tõttu, põhimõtteliselt tähelepanu pööratud.

Lõputöö on jagatud seitsmesse peatükki. Töö algab sissejuhatusega, millele järgnevad teoreetilisi aluseid ja Clarified Security OÜ'd tutvustavad peatükid. Seejärel kirjeldatakse lühidalt töös kasutatavat metodoloogiat. Viiendas peatükis esitatakse ülevaade töös kasutatavatest empiirilistest andmetest, järgnevas peatükis teostatakse andmetel põhinev analüüs ning vastavalt analüüsi tulemustele kirjeldatakse lühidalt enamesinenud turvavigu. Antud peatükis antakse ka soovitused edasisteks uurimusteks ning tehakse analüüsi põhjal järeldused. Töö lõppeb kokkuvõttega seitsmendas peatükis.

Lõputöö tulemusena antakse ajakohane ülevaade tüüpilistemast Eesti veebirakendustes esinenud turvavigadest. Samas tuuakse välja ka võrdlus, kas ja kui palju leiud erinevad muu maailma statistikast. Vastavalt analüüsi tulemustele toob lõputöö ühtlasi välja kitsaskohad, millele peaks tulevikus Eesti infotehnoloogiaspetsialistide harimisel ja koolitamisel rohkem tähelepanu pöörama ning milliseid aspekte peaks rõhutama uue veebirakenduse tellija, koostades uue rakenduse nõuete ja vajaduste nimekirja

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 68 leheküljel, 7 peatükki, 13 joonist, 5 tabelit.

Table of contents

Abbreviations

ASVS	Application Security Verification Standard
CSRF	Cross-Site Request Forgery
CSS	Cascaded Style Sheet
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
NATO CCDCoE	NATO Cooperative Cyber Defense Centre of Excellence
NDA	Non-Disclosure Agreement
OWASP	Open The chapter is finished with an analysis of Web Application
Q1	Security Project
Q2	First quarter of a year, from January to the end of March
Q3	Second quarter of a year, from April to the end of June
Q4	Third quarter of a year, from July to the end of September
XSS	Fourth quarter of a year, from October to the end of December
	Cross-Site Scripting

Table of figures

List of tables

1 Introduction

The main aim of this precluding chapter is to create a base for understanding on what this thesis is written about. The chapter begins with explanation about the background of the topic of the thesis and discusses basic problems of the area in question. This part is then followed by stating the purpose of the thesis and descriptions of delimitations to illustrate the scope. The chapter ends with a introduction to the structure of the thesis to simplify further reading.

1.1 Background

As of May 2014 there are roughly 12 billion different people, processes, data and other things connected to the Internet [1]. Somewhere in between 2008 and 2009 the amount of different devices connected to the Internet passed the worlds population and it is estimated that by 2015 there will be about 3,5 connected devices per person [2]. When taking into account the fact that there are actually only about 34% of the whole population that actually use the Internet [3], then that number can be considered to be even bigger. Most of the people that use the Internet use it not only on their computers but often also on their smartphones or tablets. Most, if not all, of these devices are equipped with some kind of web browser and thus can be considered to be used as possible web application clients.

With all that in mind it can be safely said that in case you are an owner of any kind of connected device, it is rather difficult, if not to say even impossible, not be a potential user for various kinds of web application. They have become important parts of our daily lives and include a notable amount of information on their users. It should also be mentioned that possibilities of creating different web applications are almost limitless as they are not tied to a certain operation system nor a programming language [4]. The fact that different kinds of connected devices have become extremely widespread, has lead developers to create web applications that are used as administrative or configuration interfaces for several different systems. One of the most common home user devices that are

administered through web applications is the router. Similar systems are also used for smart home systems, printers, smart-TVs, digiboxes, IP-phones, IP- and security cameras etc.

The Internet used to consist of static web pages which have now evolved into complex multifunctional applications. New technologies and components are constantly developed and integrated into old and new applications. As technology develops, attack vectors develop also. More and more totally new web applications are constantly finished and sent live. At the same time there are a lot of old(er) applications that are still in use in their original state. Systems that were once though to be safe, have become vulnerable over time. If these applications and websites are not properly maintained or updated, they might become easy targets for the attackers.

Most of the current websites include various functionalities ranging from a simple search possibility to having multiple user accounts. Almost always they also contain a lot of different kinds of data. Although most of the data is publicly visible, a notable amount is hidden in applications information systems and databases. Obviously, where there is data, there are also people who are interested in obtaining it. Attackers spend a lot of time figuring out possible web application vulnerabilities, finding them in live systems and trying to exploit these to get access to any kind of data.

Most cyber security related reports unanimously declare that websites and web applications have been one of the leading targets of cyber-attacks for years and seem to be keeping their position also in the near future [5]. For example an Acunetix survey has deemed as many as 70% of websites having vulnerabilities that could be exploited by a malicious party and lead to theft of sensitive data [6] and a recent Verizon report claimed that during 2013 there were over 3000 reported web application incidents, of which 490 had confirmed data disclosure [7]. Those numbers might not seem very significant at first glance, at least for the IT-dependent world of today, but one should keep in mind that most of web application related incidents that are reported, are done so by large companies and organizations (in case of Verizon's report 75% incidents were reported by organizations

with 1000+ members) and it is unknown how many incidents worldwide are not reported at all. Additionally, the dataset behind the report in question consists of data from 51 different cyber security related contributors from 95 countries around the world. [*ibid*]

Although security awareness is a very important and relevant issue, especially in the field of web applications, it still needs much more publicity and attention. Developers should be constantly informed and reminded of typical and most dangerous vulnerabilities so that nobody would publish an application that could be attacked and exploited as soon as it has gone live. Raising awareness and improving the general knowledge will hopefully help to improve the currently somber statistics and bring us a safer and better Internet in the future.

1.2 Problem discussion

Almost every year different companies and organizations create several reports and papers which illustrate the state of most critical web application security flaws presently in the web scene. One of most noteworthy of these is the OWASP Top Ten Project [8], which is a powerful awareness document that is compiled based upon years worth of data from different security related organizations from around the world. Therefore it gives a relatively plausible overview of the current state of web applications in general.

Still, there has not yet been a study about web application vulnerabilities that focus on applications which have been developed for the Estonian market. Knowing, which vulnerabilities are currently most common in different kinds of web based applications, will help developers to concentrate on eliminating these kinds of mistakes early in the development process. Also it will hopefully provide useful knowledge for university lecturers and (web application) security trainers about on which areas they should concentrate on more thoroughly. As the author works for Clarified Security OÜ, which is the one of the most influential penetration service providers currently in the Estonian market, the thesis will not only help to understand main and most common problems in

tested applications but also to improve one of our most popular trainings with most current problems.

1.3 Purpose of the thesis

The purpose of this thesis is to analyze anonymous data based on penetration test reports by Clarified Security OÜ to determine vulnerabilities found in Estonian web applications. The exact extent and definition of Estonian web applications will be defined in Chapter 1.5. As a result of the analysis the thesis will reveal which are the most common security flaws and how different are the most common vulnerabilities of the tested applications compared to the ones of different vulnerability reports from around the world.

1.4 Theoretical aspects

To achieve the purpose of this thesis several theoretical and empirical subjects should be studied. This subchapter provides research questions that need to be clarified to help the thesis achieve its objective.

The aim of this thesis is to analyze vulnerabilities in Estonian web applications, therefore a theoretical knowledge base on the subject should be established. To create a solid understanding of everything that will be discussed in this thesis, knowledge on different subjects should be obtained beforehand: what exactly are web applications, what is penetration testing, background information on the company that has provided most of the data for the thesis, and introduction to the methodology the company uses to create the penetration testing reports. To fully understand the methodology, some background information of the organization that has created the system should also be provided. Hence to create a satisfactory theoretical framework the following questions should be answered:

1) What are web applications?
2) What is penetration testing?
3) What are OWASP, OWASP ASVS and OWASP Top Ten?
4) Introduction of Clarified Security OÜ and their services.

5) How do Clarified Security penetration test reports represent Estonian web applications?

1.5 Limitations

A valuable masters thesis has to set limitations to its scope to investigate its research questions in a deeper level. This thesis will focus on analyzing vulnerabilities in Estonian web applications. Main identified limitations of this thesis are explained as follows.

Firstly the data used for analysis in this thesis will be based solely on penetration testing reports that have been issued between Q4 2011 and Q1 2014 (From 01.10.2011 to 30.03.2014) by Clarified Security OÜ. All these kinds of reports are protected by NDA agreements and therefore data used in the analysis is mostly anonymous. The author has information about the client (which is still considered confidential), type of tested application, enforcement time, official testing durations in work hours, corresponding OWASP ASVS [9] level, found vulnerabilities and their severity levels.

Secondly the data used was based only on penetration tests which were performed on applications that were developed specially for use in the Estonian market. Tests that were conducted for applications which were used in other countries were excluded from the sample. It should be mentioned that the author was provided with information that some of the applications that were developed for the Estonian market were developed outside Estonia by developers who are not Estonians. Hence this thesis does not only analyze application vulnerabilities that were developed by Estonians for Estonians.

All the tests that were included in the data have been conducted accordingly to the 2009 release of OWASP ASVS [11] and at least on OWASP ASVS Level 2 (Manual Verification) or higher. OWASP Level 1 or Automatic Verification was excluded as the company has not performed any tests on this level.

1.6 Thesis overview

Hereby, the layout of the thesis is introduced to simplify further reading and clarify the reasoning behind the structure. The introductory chapter briefly introduced the background situation, outlined the problem discussion, stated the purpose of the thesis and listed its limitations.

The thesis continues with theoretical aspects in Chapter 2. This chapter introduces the basics behind web applications and penetration testing in general and also gives a brief introduction to OWASP and its relevant projects - OWASP ASVS and OWASP Top Ten and their releases. Chapter 3 provides the introduction of Clarified Security OÜ and their services. It is then followed by Chapter 4 which covers the chosen research methodology for reaching the stated goals.

The subsequent chapter describes the data used in this thesis. This chapter gives as accurate information about the data used in this thesis as possible. The chapter also provides an overall summary of the first impression that the data provides.

After that Chapter 6 analyses the data using several different combinations of the dataset. In this chapter the results of the analysis are discussed, information on the most notable results is provided and the whole chapter is summarized with a conclusion. This section of the thesis also answers the research questions presented earlier and provides suggestions for further research on this topic.

The final Chapter includes the general summary of the thesis.

2 Theoretical aspects

The purpose of the theoretical chapter is to provide the necessary knowledge to fully understand the content of the following chapters and to simplify further reading. The chapter introduces different thesis related principles to provide a solid foundation to understand the following thesis and analysis.

2.1 Web applications

The easiest definition of a web application says that it is an application that is accessed by users over a network, such as the Internet or an Intranet. Usually web applications are computer software applications that are coded using a browser-supported programming language (for example in HTML using CSS) and are dependent on a common web browser to make the application usable. [12]

It can be said that these days everyone is constantly using web applications, whether they like it or not. Static text and graphics showcases have become items of the past as modern web pages have become dynamic systems, almost specific computer programs, that allow visitors to submit and retrieve data to and from databases over the Internet while using their preferred browsers [13]. Web applications create a user experience suited specially for the person in front of the screen.

The key reason behind web applications popularity is that they can be updated and maintained without installing and distributing software on client devices, but also for their basic support for cross-platform compatibility. Modern websites are formed by features like login pages, webmail, shopping carts, support and product request forms and content management systems and therefore provide businesses means to communicate with their customers. [12]

Hardly any current websites are static and use only plain text and graphics. Currently most daily used web pages do not only create a unique user experience for their visitors but also contain a lot of different kinds of data, most of which is stored in databases. If an application has been developed without giving any thought on its security aspects, these databases might be under potential threat by malicious web users. Even some older applications, that have not been updated, and might have been well secured when they were released, could have become vulnerable over time.

2.2 Penetration testing

Firstly the difference between security testing and penetration testing should be clarified. Although "penetration testing" is often used as a fancy word for any type of security testing, it is not quite so. Security testing is considered to be a method intended to disclose weaknesses in the security mechanisms of an information system that is meant to protect data and maintain functionality as intended. As a term, security testing has many different meanings and can be referred to in a number of ways. Besides penetration testing other forms of security testing include vulnerability scans, vulnerability assessments, security assessments, security audits and security reviews. [14] All the aforementioned can be used together but also separately. Therefore, not all security tests are penetration tests but a penetration test is always a security test.

Two most common but very different security testing approaches are penetration testing and vulnerability assessment. In short, a vulnerability assessment finds out what are the systems possible weaknesses and how they should and could be fixed, on the other hand a penetration test proves the existence of security issues by finding vulnerabilities, exploiting them and accessing data that should not be accessed. Hence a vulnerability assessment is meant to improve security attitude and therefore develop a better security program, whereas a penetration test shows how your defense can be defeated - the presence or lack of its current security programs effectiveness. [15, 16]

It has been said that penetration testing can be defined as a legal and authorized attempt to locate and successfully exploit computer systems for the purpose of making those systems more secure [17]. The process itself includes searching for vulnerabilities and also providing proof of concept attacks for exploit demonstration. A penetration test always ends with a report which includes recommendations for fixing and addressing discovered issues. Sometimes penetration tests are called “white hat attacks” as the attacks conducted during them are not of malicious origin. [18]

There are different sub types for penetration tests. There are “white box” tests, where most of the information about the application or system is given prior to the test (background, system type, source code etc) and “black box” tests where the attacker has only very basic or no knowledge about the system and is treated by the application as a random user [19]. Also, penetration tests can be done either manually or automatically by using penetration testing tools. In any case, all methods follow a similar workflow - first determining the scope of the test, next gathering information about the target or reconnaissance, thirdly identifying entry points and trying to exploit these in order to break into the system and end with reporting back with the findings. [17]

Although automated penetration testing is cheaper and faster than manual testing, there are several reasons why to prefer the alternative. As a rule manual testing also includes some basic automated testing and the combination of using mainly manpower with some additional automated tools gives the best results. Automated tools by themselves are very limited when it comes to finding errors in custom code, almost hopeless in finding business logic implementation mistakes (as they hunt for technical flaws) and very likely to cause problems when used against a live/production system. It should also be mentioned that automated tools only get updated once in a while and might therefore not be capable of finding newer flaws and also usually include a high percentage of false positives [20]. Choosing the right method for a penetration test mostly depends on the type, context and needed security level of the test subject. Low-priority and simple applications might be easier to automatically scan to find the bigger vulnerabilities, high-end applications which include sensitive data should definitely (also) be manually tested.

2.3 OWASP

OWASP, which is short for Open Web Application Security Project, is one of the most respected authorities in the field of web application security. It is a worldwide non-profit charitable organization, which main focus is improving software security. The organization was founded in 2001 and their main mission is to help make conversant decisions about real software security risks by making software security visible and understandable for individuals and other organizations. Anyone can participate in OWASP and all their materials are available under a free and open software license. [8, 21]

Projects built by OWASP cover many aspects of application security. They develop documentation, tools, teaching environments, guidelines, checklists and other materials for helping organizations to improve their capability to produce secure code. All materials OWASP provides are organized into different categories. There are:

- **Flagship projects**, which have proven to provide strategic value and established quality to OWASP and application security in general. Some of the most well known of OWASP Flagship projects are the OWASP Application Security Verification Standard Project, OWASP Top Ten Project, OWASP Code Review Guide Project, OWASP Testing Guide Project.
- **Lab projects**, which are typically not yet production ready but have produced a deliverable value and the community expects them to soon be ready for mainstream usage.
- **Incubator projects**, which are projects that are still being fleshed out, with ideas still being proven and development is still underway. [22]

For this thesis the concept behind two OWASP projects should be explained a bit more thoroughly.

2.3.1 OWASP ASVS

The OWASP ASVS (Application Security Verification Standard) is basically a standard for verifying the security of applications. It provides a basis for testing web application

technical security controls and it's Beta release was the first official standard publication by the OWASP in December 2008 [9]. The main aim for this project is to organize the range of coverage and level of accuracy when performing application-level security verification. The standard defines a set of technical controls for applications that should be verified as a part of security testing process. The ASVS is intended to be independent from application and development life-cycle and define requirements that can be applied across web applications without special interpretation. The project team is currently working on finalizing the ASVS 2.0 [10]. The ASVS version used for tests that the data used in this thesis is based on is the final version of ASVS 1.0, which was released in June 2009. Hence all the theoretical base featured in this chapter is also based on the ASVS 1.0.

The standard provides verification requirements that prescribe a white-list approach for security controls. The number of main security requirement areas for the 2009 release of the standard is 14. For each of the main requirements there is a matrix for what is required to test that specific requirement based on the ASVS level that is used. Each requirement in turn includes 2-15 specific verification requirements. [11]

OWASP ASVS Levels

OWASP ASVS can be tested according to four main levels, which are all numbered according to the severity of the conducted test and which increase in breadth and depth as one moves up the levels. The breadth is defined in each level by a set of security requirements that must be addressed. The depth of the verification is defined by the approach and level of rigor required in verifying each security requirement. [23] Using tools during testing on all levels is highly encouraged but their results should always be manually verified. A MITRE research found that all different application security tool vendors claims put together only cover 45% of known vulnerability types. As these tools had minimal overlap, one would have to use all of them to achieve this percentage. [24] It should also be mentioned that automated tools are very limited in finding errors in custom code and next to hopeless in finding business logic implementation errors and very likely to cause problems when used on live/production systems. [31]

Two lower levels have also been divided into sub-levels. The main idea behind different levels is pictured on figure 2.1 [11] below.

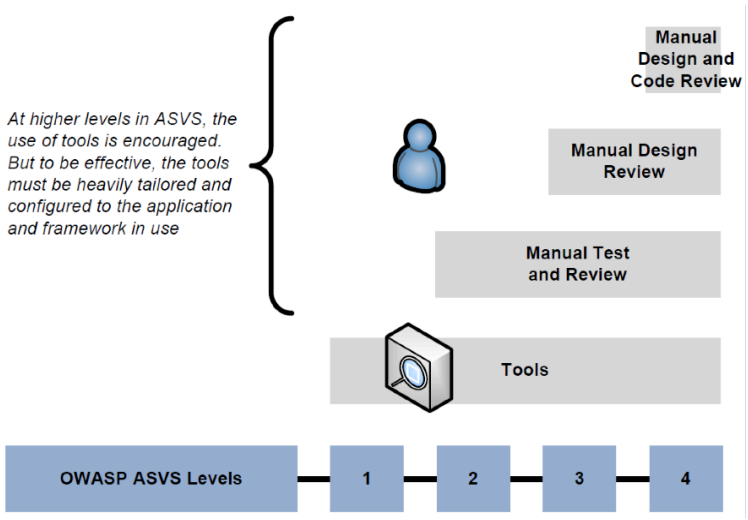

Figure 2.1 OWASP ASVS levels

If the target of verification meets all of the requirements at the level targeted, then it can be considered an OWASP ASVS Level X application, where X is the application level that the application complied with. To earn a level, all found vulnerabilities must be mitigated or fixed and the application later re-verified. None of the applications that were tested for the data used in the analysis were verified for ASVS Level 1, most of the applications were verified against Level 2 and a few against Level 3. [11]

2.3.2 OWASP Top Ten

The OWASP Top Ten is not a standard nor an application security program but an awareness project. Its mission is to raise knowledge of the currently most important and critical risks in web application security and at the same time not to include every possible risk that an application could be vulnerable to. [25] The project was first released in 2003, had minor updates in 2004 and 2007, the 2010 version was revamped to prioritize by risk [26] and the most recent release of 2013 follows the same approach [27]. The Top Ten risks are chosen based on a risk rating methodology. For each risk the project provides a

description, example vulnerabilities, example attacks, guidance on how to avoid and references to OWASP and other related resources.

The OWASP Top Ten for 2013 is based on eight datasets from seven firms that specialize in application security; four of them are consulting companies and three different tools vendors. All together this data covers over 500 000 different vulnerabilities from thousands of applications used by hundreds of organizations. [27] The items in the Top Ten are chosen based on likelihood of an application having that vulnerability (prevalence) combined with consensus estimates of likelihood of that vulnerability being discovered (detectability), it's exploitability and typical technical impact estimates, if that vulnerability would be exploited. [28]

The project targets to determine the most serious risks for many different types of organizations. The names for risks are derived from the type of attack, the type of weakness or the type of the impact they could cause. Names are chosen to reflect the risks and align with common terminology. For all of the identified risks, generic information about likelihood and technical impact is provided by using a simple ratings scheme shown on figure 2.2 [27].

Threat Agents	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
App Specific	Easy	Widespread	Easy	Severe	App / Business Specific
	Average	Common	Average	Moderate	
	Difficult	Uncommon	Difficult	Minor	

Figure 2.2 Risk ratings

The process of producing an update for the project starts with collection of prevalence data from all the data suppliers. This data is then ranked and combined to create an updated list. The determined factors are then given values based on professional opinion and after that the final order of the Top Ten is calculated. The project is firstly published as a draft candidate for initial review by the data contributors and OWASP community. Then the updated version is released as a publish release candidate that the public can review and

comment and based on this feedback the final release of the project is eventually published. [28]

The main goal of the project is to educate developers, managers, architects and organizations about the consequences of the most important web application security vulnerabilities [26].

According to the project, currently the top three most serious risks are injection (A1), broken authentication and management (A2) and XSS (A3) [29]. The full list of the OWASP Top Ten 2013 Project risks can be found in appendix 1.

3 Clarified Security OÜ

As this thesis aims to provide an adequate overview of the current state of web applications in Estonia, the author believes it is also important to know the background of the company that has provided the data to base this thesis on, therefore Clarified Security OÜ will now be briefly introduced.

3.1 Introduction

Clarified Security OÜ is an Estonian capital based information security company, whose main areas of expertise are technical security tests and practical information security trainings. The company was founded by Mehis Hakkaja in September of 2011 and currently the team consists of six full-time employees, of whom five are everyday security professionals - practical penetration testers and information security trainers.

In 2012 the company provided over 2000 hours of security testing services (of which most were manual web application tests based on OWASP ASVS) and over 550 hours of practical security trainings. In 2013 the amount of security testing services rose to approximately 2300 hours and the amount of practical trainings to 700 hours. Additionally to internal trainings the company also started providing public training groups for interested individuals. The company has also been a contract partner for NATO CCDCoE “Locked Shields” 2012, 2013 and 2014 cyber security exercises as a “red teaming” service provider. [30]

Combined, Clarified Security penetration testing team has very extensive experience in the penetration testing field. The teams certifications include: ISC2 CISSP (Certified Information Systems Security Professional), ECSA (EC-Council Certified Security Analyst), ISTQB Certified Tester Foundation Level, CWAPT (IACRB Certified Web Application Penetration Tester), OSEE (Offensive Security Exploitation Expert), GWAPT (GIAC Web Application Penetration Tester) [31].

Company's slogan is “we break security to bring clarity” which roughly means that their main aim is to take supposedly safe systems or devices into pieces to demonstrate their creators how they exactly work, which vulnerabilities they include and how a malicious counterpart could possibly exploit these security flaws.

3.2 Products and services

As previously mentioned, Clarified Security is highly specialized in mostly manual penetration testing and in practical security training courses. A significant amount of every year is also spent on preparation and execution of the “Locked Shields” cyber security exercise, where the whole team is engaged as participants and group leaders of the Red team (the attacking team). The company’s annual red teaming services amount up to 800 work hours of preparation and execution time.

3.2.1 Penetration tests

The team behind Clarified Security is highly prized of doing manual web application penetration testing. The methodology generally used is the well-proven OWASP ASVS, mostly on Level 2 or 3. Over the years the team has been asked to test a wide range of very different web applications. Tested systems have included homepages, self-service portals from both private and public sectors, e-store and e-services and solutions for internet banking and healthcare services. [30]

The following subsection aims to introduce the type of penetration report form that Clarified Security OÜ currently uses and on which the data used in the analysis is based on. The reports have been custom formed according to the needs of the Clarified Security OÜ team and based on OWASP ASVS verification level checklists. [31]

Penetration test reports

The reports start with an executive summary where the overall impression of the application and its components is provided for the client and all notable findings and

observations are shortly mentioned. The summary has different paragraphs according to how many versions of the report have been sent out during testing. Usually there are at least two – version 0.1, which includes all preliminary findings and version 1.0 with final review results. Between the two the client is expected to have fixed the most notable and dangerous vulnerabilities reported in the first report.

The following part of the report is the scope specification. The scope defines all the different aspects of the project - the volume (work hours), the OWASP ASVS verification level, the applications or environments that will be tested and their types (if it is development, test, pre-live or live). The scope also provides a description of the test – whether white or black box approach was used, what was provided and what not provided by the client for the testers. The testing timeframe, testers traffic data and accounts used for testing are also elaborated in this section of the report.

Next chapter of the report describes all the findings that were made during the testing period with a corresponding ASVS reference. Every finding in the report includes a title, the description of the problem or situation, risk, recommendations for fixing or improvement and information about the criticality and status, accordingly. Status and criticality of each finding are marked separately as a log where every entry contains version report number, detection or reporting time, criticality at that time and, if needed, other notes. Findings are labeled and colored according to the severity ranking, which is given according to the testers best assessment of primarily technical severity. All findings are gathered under specific ASVS category (V1 - V14) and the main categories have been labeled according to the most severe finding of all findings in their subcategories. There are six different rankings for labeling findings which can be seen in the table 3.1 below.

- / INFO / NOTE	Verification requirement has not (yet) been assessed (or was not in scope) or simply marks noteworthy comments/findings that were made.
OK / FIXED	No vulnerabilities were found or findings were fixed and verified by the time of this report version update.
LOW	Low severity finding - application can go/stay "live". However, it is recommended to consider to fix findings (low urgency).
MEDIUM	Medium severity finding - application can generally go/stay "live". However, it is recommended to fix the findings.
HIGH	High severity finding - all findings should be fixed before going "live" or at the latest with next update/release cycle base of "live" applications.
CRITICAL	Critical severity finding - all findings MUST be fixed immediately. In case of "live" system, mitigate immediately or take the application off-line.

Table 3.1 Legend for findings severity rankings

Besides OWASP ASVS findings the report also includes a section for "other findings" in the end. There the testers have included some findings that do not fit under any of the "traditional" ASVS chapters or could be included in several chapters. Some of these "unclassified" findings would fit under one of the ASVS main categories but do not have a sub-category that is suitable and have therefore been classified as "other findings." [*ibid*]

3.2.2 Trainings

Besides being very focused on manual penetration testing, Clarified Security team also does a lot of practical security skills and knowledge transfer - they train both commercially and academically. The company offers all their trainings internally for interested companies and organizations and, as of rather recently, also as public trainings, where anyone interested can attend.

Currently the company offers three different trainings - Web Application Security, Hands-On Hacking Essentials and Secure Logging. First of those can be considered most relevant to the topic of this thesis as its main target audience is the developers and administrators of web applications. The main outcome of the training is to provide a thorough understanding of web application security basics and therefore help to produce better and

safer software. Therefore the research conducted during this thesis will contribute to further development of the training by outlining the vulnerabilities that are most common in Estonian web applications.

Additionally for several years the team of the company has taken part of teaching different introductory parts of their training during the Hacking Attacks and Defense course at Tallinn University of Technology Cyber Security Masters program.

4 Methodology

This chapter covers the methodological aspects of the thesis involving the research design issues, data collection and consumption. The chapter ends with the discussion of the quality issues of the used sources and data.

4.1 Research design

Research methodology indicates the procedural structure which handles the research. Its task is to describe the approach to a problem, which can be carried out in a research process. [32]

There are two categories of research strategies – quantitative and qualitative. Quantitative research is generally conducted by using scientific methods - for example by generating hypotheses, models or theories, by collecting empirical data, modeling and analysis of data. Additionally qualitative research is examination, interpretation and analysis of observations in a manner that does not include any mathematical models. This thesis has been conducted taking into account aspects of a quantitative strategy. [*ibid*]

An additional way to divide different research methods is into empirical and theoretical studies. Empirical research collects knowledge by direct and indirect experience and observations. Theoretical research in turn investigates the writings, theories and ideas of others without having any contact with the research area itself. It should be mentioned that it is not possible to conduct an empirical research without understanding the theoretical basis of the research area. This supports having a chapter for theoretical aspects in this thesis as this will strengthen the understanding of the topics for anyone not very familiar with the subject. [*ibid*]

4.2 Data collection

To conduct the analysis featured in this thesis data was collected in two ways: firstly by literature search to assemble a sufficient theoretic foundation and secondly by collecting empirical data to use in the analysis itself.

4.2.1 Literature search

The literature chosen to support the thesis includes information mainly about web applications and penetration testing but also investigating the company behind the analyzable data and their methods of penetration testing. The literature search was used to help build the theoretical framework to support the thesis. The research of this literature was mainly done by accessing different article databases through the Tallinn University of Technology library homepage, the internal library of Clarified Security OÜ, the OWASP project homepage and its subpages, using search engines like Google.com and recommended literature from courses previously taken during the Masters studies.

4.2.2 Empirical data

The literature research was followed up by collecting empirical data for the analysis by working through all available Clarified Security OÜ web application penetration test reports and compiling a generalized data table based on vulnerabilities from these reports. The data collected was strictly anonymous, as all security tests are protected by NDA agreements. There were several reports that the author did not have access for, hence the data from these reports was collected by other employees of the company who were included in the original NDAs. The company has been performing security testing in Estonia since Q3 2011. So far, they have tested over 60 web applications and conducted several security tests for networks, hardware and software.

In the reports every vulnerability was marked with a severity and this was also marked in the summary table. The table includes only the most severe rating, in case a vulnerability was tested or found more than once.

To compare the results of the analysis of Estonian web applications to the typical vulnerabilities from around the world, the author researched different recent reports and presentations on international vulnerability statistics that were made available on the Internet. According to these observations an abstract was created to exemplify the current most common vulnerabilities.

4.3 Quality and reliability of the data

To provide highest quality of information for the theoretical aspects and the following chapters, the author used several different sources for all topics covered. Many areas were thoroughly discussed in several articles on the Internet but were not included in the literature available for the author. For those topics the author investigated a wide array of different authors, timeframes and approaches to provide the most adequate information. Even though some of the used materials and literature might have been slightly outdated for the research area, the author was able to find alternative sources with more recent publications to support the possibly outdated information.

The author believes that the security test reports used to compile the data for this thesis give an adequate overview of the Estonian web application market. The company that provided the data is one of very few organizations in Estonia who offer penetration testing services not only internationally but also for the local market. The company employs a handful of certified experts who have been conducting security tests for several years already and their combined experience and testing resume is remarkably bigger than that of any of their competitors. Hence currently there is no other alternative data source to provide similar information on the vulnerability statistics for the Estonian applications. The author is aware that it is possible that not all vulnerabilities were detected in the tested applications. The testers have confirmed that the sample included several tests that should have had a longer testing duration for all vulnerabilities to be found and reported. Nevertheless the data included in the obtained sample gives a decent idea on the current status of the Estonian web applications in general.

5 Empirical data

This chapter presents the relevant empirical data collected for the thesis. It gives a more thorough overview of the data itself and provides a base for understanding the analysis following in the consecutive chapter.

5.1 Data description

As mentioned previously the data used in this thesis is based on penetration tests conducted by the team of Clarified Security OÜ between Q4 2011 and Q1 2014 (from 01.10.2011 to 30.03.2014). During this time the company conducted 60 security tests which lasted for a total of 3929 hours. Of the mentioned dataset 51 were applications that were developed for the Estonian market, and during those tests a total of 3433 work hours were spent. The nine projects that did not qualify to be used for this data were tests on software, hardware and web applications outside Estonian market.

On average the company conducted roughly five tests a quarter and spent an average of 343 work hours on these tests. The quarter that the test is placed on is determined by the date of the final report. The distribution of tests between quarters is represented in table 5.1.

Quarter	Number of penetration tests	Work hours
Q4 2011	2	106
Q1 2012	3	242
Q2 2012	2	230
Q3 2012	3	280
Q4 2012	7	370
Q1 2013	3	208
Q2 2013	7	524
Q3 2013	8	366
Q4 2013	10	774
Q1 2014	6	333
Average	5,1	343,3
Total	51	3433

Table 5.1 Distribution of penetration tests per quarters

The author has no distributable or specific information about the reports that this data is based on.

5.1.1 Test level statistics

The tests of the sample data were all tested according to OWASP ASVS Level 2 or higher. One of the tests had an opportunistic and selective approach, based on Level 2, that had a duration of under 15 hours and was meant as a quick check for the application in question. Vast majority of the tests were done based on OWASP ASVS Level 2, which consisted of a combination of Level 2A and Level 2B checks. A little more than one fifth of the tests were Level 2A or manual penetration testing verifications and three were Level 3 verifications. Visual dispersion of tests between different ASVS levels is illustrated on the figures 5.1 and 5.2 below.

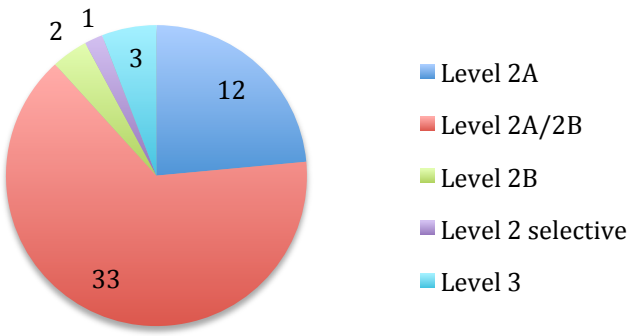

Figure 5.1 Distribution of tests according to OWASP ASVS Levels

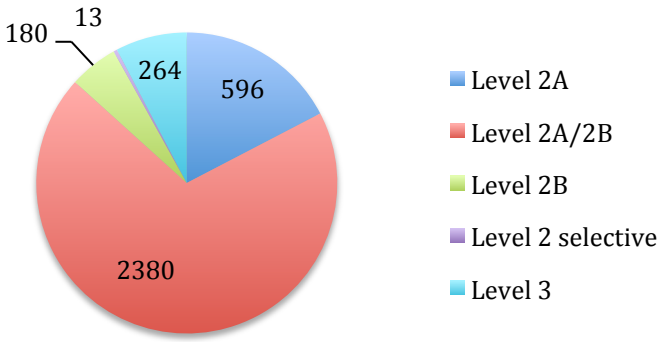

Figure 5.2 Distribution of test hours according to ASVS levels

5.1.2 Test duration statistics

The duration of a penetration test is determined according to the chosen methodology but also according to the amount of web applications functionality and its complexity. It should be noted that that the test duration was always initially recommended by Clarified Security OÜ but the final decision of the length was made by the client. Hence some security tests should have had a longer duration and for those some vulnerabilities might have been left unmentioned.

The average length of a penetration test conducted by Clarified Security OÜ during this timeframe was 67,3 hours. The average length for a Level 2A test was 49,7 hours, 90 hours for a Level 2B test and 72,1 hours for a Level 2A/2B .

Besides the previously mentioned opportunistic short test, the shortest contractual test had the duration of 18 hours and the longest one lasted for 206 hours. In total the conducted tests had 27 different durations. A little more than 50% of all tests lasted between 50 and 100 work hours with a total count of 26 and gross duration of 1921 hours. The average length of a test from that category was 73,9 hours. 35% of the tests were shorter than 50 hours with a total count of 18, a total duration of 606 hours and an average duration of 33,7 hours. There were 7 tests that lasted over 100 hours, totaling at 906 hours and averaging at 129,4 hours. The distribution of different tests according to durations is illustrated on figure 5.3.

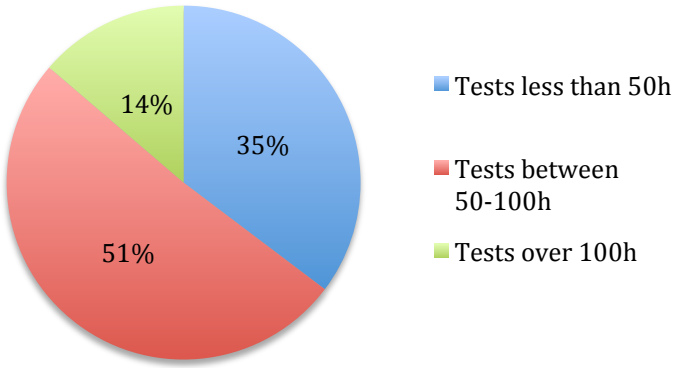

Figure 5.3 Distribution of test durations

Most favored length for a penetration test was 80 hours, followed by test durations of 40, 48 and 56 hours. All other lengths were utilized only once or twice. Exact dispersion is pictured on figure 5.4.

Based on this information the test results should be grouped according to duration rather than using individual durations when performing the analysis.

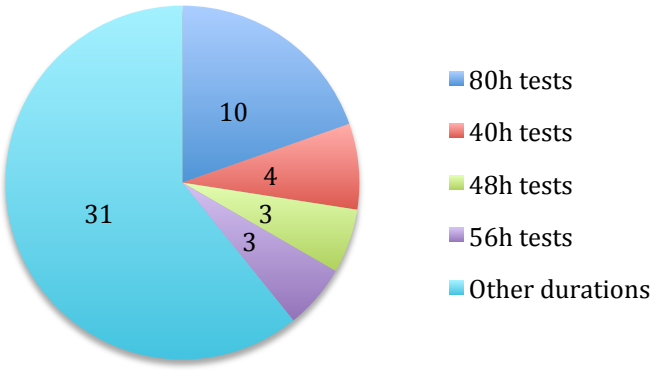

Figure 5.4 Most common test lengths

5.2 Thoroughness of the tests

Chapter 5.1 mentioned that the web applications were all tested accordingly to OWASP ASVS levels, which were very shortly mentioned in chapter 2.3.1. The following sub-chapter goes into a little more depth about the extent of these different levels to introduce the thoroughness of each conducted test. As Clarified Security OÜ only performed tests on OWASP ASVS Levels 2A, 2B, the combination of the aforementioned and 3, only these levels will be discussed. The full extent of these levels is graphically presented in appendix 2. It should be mentioned that even though V11.7 (CSRF) is listed as a Level 3 vulnerability in OWASP ASVS 2009, the team of Clarified Security has decided to include it in all of their Level 2 tests as the vulnerability is very widespread and poses a remarkable threat to any application that uses a login functionality [31]. This decision is supported by the fact that in the OWASP ASVS 2013 edition this vulnerability has been included in Level 2 verification [10].

5.2.1 OWASP ASVS Level 2A

The OWASP ASVS Level 2A or manual testing includes checking the web application for 56 verification requirements that are filed under 11 main categories. Additionally the Clarified Security OÜ team tests for V11.7 under this category.

Clarified Security OÜ recommends a “full range” OWASP ASVS Level 2A security test as the minimum level of web application testing. During this test all the required verification requirements are manually checked and therefore ensured that none of the vulnerability categories are overlooked. Although this test is manual and consequently very time consuming, it guarantees that all of the applications functionality and parameters are checked and all relevant verification requirements applied. All used testing techniques are adopted to suit applications custom-developed solutions. The testers typically use a “black box” approach to go through the whole applications by using “web proxy” type tools, intercepting and modifying all the traffic between the browser and. This method gives the testers a thorough understanding of all of the applications hidden functions and business logic.

On this level the testers are provided with same access to the systems as a random application user. Hence, all the vulnerabilities that are detected during the test could possibly be found and exploited by an user with malicious intent. By testing an application on this level the owner is reassured these typical vulnerabilities are found by a neutral party, reported and fixed before someone could utilize them.

5.2.2 OWASP ASVS Level 2B

The OWASP ASVS Level 2B or manual code review includes checking the web application for 88 verification requirements that are filed under 11 main categories. Compared to Level 2A, it does not require checking of three requirements and in turn adds 32 different verifications which are mostly related to code and code review. Additionally the Clarified Security OÜ team tests for V11.7 under this category.

Testing on OWASP ASVS Level 2B gives the penetration test a more “gray box” or even “white box” approach. Reviewing the applications source code provides the application owner with some measure of code quality as the verifiers check for use of best practices, coding style, maintainability and code efficiency. Serious application functionality bugs have also been detected during this kinds of testing. The “black box” approach, mentioned

in the previous chapter, is also generally used for Level 2B testing to assure that the application is not noticeably vulnerable to outside attacks by users with malicious intent.

5.2.3 OWASP ASVS Level 2A/2B

The OWASP ASVS Level 2A/2B includes checking the web application for a total of 91 verification requirements that are filed under 12 main categories. This level combines the requirements for Level 2A and Level 2B (including V11.7) and does not add any other requirements.

5.2.4 OWAST ASVS Level 3

The OWASP ASVS Level 3 includes checking the web application for 109 verification requirements that are filed under 13 main categories. With 18 additional requirements it is a noticeable step up from a Level 2 test.

A OWASP ASVS Level 3 test is usually chosen by customers who are very critical of their security and have applications in a sector that is easily "monetizeable" like e-banking or high volume e-shopping. A Level 3 test focuses more on the security aspects that are considered to be "behind the scenes" like looking at logs and stringent security verification requirements.

5.3 Verification requirement area statistics

When discussing the initial statistics based on the data with Clarified Security OÜ employees, they stressed that results of the tests are not that straightforward and that for some vulnerabilities exceptions should be made. There were vulnerabilities in the reports that were marked under a certain verification sub-point but which would also have fit under other sub-points in the same category or that were considerably similar. The remark was made with an emphasis of verification category V4 - Access Control Verification Requirements, where points V4.1, V4.2, V4.3, V4.6 and V4.7 represent very comparable vulnerabilities. It was brought to the authors attention that if one of those vulnerabilities

was found in an application, the application often also included one or more of the others. But when any one of them was found, the finding was reported and further findings from any of the previously mentioned categories were often left unreported. [31] At this point it should be reminded that in any case it is highly unlikely that a penetration tester is able to identify all the vulnerabilities in the system [16].

Thus the author decided to not only analyze according to individual findings but also conduct an analysis that would characterize vulnerability findings according to OWASP ASVS verification areas. For this, all individual findings under different categories were added together to create a dataset where each verification requirement category was represented by a total number of findings in this category. To also take into account the severity levels of the findings of each category, that data was also conserved.

Conducting this generalized analysis of different OWASP ASVS verification requirement areas will show which kinds of vulnerabilities have been most prevalent and critical in Estonian web applications. The results will provide important knowledge of the areas that the applications are most vulnerable in and which should be put under special attention by both the clients and developers.

5.4 Typical vulnerabilities according to other reports

Even though vulnerabilities in web applications used in Estonia have not yet been thoroughly studied, the security situation of web applications worldwide is persistently investigated and analyzed. Different organizations and security companies are constantly publishing materials and to raise awareness of the subject.

The author was able to obtain various papers, published mostly in 2013, that described the situation of vulnerabilities in international web applications. Several of the papers, that are briefly analyzed below, were used as collaboration data for the OWASP Top Ten [28].

The datasets of the materials were all remarkable but varied a great length. One of the most extensive ones was the Trustwave report, which has based their data on more than five million malicious websites, nine million web application attacks and more than 2500 conducted penetration tests [34]. Also remarkable is the dataset used for the WhiteHat Security report that included 15 000 production and pre-production websites from 650 different organizations [5]. Other papers did not include information about their dataset sizes or based their analysis on a datasets on a sample of less than 5000 websites [34, 35, 36, 37]. Even though most of the materials do not state the country of origin for their statistics, a few state that majority of the data was collected from US-based web applications [5, 36].

According to the materials a very high percent, ranging from 86 to 99 depending on the report and averaging at 94%, of web applications are put into production with at least one serious vulnerability.

When talking about the types of the vulnerabilities then all mentioned that XSS is one of the most frequently occurring. The current discovery rate of the vulnerability averages roughly at a little over 50%, meaning that every other web application is attackable by using XSS.

Other vulnerabilities that were mentioned as top problems in several papers were information leakage, content spoofing, authorization problems and CSRF. Most of those areas are known problem areas and characterized by different OWASP ASVS verification points. SQL injection was also often mentioned but more of as a warning, that having that vulnerability in an application is potentially a huge risk, but at the same time it is not a vulnerability that is often detected.

6 Analysis and discussion

The aim of this chapter is to provide information on the vulnerabilities found in Estonian web applications by thoroughly analyzing the data described in Chapter 5. The data is analyzed as individual OWASP ASVS verification requirement points and then by arranging the verification requirements by their main categories. The data is firstly analyzed by only taking into account the overall prevalence of the vulnerabilities and not paying attention the risk levels appointed to them, if not mentioned otherwise. Then the analysis is continued by analyzing prevalence of different severity rankings and afterwards additional risk ratings are appointed to the findings according to the severity ratings of these findings in penetration test reports. Afterwards the analysis is conducted according to the three duration classes of the tests of up to 50 hours, from 50 to 100 hours and over 100 hours and later very briefly according to different test levels. The chapter is finished by a section briefly describing the most notable findings and then ends with a conclusion of the analysis.

To better understand the severity of vulnerabilities, which were detected and declared a higher threat by the testers, the author decided to create a simple risk rating of multiplying the value of vulnerability finds that were higher then the level “low”. To determine a plausible coefficient the author sent out inquiries to several IT security professionals [38] and compiled a rating based on their feedback. According to this rating the vulnerabilities that were rated as “low” were left as they were with a rating of 1 and provided “medium” finds double value compared the lower rating. In turn, “high” finds were decided to be rated four times more dangerous compared to “low” risks and double in threat compared to “medium.” “Critical” finds were deemed eight times more significant than the findings that were provided with a “low” rating and in turn double in value compared to “high” finds. The figurative table of multiplications follows.

LOW	x 1
MEDIUM	x 2
HIGH	x 4
CRITICAL	x 8

Table 6.1 The multiplication of findings with higher level ratings

This modification helps to give the analysis more depth as it gives more weight to vulnerabilities with higher ratings. Therefore a vulnerability which has been found in five different applications and has been given a rating of "low" in all of the reports, would get a total rating of 5. In turn a vulnerability that has been found in three applications and has been stated as "low" in one, "high" in another and "critical" in the third, will be given a total rating of 13. So even with fewer findings this method clearly illustrates that the second vulnerability is more serious and bigger of a threat than the first one.

6.1 Analysis of individual verification requirements

The maximum possible vulnerabilities that could be found during a Level 3 penetration test is 109. This number is presented as a maximum as that is the highest security level test that the company has conducted to date. The gross total of all different detected vulnerabilities of all the tests by Clarified Security OÜ was 72 with 447 detached occurrences throughout. For example of Level 2A/2B tests, where the maximum possible vulnerability count would be 91, the count of total vulnerabilities found was 64 with 300 instances. The average number of vulnerabilities per security verification point was 6,2. The overall table of all findings can be found in appendix 3.

6.1.1 General prevalence

Of all tests overall the most frequently found vulnerability was V6.1 or, more simply said, XSS. There were 35 reports that included one of more vulnerabilities from this class. The runner-up in general prevalence was V11.7 (CSRF) with 30 mentions. Last place in the top 3 was obtained by V8.1 which stands for displaying sensitive data in error messages or stack traces. There were a total on 13 vulnerabilities that had more than ten occurrences throughout all the tests and they can be seen in figure 6.1 below.

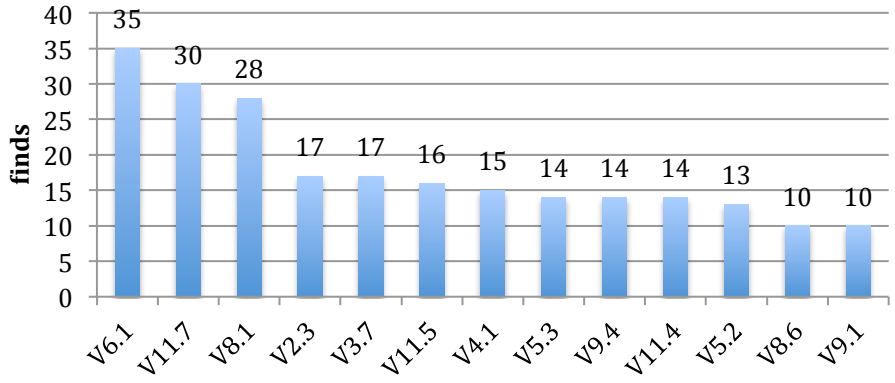

Figure 6.1 Vulnerabilities with more than ten occurrences

To better illustrate the need for a ranking system for assessing the threat caused by a vulnerability it should be said that out of the 28 finds under V8.1, 78,6% were ranked as "low" by the testers. At the same time of the findings under V4.1, which was placed seventh in the prevalence ranking, only 13% were "low" findings and at the same time 33% were deemed "high" and 33% "critical."

6.1.2 Severity rankings by the testers

Out of all the reports nine different verification points were deemed as "critical" with a total of 15 finds out of all the tests. The average amount of "critical" findings in all security verification requirement points was 1,6. There were 30 verification points that were classified as "high" and 38 points that were classified as "medium" with both having a little more than 100 total findings. Average number of "high" findings was 3,4 and the average of "medium" was 2,7. The most frequent vulnerability category was "low" with 61 points that were deemed that ranking and over 220 occurrences. The average amount of "low" findings per category was 3,7. The count and sum of findings can be seen in table 6.2. It should be mentioned that the count of vulnerabilities does not add up as there are overlapping vulnerabilities in all the categories.

	Total	Critical	High	Medium	Low
Count of vulnerabilities	72	9	30	38	61
Sum of vulnerabilities	447	15	102	104	226

Table 6.2 Severity rankings with the highest prevalence

The vulnerability that had the most "critical" ratings was V4.1 with five of its 15 mentions being ranked accordingly. For the same reason, it placed third in the top of highest rated vulnerabilities. Other verification points had two or less "critical" rankings. The vulnerability with the highest count of "high" ratings was once again V6.1 with a total number of 22 occurrences and is followed up by V11.7 with 18. Other points were ranked as "high" six or fewer times.

The top for the ranking "medium" is a bit more uniform compared to the two previous ones in the sense that the differences between the highest counts are a bit more subtle. Most popular occurrences are similar to the leaders from the "high" ranking with V6.1 leading with ten occurrences and V11.7 tailing with nine. Third most detected vulnerability was V2.3 with eight finds.

Although the sum of all vulnerabilities for ranking "low" was remarkably greater than for all other rankings, the high end of their found vulnerabilities was not very different compared to others. The most prevalent vulnerability with the "low" ranking was V8.1 with 22 instances followed up with two vulnerabilities with 13 (V11.5) and 12 (V11.4) occurrences. The most mentioned ASVS findings are listed as tables in appendix 4.

6.1.3 Risk ratings

When providing all the findings with risk ratings, as explained in the introduction, the top of vulnerabilities with the highest risk changes a little. Of the rated vulnerabilities, four out of the top five are included in the list of vulnerabilities with more than ten occurrences which was presented in figure 6.1. To illustrate the formation of these ratings the top five of highest rated vulnerabilities are presented below in table 6.3.

Verification	Ranking	Critical	High	Medium	Low
V6.1	111	0	22	10	3
V11.7	93	0	18	9	3
V4.1	68	5	5	3	2
V8.1	44	1	2	3	22
V4.3	35	2	4	1	1

Table 6.3 Highest rated vulnerabilities

As it can be seen the most prevalent vulnerability, V6.1, was found in 35 different applications. It was rated as "high" in 22, "medium" in ten and "low" in three applications. So the method gives V6.1 a rating of 111. Vulnerability V4.1 (access for only functions for which the users possess specific authorization for) places third. This ranking is caused by the amount of "critical" and "high" ratings in the reports. The vulnerability V4.3 (access for only authorized data files) was not featured in figure 6.1 as it only had a total of eight occurrences throughout all the tests, but as it can be seen in table 6.3, six of those instances were rated as vulnerabilities with ratings higher than medium - four were rated as "high" and two even as "critical." Therefore it has a remarkably high rating and is placed in the fifth place of overall vulnerabilities.

6.1.4 Test durations

As the dispersion of tests according to their levels is very unequal, with Level 2A/2B tests forming majority of sample, the author decided to conduct an analysis of the tests based on different duration ranges. The initial analysis showed that the most common vulnerabilities seem to remain the same despite the method of analysis and the author believes this brief analysis will confirm this hypothesis.

Vulnerabilities of tests under 50 hours

The sample of data based on penetration tests which had a duration of less than 50 hours included a total of 18 reports. In this sample the total of detected ASVS vulnerabilities was 53 out of 91 as the highest test level in this sample was a full Level 2.

When counting the overall findings of this data, the maximum count is achieved by two vulnerabilities. Both V6.1 and V11.7 share this position with 10 instances. V10.3, V3.7 and V11.4 all have 6 findings and can therefore be considered second most popular.

In case of using the risk rating methodology described earlier, the highest scorer is once again V6.1 with 28 (5 “high”, 3 “medium” and 2 “low” ratings) followed by V11.7 that achieved a score of 24. The third placing vulnerability was V10.3 (verification of usage of TLS services for all connections that are authenticated or involve sensitive data) with 22 points. Other vulnerability verification points got a rating score of 19 or less.

Vulnerabilities of tests from 50 to 100 hours

The second sample was the largest and was composed based on a total of 26 test results. The highest level test in this sample was a Level 3 so the maximum of possible vulnerabilities for this sample was 109. Out of these 56 different vulnerabilities were detected.

Similarly to the previous sample two verification points tied for most frequent finds in this sample with both V6.1 and V8.1 having 18 findings. V11.7 places third with 16 findings.

By risk rating V6.1 comes first once again by having a total score of 57 based on 18 finds and is followed by V11.7 with 53 based on 16 finds. Third placer is V8.1 with a significantly lower score of 29 from 18 finds. This clearly shows that V6.1 and V11.7 have noticeably higher ranking vulnerabilities in this sample.

Vulnerabilities of tests of over 100 hours

The last sample includes data from a total of 7 tests. In this sample the total amount of different vulnerabilities found was 46 out of a Level 3 maximum of 109.

The most frequently found vulnerability was once again V6.1 which could be found in the reports for all of the seven tests. V2.3 (account lockup after too many authentication attempts) was present in six and V8.1 in five.

According to the risk rating methodology the top three risks were the same as according to prevalence analysis. The highest scorer was once again V6.1 with a rating of 20, followed by V2.3 with 14 and V11.7 with 12.

6.1.5 Test levels

Although the distribution according to levels is pretty uneven, a short analysis should be conducted nevertheless. After the initial analysis it can already be said that there is no question about the most common threat. When looking at the penetration tests results distributed by levels the frontrunner in all distributions is still clearly V6.1, both with prevalence and according to the risk rating. The author decided to list vulnerabilities according to their risk rating first and when necessary, additionally according to total count. This listing was needed as there were a few vulnerabilities that had the same total count of findings which were both aiming for third position in the top.

As already mentioned previously the V6.1 vulnerability is dominant in all samples - in Level 2A with a count of eight out of 12 and rating of 22, in Level 2A/2B with a count of 23 out of 34 and a rating of 75 and also in all other levels with a count of five out of six and a rating of 16.

Based on these two values the V11.7 vulnerability placed second almost unanimously. Out of 34 Level 2A/2B tests it was detected in 22 and had a threat rating of 65. It was also found in six out of the 12 Level 2A tests and had a total score of 20. In the combined statistics for other levels this vulnerability placed third with three mentions out of six and an overall rating of nine.

The third vulnerability with the highest risk rating was V8.1 for Level 2A/2B with 18 findings and a rating of 30. For combined data from other level tests this vulnerability places second with five findings and a overall rating of nine. For Level 2A tests the third vulnerability with the highest risk was V4.1.

6.1.6 Non-ASVS vulnerabilities

There were two vulnerabilities that were mentioned in a notable amount of test reports but which are not in the list of official ASVS vulnerabilities. Both of these were present in a similar amount of reports in total but both had a rating of "low" in all. More prevalent was the vulnerability of exposing server version numbers which was reported in 26 reports. Frameable response vulnerability was found in 25 applications. As they both only had mentions with the level "low," their calculated ranking remained the same as the total of findings. There were 24 other non-ASVS vulnerabilities reported throughout the data but all of these totaled in five or less instances.

6.1.7 Conclusion

The initial analysis confirms that the overall situation with Estonian web applications seems to similar to what foreign reports and papers are saying about the situation elsewhere, atleast concerning the most common vulnerability.

XSS is a dominant vulnerability in all of the different data samples used, both by overall prevalence and also according to the simplified risk rating used in this thesis. CSRF, which was also often mentioned in foreign reports, comes in as strong second. Third placing vulnerability differs according to the used sample - V8.1 if looking only at prevalence, V4.1 when using the risk rating system. In case of using the samples grouped by durations or levels, V6.1 and V11.7 still clearly overrule all other verification points, while V8.1, V2.3, V10.3 and V4.1 are mentioned for some samples.

The vulnerability with the highest number of "critical" finds was V4.1. V6.1 had the most "high" and "medium" rated vulnerabilities and the vulnerability with the most "low" findings from the ASVS checklist was V8.1. If also taking into account the notable non-ASVS finds, then V8.1 is left third after the findings of vulnerabilities of exposing server version numbers and frameable responses.

6.2 Analysis according to verification requirement categories

After sorting the findings according to the OWASP ASVS security verification categories, surprisingly, rather different trends are outlined. The 447 detached occurrences are now divided between 13 verification requirement categories. The average count of vulnerabilities per category is 32. When also looking at the severity rankings for the vulnerabilities provided in the security test reports the average number of "critical" finds per category is 1,1, while "high" and "medium" findings are pretty equal with the averages of 7,2 and 7,4. There are averagely 16,1 "low" finds per category. The full table with grouped vulnerabilities can be examined in appendix 6.

6.2.1 General prevalence

According to general prevalence the most present vulnerability category in the tested applications is V11 (HTTP Security Verification) with a total of 77 findings. V3 (Session Management) is remarkably placed second with 67 findings and V2 (Authentication) third by 52. The explanation of V3 placing so high in the ranking is caused by the amount of "low" findings in the category - the 40 findings constitute almost 60% of the total. All rankings can be seen in figure 6.2.

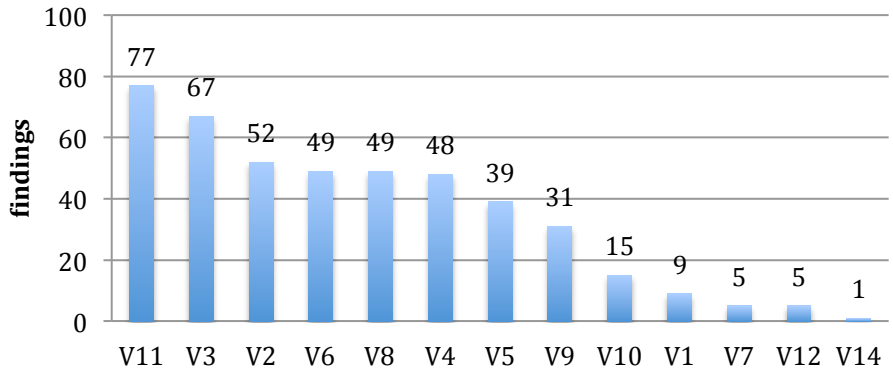

Figure 6.2 Prevalence of findings grouped by OWASP ASVS categories

6.2.2 Severity ratings by the testers

To further conduct the analysis the severity ratings that were assigned by the penetration testers should also be taken into account. This part of the analysis only includes prevalence of these specific ratings. The category with the most “critical” findings is V4 with nine findings, followed by V6 and V2 with two findings. The highest number of findings with the ranking “high” was 26 and all those vulnerabilities belong to verification category V6. V11 places second with 19 findings and V4 third with 16.

V3 had the biggest occurrence of “medium” findings with 18. V11 also had a remarkable number of findings with 17 and V6 was third with 15. The most modest security ranking was once again numerically the largest with the first placing category of V11 having 41 findings. Runner-up was V3 with 40 and V3 placed third with 32 findings. These most mentioned ASVS findings per category are listed as tables in appendix 5.

6.2.3 Risk ratings

After providing all categories with risk ratings, as was done while analyzing the vulnerabilities individually, the highest ranking category is V4 (Access Control). It should be mentioned that the different vulnerabilities listed in this category were one of the main reasons for also conducting an alternative, categorized, analysis of the data. V6 (Output Encoding/Escaping), which requirement V6.1 was an absolute frontrunner in individual analysis, also placed high. The category for the other frontrunner of individual analysis, V11, also placed in the top three in this ranking. The vulnerability categories with the highest risk rankings are illustrated on figure 6.3.

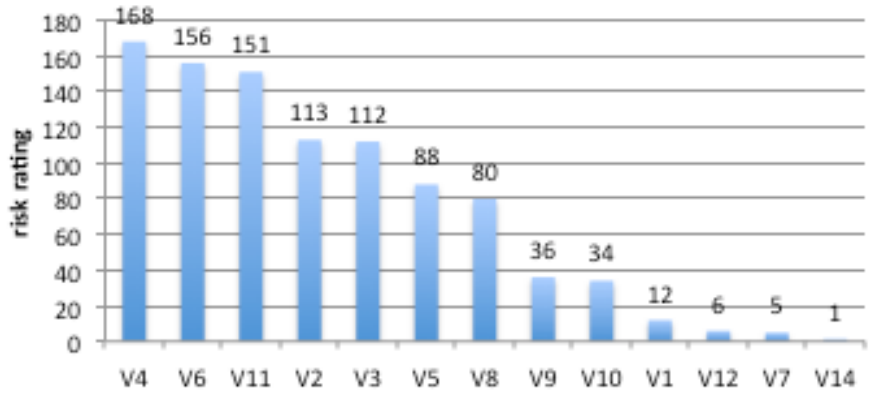

Figure 6.3 Findings according to risk rating grouped by OWASP ASVS categories

To better illustrate the difference between these two methods, the various dispersions of vulnerability rankings are now presented. Firstly, the figure 6.4 pictures the top five of general prevalence and demonstrates their dispersion of vulnerabilities with different severity rankings. Secondly for comparison figure 6.5 presents the top five categories according to risk ratings while using the same layout.

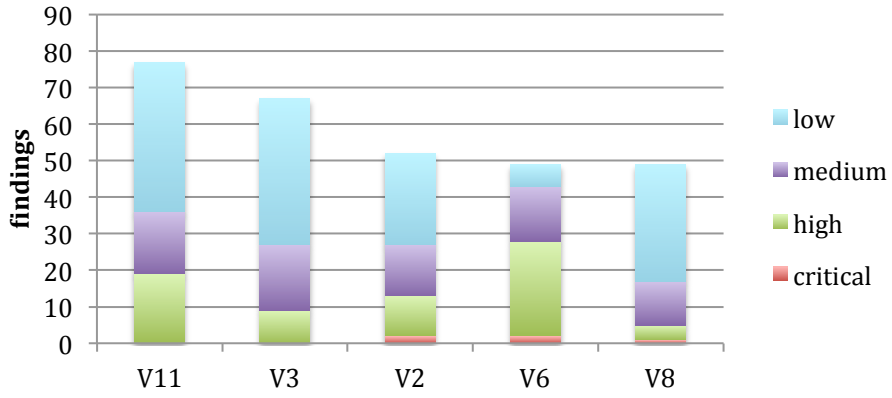

Figure 6.4 Dispersion of severity rankings in most prevalent ASVS categories according to vulnerability count

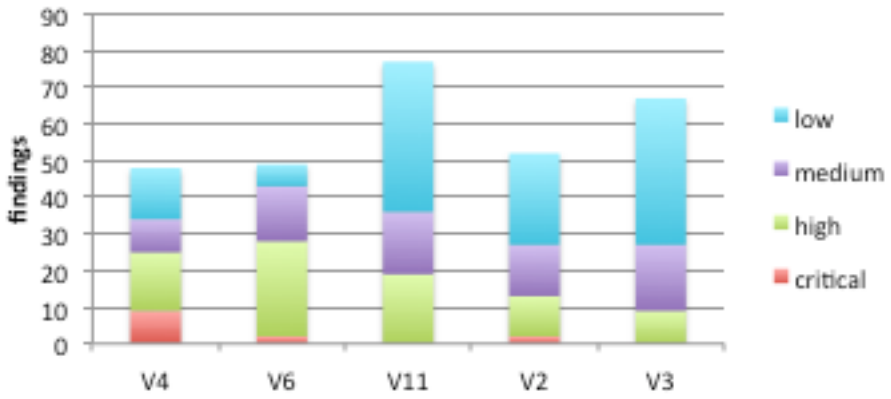

Figure 6.5 Dispersion of severity rankings for ASVS categories with the highest risk rating according to vulnerability count

It can be clearly seen that by using these two methods the verification requirement categories with the highest risk ratings almost do not overlap with most prevalent ones. The exceptions here are V11 and V2 which are present in both figures. To illustrate the differences in these two samples even better figures 6.6 and 6.7 present the same categories as in figures 6.4 and 6.5 but by previewing them according to the vulnerability risk rating system used in this thesis.

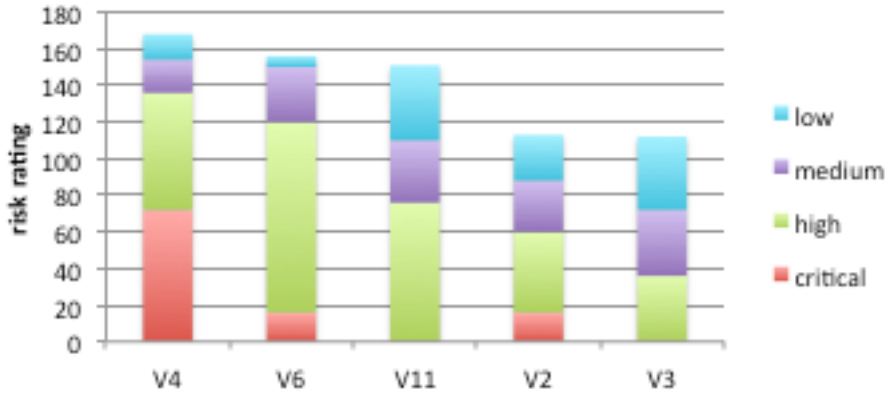

Figure 6.6 Dispersion of severity rankings for ASVS categories with the highest risk ratings

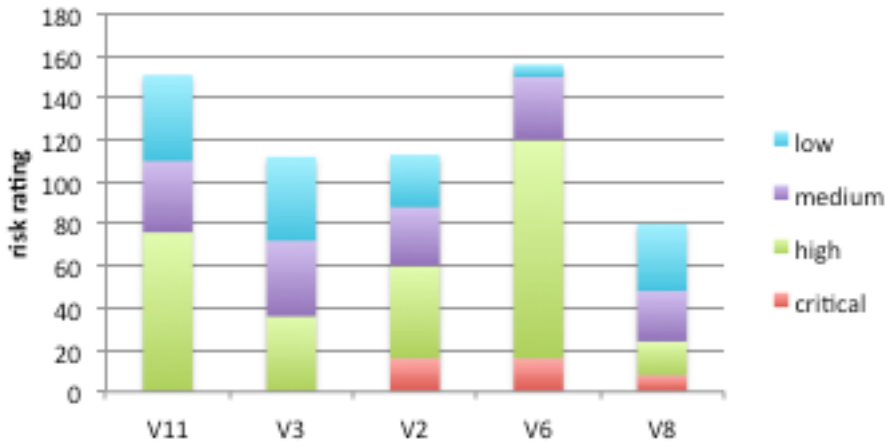

Figure 6.7 Dispersion of severity rankings in most prevalent ASVS categories according to the risk rating

6.2.4 Test durations

To conclude the analysis based on the grouped data it will now be analyzed according to the same three test duration ranges that were used earlier. Similar pattern of analysis is used as was used for precedent individual analysis.

In the sample compiled based on the tests with durations of less than 50 hours the most prevalent category was V3 with 24 findings, followed up by V2 and V11 which both had 15 findings. According to the risk rating the highest scoring verification category is V6 with a score of 59 points. Although its prevalence was only 14, this score is reached thanks to the presence of two "critical" and seven "high" findings in the sample. V4 is ranked second with a score of 54 and V11 third with 40.

Of the results of test that lasted between 50 and 100 hours the highest prevalence was 38 which was reached by V11. V3 is second with 30 findings and V8 third with 27. According to risk ratings V11 scores highest with a rating of 80. V6 scored 73 and V4 59.

In case of tests that had a duration of over 100 hour the prevalence top was very even. Two categories had 15 findings (V11 and V2) and two 13 (V3 and V4). The frontrunner of risk

rankings in this sample was V4 with a score of 55. V2 is placed second with 36 and V11 and V3 share the third position with 31.

6.2.5 Test levels

When looking at the grouped results then the highest rated categories were V6 for Level 2A tests with a rating of 28 and a total count of 10 different findings, V4 for Level 2A/2B tests with a ranking of 119 and a total count of 30 and V4 again for other levels with a ranking of 23 and a total count of 6.

For Level 2A tests V4 achieved second position with a rating of 24 and prevalence of 7 and V5 ended up third with the same prevalence but by a score of 17. In case of Level 2A/2B tests V6 got an impressive rating of 112 which placed it as second even though it had the highest findings count of 34 in the category. V5 was third with a score of 66 and a prevalence of 18.

For other categories V6 (rating 16, prevalence 5) and V11 (rating 15, prevalence 11) were also noteworthy.

6.2.6 Conclusion

The analysis of the data grouped according to OWASP ASVS categories gave remarkably different results than the analysis conducted by using individual vulnerabilities. When observing the results given by the different types of analysis, no clearly dominant vulnerability category appears. By that conclusion it could almost be said that all categories, that were listed as top three categories during different analysis phases, could be considered rather hazardous.

The category of V11 does still appear a bit more frequently then others. It occurs, however, mostly in third place according to rankings. The category is mentioned in all the different analysis types, only lacking in the sample with most “critical” findings as the highest rating for V11 findings was “high.” It should also be mentioned that more than 53% of all findings

in this category were ranked as "low," hence even though rather prevalent, the vulnerabilities in this category should not be considered a very remarkable threat.

The categories of V4 and V6 were mentioned less than V11 but almost equally often compared to each other. Still, both should be regarded more alarming than V11. 57% of findings in V6 and 52% of findings in V4 were ranked "high" or "critical." Between them V4 can be considered riskier as 19% of the findings in that category were ranked as "critical." Both categories got several highest mentions in top three-s of different analysis phases according to their prevalence and risk. The author believes that the appearance of V6 in this part of the analysis should be emphasized as its verification point V6.1 (XSS) was a noticeable frontrunner when analyzing the individual findings separately.

6.3 Notable requirements and categories

The preceding analysis reveals that there are some specific vulnerability requirements and verification categories that are worthy of a more thorough introduction. In the following subchapter the author briefly presents the most notable individual verification requirements and the most mentioned verification categories and explains why they consistently exist.

6.3.1 Individual verification requirements

The analysis clearly shows that there are two individual vulnerabilities that overshadow all others in the sample and remain dominant even when looking at the different sub-samples. The trend is also not dependent on the chosen method of analysis. This sub-chapter introduces these vulnerabilities a bit more thoroughly, explains the possible attack vectors that they can provide for the attackers and gives a short explanation how to mitigate these vulnerabilities.

Cross-Site Scripting

The OWASP ASVS verification point V6.1 description asks for verification of all untrusted data, that is output to HTML (including HTML elements, HTML attributes, JavaScript data values, CSS blocks and URI attributes), is properly escaped for the applicable context [11]. Commonly these kinds of vulnerabilities are known as Cross-Site Scripting flaws or just XSS for short. There are three different kinds of XSS: stored, reflected and DOM based [27].

This vulnerability occurs when an application sends untrusted data to a web browser without properly validating or escaping it. Without proper output escaping or validation, such input might be treated as active content in the browser. Although most XSS vulnerabilities occur originating from user inputs, the text-based attack scripts can also be sent from internal sources such as data from a database. XSS provides attackers with possibilities to execute scripts in the users browser which in turn can deface sites, insert hostile content, redirect users to malicious sites or even hijack user sessions. [*ibid*]

The preferred option for mitigating the occurrence of XSS is to escape all untrusted data based on the context that the data is placed into. An alternative but partial defense method is using whitelisting for input validation. It is only a partial solution as many applications require using special characters. Using auto-sanitization libraries or use of content security policy can also be used as mitigation methods against XSS. [*ibid*]

In OWASP Top Ten 2010 the XSS vulnerability placed second [26] and in the 2013 edition the vulnerability was placed third [27]. XSS can also be considered an injection attack and as that it is placed first in both OWASP Top Ten 2010 and 2013 [26, 27].

Cross-Site Request Forgery

The OWASP ASVS verification point V11.7 describes the mechanism required to defend an application against Cross-Site Request Forgery attacks. The point asks for verification that the application generates a strong random token as part of all links and forms associated with transactions or accessing sensitive data, and that the application verifies the presence of this token with the proper value for the current user when processing these requests. [11]

CSRF uses the knowledge that that most web applications permit attackers to foresee all the details of a particular action. A CSRF attack obligates a logged-on users browser to send a fake HTTP request to a vulnerable web application. This request includes victim's session cookie and other automatically included authentication data. By obtaining this information the attacker can make the users browser to create requests that the vulnerable applications handles as legitimate requests coming from original user. By using this type of attack the victim can be tricked into performing different operations their account is authorized to do, for example changing or updating their account details or even making purchases. [27]

The easiest way of avoiding CSRF attacks is to add an unique token to each HTTP request that include data provided from the client side as it is important to ensure that the data posted to the server is not of malicious origin. These tokens should differentiate per user session. Everything sent to the server should be forwarded using the HTTP POST method. An alternative solution is to require the user to reauthenticate when performing notable operations. [*ibid*]

6.3.2 Verification categories

There were three OWASP ASVS verification sections that stood out when analyzing the data grouped by general categories. Next these categories are shortly introduced according to their definitions according to the OWASP ASVS Web Application Standard [11] and explained why they are consistently present in web applications.

V4 (Access Control Verification)

The verifications points under this section define how an application can safely accomplish access control. In most applications, access control must be present in various different locations across the application layers. This category construes verification requirements for access controls for URLs, business functions, data, services, and files. There are 15 verification requirements listed under this category. [11] The appearance of vulnerabilities listed under this category is persistent mostly because of incorrect implementation of

authentication verification systems. Access rights are verified insufficiently and in case the forwarded data is formatted as the server expects it to be, it is accepted without further controls. In case of more simpler systems, human error is also a considerable factor. For example when a developer has forgotten to add a necessary clause or condition into the code. [31]

V6 (Output Encoding/Escaping)

The requirements listed under this category ask for verification that output is properly encoded or escaped so that it is safe for external applications. There are ten verification requirements listed under this category. [11] The vulnerabilities listed under this category provide various attack possibilities by using different combinations of encodings, input and defense methods etc. For more complex systems the developer should be highly knowledgeable of all the different variations that these types of attacks provide. As Clarified Security OÜ Web Application Security trainings have proven, when questioned about security problems related with output encoding and escaping, the developers have heard a few related keywords and think they are aware of these types of vulnerabilities and how to avoid them, but in reality their knowledge on the topic is deficient. [31]

V11 (HTTP Security Verification Requirements)

The requirements under this category define what can be used to verify security related to HTTP requests, responses, sessions, cookies, headers, and logging. There are seven verification requirements listed under this category. [11] The prevalence of V11 vulnerabilities is also caused by general lack of knowledge of the importance on these security related problems. [31]

6.4 Conclusion of analysis

The two different methods of analyzing the findings individually and grouped by OWASP ASVS verification categories gave surprisingly different results. When looking at found vulnerabilities individually then certain verification clearly stand out, while when observing the statistics when grouped, the front ranking categories are pretty uniform.

The first part of the analysis clearly shows that both of the used methods and all the different samples outline similar vulnerabilities, when looking at them individually. By far the most frequently mentioned out of the list of ASVS verifications was V6.1 which stands for XSS. This vulnerability was the most prevalent and had the highest risk rating in almost all the different samples used in the thesis. It also had the most "high" and "medium" level finds overall. In two samples (duration up to 50 hours and durations between 50 and 100 hours) the vulnerability tied first with two other frontrunners. The second most mentioned vulnerability was V11.7 - CSRF. Generally it was tailing right after V6.1 according to prevalence and risk rating, but in a few samples it was listed as third most common. From individual vulnerabilities also worth mentioning is V8.1 (outputting sensitive data into error messages) that placed mostly in third place in different analysis.

When analyzing the results for categorized vulnerabilities, it is more difficult to determine whether there is a most common or popular category. The V11 (HTTP Security) verification requirement seems to stand out as it is mentioned slightly more than other categories. But as already mentioned it's rankings in the different analysis were more of on an average side hence making it more of a minor threat. Other more frequently mentioned vulnerability categories in this analysis were V4 and V6. Both of them had higher mentions and were frontrunners when examining samples from different approaches.

According to these conclusions it can be claimed that generally vulnerabilities from OWASP ASVS categories V11 (HTTP security), V4 (Access Control) and V6 (Output Encoding/Escaping) are most common in Estonian web applications. In case information on more specific vulnerabilities is requested, the author proposes OWASP ASVS verification requests V6.1 and V11.7 as most common in Estonian web applications.

When comparing these results to the information that are included in OWASP Top Ten 2013 then the results are not that surprising. As already mentioned the most common individual vulnerability of V6.1 and its category V6 can be considered to file under both A1 (Injection) and A3 (specifically Cross-Site Scripting) sections of the OWASP Top Ten. The

category of V4 is listed under A4 (Insecure Direct Object References) and A7 (Missing Function Level Access Control). V11.7 and its category V11 can be generally filed under A8 (CSRF). This comparison clearly shows that the vulnerabilities that have been deemed most critical all around the world by OWASP are also relevant for Estonian web applications.

6.5 Further research

All published security reports are telling the same things. Web applications are currently certainly one of the most popular targets of cyber attacks. The status of security in web applications in general and around the world is constantly studied and analyzed and every year there are reports published which reflect on the current status of the situation.

Although the Estonian market for web applications is not that big nor prominent, the fact that we are considered to be one of the leading e-countries in the world clearly states that the status of web applications launched for our market should be improved and researched also in the future. This thesis gave a first glimpse and a vague idea of the areas status as of spring of 2014 but it would most certainly be beneficial to re-evaluate the status and vulnerabilities again in a few years. This could show if the developers and clients have learned from their previous mistakes and whether they have implemented means to create more secure applications.

7 Summary

As stated in the purpose of this thesis, emphasis was given to analyze the current status of security vulnerabilities that have been found in Estonian web applications. As a result of various types of analysis the most frequent security vulnerabilities were revealed.

To create a plausible analysis, firstly a comprehensive theoretical overview of the area had to be obtained. This was then followed by a short introduction to Clarified Security OÜ and its activities as a source of the required vulnerability data. Next the collection of empirical data was conducted for the analysis by working through all the relevant Clarified Security OÜ web application penetration test reports and compiling generalized data based on the vulnerability findings of these reports. The dataset of vulnerabilities was based on 51 suitable penetration testing reports that were conducted over the past three years and consisted of 447 individual vulnerability data points which were categorized according to 72 OWASP ASVS verification requirements. This means that findings were collected by specific requirements and multiple findings per verification requirement were treated as one data point. After a brief introduction of the obtained dataset it was then thoroughly analyzed. Firstly grouped by individual OWASP ASVS verification requirements and then by aggregating all vulnerabilities by the main OWASP ASVS verification requirement categories. The analysis for both aggregation levels was conducted overall according to prevalence and by using a risk rating system, by prevalence while taking into account severity levels assigned in the reports, then according to test durations and finally briefly according to test levels.

The results of the analysis showed that the most typical vulnerabilities that were found in Estonian web applications were relatively similar to the vulnerability classes in the OWASP Top Ten and also to the vulnerabilities mentioned in various international web application vulnerability reports. When analyzing individual vulnerabilities at a requirements level then OWASP ASVS verification points V6.1 (XSS) and V11.7 (CSRF) clearly stood out. If looking at all the found vulnerabilities combined under OWASP ASVS main categories then

vulnerabilities from OWASP ASVS categories V11 (HTTP security), V4 (Access Control) and V6 (Output Encoding/Escaping) were most commonly found in Estonian web applications.

This means that the flaws that are current and problematic in other parts of the world are also very relevant to Estonian web applications. Majority of these "most typical" vulnerabilities mentioned in this thesis have been around since the very first websites were made and their common security issues addressed, yet developers still constantly overlook them when writing code or do not even have the knowledge how to avoid them in the first place. At the same time the clients and project leaders apparently are not aware of the seriousness of these vulnerabilities and hence don't point out these areas as something to avoid in the final product.

This thesis provided information on which OWASP ASVS vulnerability verification requirements are in fact most commonly missed in Estonian tested applications and which OWASP ASVS verification categories are remarkably more problematic. The Clarified Security OÜ team was provided with confirmed information on which vulnerability areas are dominant in Estonian web applications and can use this knowledge to further develop their web application security trainings to give these subjects a bigger emphasis. Another reconfirmed necessity is to bring the relevant awareness to the various management levels, not just to the developers. The results of this thesis should help to convince organizations, that develop and tender web applications, to educate their employees at various levels to acknowledge that these types of vulnerabilities are still very common and should be properly addressed from the earlier phases of development process.

Every day new web applications are brought on-line, at one point maybe also security tested and similar vulnerabilities are detected over and over again. To educate the current and future IT specialists all these areas should be emphasized during various subjects in school and through additional training when needed. Only by improving the common knowledge of these areas at various software development levels can we improve the state of Estonian web security.

References

[1] Cisco; Connections Counter: The Internet of Everything in Motion [www] http://newsroom.cisco.com/feature-content?type=webcontent&articleId=1208342 (21.05.2014)

[2] Evans, Dave; The Internet of Things - How the Next Evolution of the Internet Is Changing Everything . Cisco Internet Business Solutions Group, April 2011, 11 pages. [www] http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf (21.05.2014)

[3] World Internet Stats: Usage and Population Statistics. [www] http://www.internetworldstats.com/stats.htm (21.05.2014)

[4]Nations, Daniel; Web Applications. [www] http://webtrends.about.com/od/webapplications/a/web_application.htm (15.04.2014)

[5] WhiteHat Security; Website Security Statistics Report. May 2013; 53 pages. [www] https://www.whitehatsec.com/assets/WPstatsReport_052013.pdf (04.05.2014)

[6] Abela, Robert; 70% of websites at immediate risk of being hacked! February 2007. [www] http://www.acunetix.com/blog/news/70-percent-websites-immediate-risk-hacked/ (08.05.2014)

[7] Verizon; 2014 Data Breach Investigations Report (DBIR). April 2014; 60 pages. [www] http://www.verizonenterprise.com/DBIR/2014/ (05.05.2014)

[8] OWASP; About. [www] https://www.owasp.org/index.php/About_OWASP (30.03.2014)

[9] OWASP Application Security Verification Standard Project. [www] https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project (01.04.2014)

[10] OWASP Application Security Verification Standard 2013 – Web Application Standard 2.0 (Beta). 2013; 44 pages. [www] http://sourceforge.net/projects/owasp/files/ASVS/OWASP%20ASVS%202013%20Beta%20_v1.0.pdf/download (15.04.2014)

[11] OWASP Application Security Verification Standard 2009 – Web Application Standard 1.0 (Final release). 2009; 42 pages. [www] http://www.owasp.org/images/4/4e/OWASP_ASVS_2009_Web_App_Std_Release.pdf (15.04.2014)

[12] Mashable; Web Apps. [www] http://mashable.com/category/web-apps/ (20.04.2014)

[13] Acunetix; Web Applications: What are They? What of Them? [www] http://www.acunetix.com/websitesecurity/web-applications/ (22.04.2014)

[14] Naor, Ido; Introduction to Security Testing. October 2013. [www] http://university.utest.com/introduction-to-security-testing/ (15.05.2014)

[15] Drew, Steven; Vulnerability Assessments Versus Penetration Tests. March 2006. [www] http://www.secureworks.com/resources/newsletter/2006-03/ (30.03.2014)

[16] SANS Institute Analyst Program; Penetration Testing: Assessing Your Overall Security Before Attackers Do. June 2006, 17 pages. [www] https://www.sans.org/reading-room/analysts-program/PenetrationTesting-June06 (25.04.2014)

[17] Engebretson, Patrick; The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy. Oxford, August 2011; 180 pages.

[18] Rouse, Margaret; Penetration testing; May 2011 [www] http://searchsoftwarequality.techtarget.com/definition/penetration-testing (25.03.2014)

[19] Petukhov, Andrey; Detecting Security Vulnerabilities in Web Applications Using Dynamic Analysis with Penetration Testing. Belgium 2008, 16 pages. [www] https://www.owasp.org/images/3/3e/OWASP-AppSecEU08-Petukhov.pdf

[20] Burris, Jeremy; The great debate: automated vs. manual penetration testing. December 2013, 2 pages. [www] http://onlinedigeditions.com/publication/?i=183413&p=8 (22.04.2014)

[21] OWASP Foundation; OWASP Testing Guide 2007 v2.0. 2007, 344 pages. (30.03.2014)

[22] OWASP Project Inventory. [www] https://www.owasp.org/index.php/Category:OWASP_Project (01.04.2014)

[23] Miessler, Daniel; An Introduction to the OWASP ASVS. January 2013. [www] http://h30499.www3.hp.com/t5/Fortify-Application-Security/An-Introduction-to-the-OWASP-ASVS/ba-p/5932421#.U2JVIV7V3B0 (01.05.2014)

[24] OWASP ASVS Project presentation in English [www] http://www.owasp.org/images/7/71/About_OWASP_ASVS.ppt (04.04.2014)

[25] OWASP Top Ten Project. [www] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[26] OWASP Top 10 – 2010; The Ten Most Critical Web Application Security Risks (Final release). November 2009; 22 pages. [www] http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf

[27] OWASP Top 10 – 2013; The Ten Most Critical Web Application Security Risks (Final release). February 2013; 22 pages.
[www] http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf

[28] OWASP Top Ten 2013 Project Methodology. [www]
https://www.owasp.org/index.php/Top_10_2013/ProjectMethodology

[29] Wichers, Dave; OWASP Top Ten 2013 – Changes from OWASP Top Ten 2010. [www]
http://owasptop10.googlecode.com/files/OWASP_Top-10_2013%20-%20Changes-from-2010.pptx

[30] Hakkaja, Mehis; Clarified Security OÜ founder and CEO. Authors inquiry. (April 2014)

[31] Clarified Security OÜ internal materials. Tallinn, Estonia.

[32] Remenyi, Dan, Williams, Brian, Money, Arthur and Swartz, Ethne; Doing Research in Business and Management: An Introduction to Process and Method. Sage Publications, London, 1998; 320 pages.

[33] Trustwave; 2013 Global Security Report. [www]
http://www2.trustwave.com/rs/trustwave/images/2013-Global-Security-Report.pdf (13.05.2014)

[34] Tudor, Jan; Context Information Security White Paper; Web Application Vulnerability Statistics 2013. June 2013, 37 pages. [www]
http://www.contextis.com/files/Web_Application_Vulnerability_Statistics_-_June_2013.pdf (02.01.2014)

[35] Cenzic; Application Vulnerability Trends Report 2014. 2013, 11 pages. [www]
http://www.cenzic.com/downloads/Cenzic_Vulnerability_Report_2014.pdf (05.05.2014)

[36] IVIz Security; Web Application Vulnerability Statistics of 2012. February 2013, 10 pages. [www] http://www.ivizsecurity.com/Web-Application-Vulnerability-Statistics-of-2012.html (01.05.2014)

[37] Veracode; State of Software Security Report. December 2011, 58 pages.[www] http://info.veracode.com/rs/veracode/images/VERACODE-SOSS-V4.PDF (03.05.2014)

[38] Authors inquiries for creating a simple risk rating. (May 2014)

- Allingu, Martti; Head of Internal Audit and IT Security Department at Centre of Registers and Information Systems
- Hallas, Tiit; Head of IT Security at IT and Development Centre at the Estonian Ministry of the Interior (SMIT)
- Koger, Aivo; Information Security Officer at Elion Enterprises Ltd
- Kulman, Kaur; Head Specialist of the Security Research and Development Department at Estonian Information Systems Authority
- Kääp, Jaanus; Information Security Expert at Clarified Security
- Mägi, Klaid; Head of Information Security at IT Centre of the Estonian Ministry of Finance
- Peekma, Mait; Information Security Expert at Clarified Security
- Rattas, Ragnar; Head of CIIP Section at Estonian Information Systems Authority
- Tallinn, Liisa; Risk Manager at Estonian Information Systems Authority

Appendices

Appendix 1 OWASP Top Ten Application Security Risks – 2013

Description of OWASP Top Ten 2013 Application Security Risks [27].

Risk name	Risk description
A1 – Injection	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2 – Broken Authentication and Session Management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
A3 – Cross-Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4 – Insecure Direct Object References	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5 – Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
A6 – Sensitive Data Exposure	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

Risk name	Risk description
A7 – Missing Function Level Access Control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
A8 - Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A9 - Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
A10 – Unvalidated Redirects and Forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

Appendix 2 OWASP ASVS verification levels

Compiled by Elar Lang of Clarified Security OÜ according to OWASP Application Security Verification Standard 2009 [11].

	Verification description	2A	2B	2A/2B	3
V1	**Security Architecture Documentation Requirements**				
V1.1	Verify that all application components (either individual or groups of source files, libraries, and/or executables) that are present in the application are identified.	x	x	x	x
V1.2	Verify that all components that are not part of the application but that the application relies on to operate are identified.	x	x	x	x
V1.3	Verify that a high-level architecture for the application has been defined.	x	x	x	x
V1.4	Verify that all application components are defined in terms of the business functions and/or security functions they provide.				x

	Verification description	2A	2B	2A/2B	3
V1.5	Verify that all components that are not part of the application but that the application relies on to operate are defined in terms of the business functions and/or security functions they provide.				x
V1.6	Verify that threat modeling information has been provided.				x
V2	**Authentication Verification Requests**				
V2.1	Verify that all pages and resources require authentication except those specifically intended to be public.	x	x	x	x
V2.2	Verify that all password fields do not echo the user's password when it is entered, and that password fields (or the forms that contain them) have autocomplete disabled.	x	x	x	x
V2.3	Verify that if a maximum number of authentication attempts is exceeded, the account is locked for a period of time long enough to deter brute force attacks.	x	x	x	x
V2.4	Verify that all authentication controls are enforced on the server side.	x	x	x	x
V2.5	Verify that all authentication controls (including libraries that call external authentication services) have a centralized implementation.		x	x	x
V2.6	Verify that all authentication controls fail securely.	x	x	x	x
V2.7	Verify that the strength of any authentication credentials are sufficient to withstand attacks that are typical of the threats in the deployed environment.	x	x	x	x
V2.8	Verify that all account management functions are at least as resistant to attack as the primary authentication mechanism.	x	x	x	x
V2.9	Verify that users can safely change their credentials using a mechanism that is at least as resistant to attack as the primary authentication mechanism.	x	x	x	x
V2.10	Verify that re-authentication is required before any application-specific sensitive operations are permitted.	x	x	x	x
V2.11	Verify that after an administratively-configurable period of time, authentication credentials expire.	x	x	x	x
V2.12	Verify that all authentication decisions are logged.		x	x	x
V2.13	Verify that account passwords are salted using a salt that is unique to that account (e.g., internal user ID, account creation) and hashed before storing.		x	x	x
V2.14	Verify that all authentication credentials for accessing services external to the application are encrypted and stored in a protected location (not in source code).		x	x	x
V2.15	Verify that all code implementing or using authentication controls is not affected by any malicious code.				
V3	**Session Management Verification Requirements**				
V3.1	Verify that the framework's default session management control implementation is used by the application.	x	x	x	x
V3.2	Verify that sessions are invalidated when the user logs out.	x	x	x	x
V3.3	Verify that sessions timeout after a specified period of inactivity.	x	x	x	x
V3.4	Verify that sessions timeout after an administratively-configurable maximum time period regardless of activity (an absolute timeout).				x
V3.5	Verify that all pages that require authentication to access them have logout links.	x	x	x	x

	Verification description	2A	2B	2A/2B	3
V3.6	Verify that the session id is never disclosed other than in cookie headers; particularly in URLs, error messages, or logs. This includes verifying that the application does not support URL rewriting of session cookies.		x	x	x
V3.7	Verify that the session id is changed on login.	x	x	x	x
V3.8	Verify that the session id is changed on reauthentication.	x	x	x	x
V3.9	Verify that the session id is changed or cleared on logout.	x	x	x	x
V3.10	Verify that only session ids generated by the application framework are recognized as valid by the application.	x			x
V3.11	Verify that authenticated session tokens are sufficiently long and random to withstand attacks that are typical of the threats in the deployed environment.				x
V3.12	Verify that cookies which contain authenticated session tokens/ids have their domain and path set to an appropriately restrictive value for that site.				x
V3.13	Verify that all code implementing or using session management controls is not affected by any malicious code.				
V4	**Access Control Verification Requirements**				
V4.1	Verify that users can only access protected functions for which they possess specific authorization.	x	x	x	x
V4.2	Verify that users can only access URLs for which they possess specific authorization.	x	x	x	x
V4.3	Verify that users can only access data files for which they possess specific authorization.	x	x	x	x
V4.4	Verify that direct object references are protected, such that only authorized objects are accessible to each user.	x	x	x	x
V4.5	Verify that directory browsing is disabled unless deliberately desired.	x		x	x
V4.6	Verify that users can only access services for which they possess specific authorization.	x	x	x	x
V4.7	Verify that users can only access data for which they possess specific authorization.	x	x	x	x
V4.8	Verify that access controls fail securely.	x	x	x	x
V4.9	Verify that the same access control rules implied by the presentation layer are enforced on the server side.	x	x	x	x
V4.10	Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.	x	x	x	x
V4.11	Verify that all access controls are enforced on the server side.	x	x	x	x
V4.12	Verify that there is a centralized mechanism (including libraries that call external authorization services) for protecting access to each type of protected resource.		x	x	x
V4.13	Verify that limitations on input and access imposed by the business on the application (such as daily transaction limits or sequencing of tasks) cannot be bypassed.	x	x	x	x
V4.14	Verify that all access control decisions can be logged and all failed decisions are logged.		x	x	x
V4.15	Verify that all code implementing or using access controls is not affected by any malicious code.				
V5	**Input Validation Verification Requirements**				

	Verification description	2A	2B	2A/2B	3
V5.1	Verify that the runtime environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows.	x	x	x	x
V5.2	Verify that a positive validation pattern is defined and applied to all input.	x	x	x	x
V5.3	Verify that all input validation failures result in input rejection or input sanitization.	x	x	x	x
V5.4	Verify that a character set, such as UTF-8, is specified for all sources of input.	x	x	x	x
V5.5	Verify that all input validation is performed on the server side.	x	x	x	x
V5.6	Verify that a single input validation control is used by the application for each type of data that is accepted.		x	x	x
V5.7	Verify that all input validation failures are logged.		x	x	x
V5.8	Verify that all input data is canonicalized for all downstream decoders or interpreters prior to validation.				x
V5.9	Verify that all input validation controls are not affected by any malicious code.				
V6	**Output Encoding/Escaping Verification Requirements**				
V6.1	Verify that all untrusted data that are output to HTML (including HTML elements, HTML attributes, javascript data values, CSS blocks, and URI attributes) are properly escaped for the applicable context.	x	x	x	x
V6.2	Verify that all output encoding/escaping controls are implemented on the server side.	x	x	x	x
V6.3	Verify that output encoding /escaping controls encode all characters not known to be safe for the intended interpreter.		x	x	x
V6.4	Verify that all untrusted data that is output to SQL interpreters use parameterized interfaces, prepared statements, or are escaped properly.		x	x	x
V6.5	Verify that all untrusted data that are output to XML use parameterized interfaces or are escaped properly.		x	x	x
V6.6	Verify that all untrusted data that are used in LDAP queries are escaped properly.		x	x	x
V6.7	Verify that all untrusted data that are included in operating system command parameters are escaped properly.		x	x	x
V6.8	Verify that all untrusted data that are output to any interpreters not specifically listed above are escaped properly.		x	x	x
V6.9	Verify that for each type of output encoding/escaping performed by the application, there is a single security control for that type of output for the intended destination.				x
V6.10	Verify that all code implementing or using output validation controls is not affected by any malicious code.				
V7	**Cryptography Verification Requirements**				
V7.1	Verify that all cryptographic functions used to protect secrets from the application user are implemented server side.	x	x	x	x
V7.2	Verify that all cryptographic modules fail securely.	x	x	x	x
V7.3	Verify that access to any master secret(s) is protected from unauthorized access (A master secret is an application credential stored as plaintext on disk that is used to protect access to security configuration information).		x	x	x
V7.4	Verify that password hashes are salted when they are created.		x	x	x

	Verification description	2A	2B	2A/2B	3
V7.5	Verify that cryptographic module failures are logged.		x	x	x
V7.6	Verify that all random numbers, random file names, random GUIDs, and random strings are generated using the cryptographic module's approved random number generator when these random values are intended to be unguessable by an attacker.		x	x	x
V7.7	Verify that cryptographic modules used by the application have been validated against FIPS 140-2 or an equivalent standard. (See http://csrc.nist.gov/groups/STM/cmvp/validation.html).				x
V7.8	Verify that cryptographic modules operate in their approved mode according to their published security policies (See http://csrc.nist.gov/groups/STM/cmvp/validation.html).				x
V7.9	Verify that there is an explicit policy for how cryptographic keys are managed (e.g., generated, distributed, revoked, expired). Verify that this policy is properly enforced.				x
V7.10	Verify that all code supporting or using a cryptographic module is not affected by any malicious code.				
V8	**Error Handling and Logging Verification Requirements**				
V8.1	Verify that that the application does not output error messages or stack traces containing sensitive data that could assist an attacker, including session id and personal information.	x	x	x	x
V8.2	Verify that all server side errors are handled on the server.	x	x	x	x
V8.3	Verify that all logging controls are implemented on the server.	x	x	x	x
V8.4	Verify that error handling logic in security controls denies access by default.	x	x	x	x
V8.5	Verify security logging controls provide the ability to log both success and failure events that are identified as security-relevant.		x	x	x
V8.6	Verify that each log event includes: a time stamp from a reliable source, severity level of the event, an indication that this is a security relevant event (if mixed with other logs), the identity of the user that caused the event (if there is a user associated with the event), the source IP address of the request associated with the event, whether the event succeeded or failed, and a description of the event.		x	x	x
V8.7	Verify that all events that include untrusted data will not execute as code in the intended log viewing software.		x	x	x
V8.8	Verify that security logs are protected from unauthorized access and modification.		x	x	x
V8.9	Verify that there is a single logging implementation that is used by the application.		x	x	x
V8.10	Verify that that the application does not log application-specific sensitive data that could assist an attacker, including user's session ids and personal or sensitive information.		x	x	x
V8.11	Verify that a log analysis tool is available which allows the analyst to search for log events based on combinations of search criteria across all fields in the log record format supported by this system.		x	x	x
V8.12	Verify that all code implementing or using error handling and logging controls is not affected by any malicious code.				
V9	**Data Protection Verification Requirements**				
V9.1	Verify that all forms containing sensitive information have disabled client side caching, including autocomplete features.	x	x	x	x

	Verification description	2A	2B	2A/2B	3
V9.2	Verify that the list of sensitive data processed by this application is identified, and that there is an explicit policy for how access to this data must be controlled, and when this data must be encrypted (both at rest and in transit). Verify that this policy is properly enforced.		x	x	x
V9.3	Verify that all sensitive data is sent to the server in the HTTP message body (i.e., URL parameters are never used to send sensitive data).	x		x	x
V9.4	Verify that all cached or temporary copies of sensitive data sent to the client are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data (e.g., the proper no-cache and no-store Cache-Control headers are set).		x	x	x
V9.5	Verify that all cached or temporary copies of sensitive data stored on the server are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.		x	x	x
V9.6	Verify that there is a method to remove each type of sensitive data from the application at the end of its required retention period.				x
V10	**Communication Security Verification Requirements**				
V10.1	Verify that a path can be built from a trusted CA to each Transport Layer Security (TLS) server certificate, and that each server certificate is valid.	x	x	x	x
V10.2	Verify that failed TLS connections do not fall back to an insecure connection.	x		x	x
V10.3	Verify that TLS is used for all connections (including both external and backend connections) that are authenticated or that involve sensitive data or functions.		x	x	x
V10.4	Verify that backend TLS connection failures are logged.		x	x	x
V10.5	Verify that certificate paths are built and verified for all client certificates using configured trust anchors and revocation information.		x	x	x
V10.6	Verify that all connections to external systems that involve sensitive information or functions are authenticated.		x	x	x
V10.7	Verify that all connections to external systems that involve sensitive information or functions use an account that has been set up to have the minimum privileges necessary for the application to function properly.		x	x	x
V10.8	Verify that there is a single standard TLS implementation that is used by the application that is configured to operate in an approved mode of operation (See http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf).				x
V10.9	Verify that specific character encodings are defined for all connections (e.g., UTF-8).				x
V11	**HTTP Security Verification Requirements**				
V11.1	Verify that redirects do not include unvalidated data.	x	x	x	x
V11.2	Verify that the application accepts only a defined set of HTTP request methods, such as GET and POST.	x	x	x	x
V11.3	Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8).	x	x	x	x

	Verification description	2A	2B	2A/2B	3
V11.4	Verify that the HTTPOnly flag is used on all cookies that do not specifically require access from JavaScript.	x	x	x	x
V11.5	Verify that the secure flag is used on all cookies that contain sensitive data, including the session cookie.	x	x	x	x
V11.6	Verify that HTTP headers in both requests and responses contain only printable ASCII characters.	x	x	x	x
V11.7	Verify that the application generates a strong random token as part of all links and forms associated with transactions or accessing sensitive data, and that the application verifies the presence of this token with the proper value for the current user when processing these requests.				x
V12	**Security Configuration Verification Requirements**				
V12.1	Verify that all security-relevant configuration information is stored in locations that are protected from unauthorized access.		x	x	x
V12.2	Verify that all access to the application is denied if the application cannot access its security configuration information.		x	x	x
V12.3	Verify that all changes to the security configuration settings managed by the application are logged in the security event log.				x
V12.4	Verify that the configuration store can be output in a human-readable format to facilitate audit.				
V13	**Malicious Code Search Verification Requirements**				
V13.1	Verify that no malicious code is in any code that was either developed or modified in order to create the application.				
V13.2	Verify that the integrity of interpreted code, libraries, executables, and configuration files is verified using checksums or hashes.				
V14	**Internal Security Verification Requirements**				
V14.1	Verify that the application protects user and data attributes and policy information used by access controls from unauthorized access or modification.				x
V14.2	Verify that security control interfaces are simple enough to use that developers are likely to use them correctly.				
V14.3	Verify that the application properly protects shared variables and resources from inappropriate concurrent access.				

Appendix 3 Individual findings according to OWASP ASVS verification categories

Verification Requirement		Total findings	Critical	High	Medium	Low
V1	**Security Architecture Documentation Requirements**					
V1.1	Verify that all application components (either individual or groups of source files, libraries, and/or executables) that are present in the application are identified.	**3**				3
V1.2	Verify that all components that are not part of the application but that the application relies on to operate are identified.	**5**		1		4
V1.3	Verify that a high-level architecture for the application has been defined.					

Verification Requirement		Total findings	Critical	High	Medium	Low
V1.4	Verify that all application components are defined in terms of the business functions and/or security functions they provide.					
V1.5	Verify that all components that are not part of the application but that the application relies on to operate are defined in terms of the business functions and/or security functions they provide.					
V1.6	Verify that threat modeling information has been provided.	**1**				1
V2	**Authentication Verification Requests**					
V2.1	Verify that all pages and resources require authentication except those specifically intended to be public.	**6**	1	4		1
V2.2	Verify that all password fields do not echo the user's password when it is entered, and that password fields (or the forms that contain them) have autocomplete disabled.	**8**			2	6
V2.3	Verify that if a maximum number of authentication attempts is exceeded, the account is locked for a period of time long enough to deter brute force attacks.	**17**		3	8	6
V2.4	Verify that all authentication controls are enforced on the server side.	**2**		1		1
V2.5	Verify that all authentication controls (including libraries that call external authentication services) have a centralized implementation.					
V2.6	Verify that all authentication controls fail securely.					
V2.7	Verify that the strength of any authentication credentials are sufficient to withstand attacks that are typical of the threats in the deployed environment.	**6**	1	1	1	3
V2.8	Verify that all account management functions are at least as resistant to attack as the primary authentication mechanism.					
V2.9	Verify that users can safely change their credentials using a mechanism that is at least as resistant to attack as the primary authentication mechanism.	**2**				2
V2.10	Verify that re-authentication is required before any application-specific sensitive operations are permitted.	**2**		1		1
V2.11	Verify that after an administratively-configurable period of time, authentication credentials expire.					
V2.12	Verify that all authentication decisions are logged.	**5**			3	2
V2.13	Verify that account passwords are salted using a salt that is unique to that account (e.g., internal user ID, account creation) and hashed before storing.	**3**		1		2
V2.14	Verify that all authentication credentials for accessing services external to the application are encrypted and stored in a protected location (not in source code).	**1**				1
V2.15	Verify that all code implementing or using authentication controls is not affected by any malicious code.					
V3	**Session Management Verification Requirements**					

Verification Requirement		Total findings	Critical	High	Medium	Low
V3.1	Verify that the framework's default session management control implementation is used by the application.					
V3.2	Verify that sessions are invalidated when the user logs out.	**3**			3	
V3.3	Verify that sessions timeout after a specified period of inactivity.	**7**			1	6
V3.4	Verify that sessions timeout after an administratively-configurable maximum time period regardless of activity (an absolute timeout).	**2**				2
V3.5	Verify that all pages that require authentication to access them have logout links.	**8**		1	1	6
V3.6	Verify that the session id is never disclosed other than in cookie headers; particularly in URLs, error messages, or logs. This includes verifying that the application does not support URL rewriting of session cookies.	**8**			2	6
V3.7	Verify that the session id is changed on login.	**17**		3	6	8
V3.8	Verify that the session id is changed on reauthentication.	**3**		3		
V3.9	Verify that the session id is changed or cleared on logout.	**8**			3	5
V3.10	Verify that only session ids generated by the application framework are recognized as valid by the application.	**8**		2	2	4
V3.11	Verify that authenticated session tokens are sufficiently long and random to withstand attacks that are typical of the threats in the deployed environment.					
V3.12	Verify that cookies which contain authenticated session tokens/ids have their domain and path set to an appropriately restrictive value for that site.	**3**				3
V3.13	Verify that all code implementing or using session management controls is not affected by any malicious code.					
V4	**Access Control Verification Requirements**					
V4.1	Verify that users can only access protected functions for which they possess specific authorization.	**15**	5	5	3	2
V4.2	Verify that users can only access URLs for which they possess specific authorization.	**3**		1		2
V4.3	Verify that users can only access data files for which they possess specific authorization.	**8**	2	4	1	1
V4.4	Verify that direct object references are protected, such that only authorized objects are accessible to each user.	**6**	1	3	1	1
V4.5	Verify that directory browsing is disabled unless deliberately desired.	**5**		1	1	3
V4.6	Verify that users can only access services for which they possess specific authorization.	**5**	1	1	2	1
V4.7	Verify that users can only access data for which they possess specific authorization.	**3**		1		2
V4.8	Verify that access controls fail securely.	**1**				1
V4.9	Verify that the same access control rules implied by the presentation layer are enforced on the server side.	**1**			1	

Verification Requirement		Total findings	Critical	High	Medium	Low
V4.10	Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.					
V4.11	Verify that all access controls are enforced on the server side.					
V4.12	Verify that there is a centralized mechanism (including libraries that call external authorization services) for protecting access to each type of protected resource.					
V4.13	Verify that limitations on input and access imposed by the business on the application (such as daily transaction limits or sequencing of tasks) cannot be bypassed.					
V4.14	Verify that all access control decisions can be logged and all failed decisions are logged.	**1**				1
V4.15	Verify that all code implementing or using access controls is not affected by any malicious code.					
V5	**Input Validation Verification Requirements**					
V5.1	Verify that the runtime environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows.	**1**				1
V5.2	Verify that a positive validation pattern is defined and applied to all input.	**13**		2	7	4
V5.3	Verify that all input validation failures result in input rejection or input sanitization.	**14**		6	2	6
V5.4	Verify that a character set, such as UTF-8, is specified for all sources of input.	**2**				2
V5.5	Verify that all input validation is performed on the server side.	**7**	1	3		3
V5.6	Verify that a single input validation control is used by the application for each type of data that is accepted.					
V5.7	Verify that all input validation failures are logged.	**2**				2
V5.8	Verify that all input data is canonicalized for all downstream decoders or interpreters prior to validation.					
V5.9	Verify that all input validation controls are not affected by any malicious code.					
V6	**Output Encoding/Escaping Verification Requirements**					
V6.1	Verify that all untrusted data that are output to HTML (including HTML elements, HTML attributes, javascript data values, CSS blocks, and URI attributes) are properly escaped for the applicable context.	**35**		22	10	3
V6.2	Verify that all output encoding/escaping controls are implemented on the server side.					
V6.3	Verify that output encoding /escaping controls encode all characters not known to be safe for the intended interpreter.	**1**		1		
V6.4	Verify that all untrusted data that is output to SQL interpreters use parameterized interfaces, prepared statements, or are escaped properly.	**9**	2	3	1	3
V6.5	Verify that all untrusted data that are output to XML use parameterized interfaces or are escaped properly.	**2**			2	

Verification Requirement		Total findings	Critical	High	Medium	Low
V6.6	Verify that all untrusted data that are used in LDAP queries are escaped properly.	**1**			1	
V6.7	Verify that all untrusted data that are included in operating system command parameters are escaped properly.					
V6.8	Verify that all untrusted data that are output to any interpreters not specifically listed above are escaped properly.	**1**			1	
V6.9	Verify that for each type of output encoding/escaping performed by the application, there is a single security control for that type of output for the intended destination.					
V6.10	Verify that all code implementing or using output validation controls is not affected by any malicious code.					
V7	**Cryptography Verification Requirements**					
V7.1	Verify that all cryptographic functions used to protect secrets from the application user are implemented server side.					
V7.2	Verify that all cryptographic modules fail securely.					
V7.3	Verify that access to any master secret(s) is protected from unauthorized access (A master secret is an application credential stored as plaintext on disk that is used to protect access to security configuration information).					
V7.4	Verify that password hashes are salted when they are created.	**2**				2
V7.5	Verify that cryptographic module failures are logged.					
V7.6	Verify that all random numbers, random file names, random GUIDs, and random strings are generated using the cryptographic module's approved random number generator when these random values are intended to be unguessable by an attacker.	**2**				2
V7.7	Verify that cryptographic modules used by the application have been validated against FIPS 140-2 or an equivalent standard. (See http://csrc.nist.gov/groups/STM/cmvp/validation.html).					
V7.8	Verify that cryptographic modules operate in their approved mode according to their published security policies (See http://csrc.nist.gov/groups/STM/cmvp/validation.html).	**1**				1
V7.9	Verify that there is an explicit policy for how cryptographic keys are managed (e.g., generated, distributed, revoked, expired). Verify that this policy is properly enforced.					
V7.10	Verify that all code supporting or using a cryptographic module is not affected by any malicious code.					
V8	**Error Handling and Logging Verification Requirements**					
V8.1	Verify that that the application does not output error messages or stack traces containing sensitive data that could assist an attacker, including session id and personal information.	**28**	1	2	3	22

Verification Requirement		Total findings	Critical	High	Medium	Low
V8.2	Verify that all server side errors are handled on the server.	**2**			1	1
V8.3	Verify that all logging controls are implemented on the server.	**2**		1	1	
V8.4	Verify that error handling logic in security controls denies access by default.					
V8.5	Verify security logging controls provide the ability to log both success and failure events that are identified as security-relevant.	**2**				2
V8.6	Verify that each log event includes: a time stamp from a reliable source, severity level of the event, an indication that this is a security relevant event (if mixed with other logs), the identity of the user that caused the event (if there is a user associated with the event), the source IP address of the request associated with the event, whether the event succeeded or failed, and a description of the event.	**10**			5	5
V8.7	Verify that all events that include untrusted data will not execute as code in the intended log viewing software.	**1**		1		
V8.8	Verify that security logs are protected from unauthorized access and modification.	**2**			2	
V8.9	Verify that there is a single logging implementation that is used by the application.	**1**				1
V8.10	Verify that that the application does not log application-specific sensitive data that could assist an attacker, including user's session ids and personal or sensitive information.	**1**				1
V8.11	Verify that a log analysis tool is available which allows the analyst to search for log events based on combinations of search criteria across all fields in the log record format supported by this system.					
V8.12	Verify that all code implementing or using error handling and logging controls is not affected by any malicious code.					
V9	**Data Protection Verification Requirements**					
V9.1	Verify that all forms containing sensitive information have disabled client side caching, including autocomplete features.	**10**			2	8
V9.2	Verify that the list of sensitive data processed by this application is identified, and that there is an explicit policy for how access to this data must be controlled, and when this data must be encrypted (both at rest and in transit). Verify that this policy is properly enforced.					
V9.3	Verify that all sensitive data is sent to the server in the HTTP message body (i.e., URL parameters are never used to send sensitive data).	**5**				5
V9.4	Verify that all cached or temporary copies of sensitive data sent to the client are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data (e.g., the proper no-cache and no-store Cache-Control headers are set).	**14**			3	11

Verification Requirement		Total findings	Critical	High	Medium	Low
V9.5	Verify that all cached or temporary copies of sensitive data stored on the server are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.	**2**				2
V9.6	Verify that there is a method to remove each type of sensitive data from the application at the end of its required retention period.					
V10	**Communication Security Verification Requirements**					
V10.1	Verify that a path can be built from a trusted CA to each Transport Layer Security (TLS) server certificate, and that each server certificate is valid.					
V10.2	Verify that failed TLS connections do not fall back to an insecure connection.	**2**			1	1
V10.3	Verify that TLS is used for all connections (including both external and backend connections) that are authenticated or that involve sensitive data or functions.	**8**		5	3	
V10.4	Verify that backend TLS connection failures are logged.					
V10.5	Verify that certificate paths are built and verified for all client certificates using configured trust anchors and revocation information.					
V10.6	Verify that all connections to external systems that involve sensitive information or functions are authenticated.					
V10.7	Verify that all connections to external systems that involve sensitive information or functions use an account that has been set up to have the minimum privileges necessary for the application to function properly.					
V10.8	Verify that there is a single standard TLS implementation that is used by the application that is configured to operate in an approved mode of operation (See http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf).	**5**				5
V10.9	Verify that specific character encodings are defined for all connections (e.g., UTF-8).					
V11	**HTTP Security Verification Requirements**					
V11.1	Verify that redirects do not include unvalidated data.	**5**			3	2
V11.2	Verify that the application accepts only a defined set of HTTP request methods, such as GET and POST.	**6**				6
V11.3	Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8).	**6**		1		5
V11.4	Verify that the HTTPOnly flag is used on all cookies that do not specifically require access from JavaScript.	**14**			2	12
V11.5	Verify that the secure flag is used on all cookies that contain sensitive data, including the session cookie.	**16**			3	13
V11.6	Verify that HTTP headers in both requests and responses contain only printable ASCII characters.					

Verification Requirement		Total findings	Critical	High	Medium	Low
V11.7	Verify that the application generates a strong random token as part of all links and forms associated with transactions or accessing sensitive data, and that the application verifies the presence of this token with the proper value for the current user when processing these requests.	**30**		18	9	3
V12	**Security Configuration Verification Requirements**					
V12.1	Verify that all security-relevant configuration information is stored in locations that are protected from unauthorized access.	**5**			1	4
V12.2	Verify that all access to the application is denied if the application cannot access its security configuration information.					
V12.3	Verify that all changes to the security configuration settings managed by the application are logged in the security event log.					
V12.4	Verify that the configuration store can be output in a human-readable format to facilitate audit.					
V13	**Malicious Code Search Verification Requirements**					
V13.1	Verify that no malicious code is in any code that was either developed or modified in order to create the application.					
V13.2	Verify that the integrity of interpreted code, libraries, executables, and configuration files is verified using checksums or hashes.					
V14	**Internal Security Verification Requirements**					
V14.1	Verify that the application protects user and data attributes and policy information used by access controls from unauthorized access or modification.	**1**				1
V14.2	Verify that security control interfaces are simple enough to use that developers are likely to use them correctly.					
V14.3	Verify that the application properly protects shared variables and resources from inappropriate concurrent access.					

Appendix 4 Severity ratings with most findings of individual verification points

OWASP ASVS verification requirement		total findings	"critical" findings	% of findings
V4.1	Verify that users can only access protected functions for which they possess specific authorization.	15	**5**	33
V4.3	Verify that users can only access data files for which they possess specific authorization.	8	**2**	25
V6.4	Verify that all untrusted data that is output to SQL interpreters use parameterized interfaces, prepared statements, or are escaped properly.	9	**2**	22

OWASP ASVS verification requirement		total findings	"high" findings	% of findings
V6.1	Verify that all untrusted data that are output to HTML (including HTML elements, HTML attributes, javascript data values, CSS blocks, and URI attributes) are properly escaped for the applicable context.	35	**22**	63
V11.7	Verify that the application generates a strong random token as part of all links and forms associated with transactions or accessing sensitive data, and that the application verifies the presence of this token with the proper value for the current user when processing these requests.	30	**18**	60
V5.3	Verify that all input validation failures result in input rejection or input sanitization.	14	**6**	43

OWASP ASVS verification requirement		total findings	"medium" findings	% of findings
V6.1	Verify that all untrusted data that are output to HTML (including HTML elements, HTML attributes, javascript data values, CSS blocks, and URI attributes) are properly escaped for the applicable context.	35	**10**	29
V11.7	Verify that the application generates a strong random token as part of all links and forms associated with transactions or accessing sensitive data, and that the application verifies the presence of this token with the proper value for the current user when processing these requests.	30	**9**	30
V2.3	Verify that if a maximum number of authentication attempts is exceeded, the account is locked for a period of time long enough to deter brute force attacks.	17	**8**	47

OWASP ASVS verification requirement		total findings	"low" findings	% of findings
V8.1	Verify that that the application does not output error messages or stack traces containing sensitive data that could assist an attacker, including session id and personal information.	28	**22**	79
V11.5	Verify that the secure flag is used on all cookies that contain sensitive data, including the session cookie.	16	**13**	81
V11.4	Verify that the HTTPOnly flag is used on all cookies that do not specifically require access from JavaScript.	14	**12**	86

Appendix 5 Severity ratings with most findings according to verification categories

OWASP ASVS verification category		total findings	"critical" findings	% of findings
V4	Access Control Verification Requirements	48	**9**	19
V6	Output Encoding/Escaping Verification Requirements	49	**2**	4
V2	Authentication Verification Requests	52	**2**	4

OWASP ASVS verification category		total findings	"high" findings	% of findings
V6	Output Encoding/Escaping Verification Requirements	49	**26**	53
V11	Security Architecture Documentation Requirements	77	**19**	25
V4	Access Control Verification Requirements	48	**16**	33

OWASP ASVS verification category		total findings	"medium" findings	% of findings
V3	Session Management Verification Requirements	67	**18**	27
V11	Security Architecture Documentation Requirements	77	**17**	22
V6	Output Encoding/Escaping Verification Requirements	49	**15**	31

OWASP ASVS verification category		total findings	"low" findings	% of findings
V11	Security Architecture Documentation Requirements	77	**41**	53
V3	Session Management Verification Requirements	67	**40**	60
V2	Authentication Verification Requests	49	**32**	65

Appendix 6 Findings grouped according to OWASP ASVS verification categories

Verification requirement		Total findings	Critical	High	Medium	Low
V1	Security Architecture Documentation Requirements	9		1		8
V2	Authentication Verification Requests	52	2	11	14	25
V3	Session Management Verification Requirements	67		9	18	40
V4	Access Control Verification Requirements	48	9	16	9	14
V5	Input Validation Verification Requirements	39	1	11	9	18
V6	Output Encoding/Escaping Verification Requirements	49	2	26	15	6
V7	Cryptography Verification Requirements	5				5
V8	Error Handling and Logging Verification Requirements	49	1	4	12	32
V9	Data Protection Verification Requirements	31			5	26
V10	Communication Security Verification Requirements	15		5	4	6
V11	HTTP Security Verification Requirements	77		19	17	41
V12	Security Configuration Verification Requirements	5			1	4
V13	Malicious Code Search Verification Requirements					
V14	Internal Security Verification Requirements	1				1