


## Security audit of website based on WordPress

Sergey Alto

Bachelor's Thesis  
Degree Programme in  
Business Information Technology  
2017



<b>Author</b> Alto Sergey	<b>Year of entry</b> 2015
<b>Degree programme</b> Business Information Technology	
<b>Report/thesis title</b> Security audit of website based on WordPress	<b>Number of pages and appendix pages</b> 53 + 8
<b>Thesis advisor</b> Olavi Korhonen	
<p>This Bachelor's thesis discovers how to check security level of website based on WordPress content management system. Many websites are using WordPress CMS today; it means that security is the one of most important things to consider.</p> <p>The main purpose was to collect information about possible ways of testing security level, how to predict different kind of attacks exist today and improve level of security protection.</p> <p>The thesis consists of 3 parts:</p> <p>Theory section, which consists of theoretical research of WordPress platform, its core components and discovering ways of possible attacks.</p> <p>Empirical part, which includes information about possible attacks and security checks and design of a security testing cases.</p> <p>Third part is implementation of previous part, which demonstrate methods and tools on the real website. Results are collecting and placing into final report that contains analysis of test outcomes and recommendations for security improvement. Work for thesis have done in time from March to May of 2017.</p> <p>The material was collected as follows:</p> <p>Online sources were used to collect ideas and build the plan of security audit.</p> <p>Free and open-source tools have been used for practical implementation.</p> <p>Finally, the conclusion of the thesis indicates achieved outcomes and evaluates project work.</p>	
<b>Keywords</b> WordPress, security, vulnerabilities, threats, audit.	

## Table of contents

1	Glossary.....	1
2	Introduction .....	3
3	Environment.....	4
4	Theoretical framework.....	5
4.1	CMS.....	5
4.2	WordPress.....	6
4.3	WordPress History .....	7
4.4	WordPress structure and installation.....	10
4.4.1	Database .....	11
4.4.2	Set of files and folders with a code.....	12
4.4.3	WordPress extensions .....	13
4.4.4	Dashboard .....	14
4.5	Possible ways to hack.....	14
4.5.1	Getting access to SQL database.....	14
4.5.2	Getting access to WordPress dashboard .....	15
4.5.3	Getting access to core files and folders of WordPress .....	16
4.5.4	Checking WordPress platform, plugins and themes vulnerabilities.....	16
5	Empirical part .....	17
5.1	Strategy .....	17
5.2	Test cases design .....	17
5.2.1	Information gathering .....	18
5.2.2	WordPress folder locations and its accessibility .....	21
5.2.3	Extensions vulnerabilities test .....	22
5.2.4	Brute force user's attack test.....	23
5.2.5	Discovering the scripts of Website. ....	24
5.2.6	SQL Injection and common SQL attack's test .....	26
5.2.7	XSS attack test .....	27
5.2.8	File upload test.....	28

5.3	Preparing to testing.....	28
6	Security audit report 1 .....	29
6.1	Test case 1 implementation. Gathering information.....	29
6.1.1	Running the test 1 .....	29
6.1.2	Analysis of the test 1 results.....	33
6.1.3	Recommendations for security protection improvement .....	37
6.2	Test case 2 implemetation. WP-folders location and its accessibility test.....	38
6.2.1	Running the test 2.....	38
6.2.2	Analysis of the test 2 results.....	39
6.3	Test case 3. Extensions vulnerabilities detection .....	40
6.3.1	Running the test 3.....	40
6.3.2	Analysis of the test 3 results.....	40
6.3.3	Recommendations for security improvement.....	42
6.4	Test case 4 implementation. Brute Force attack.....	43
6.4.1	Running the test 4.....	43
6.4.2	Analysis of the test results 4.....	43
6.5	Test case 5. Discovering the scripts of Website. ....	44
6.5.1	Running the test 5.....	44
6.5.2	Analysis of the test 5 results.....	44
6.5.3	Recommendations for security improvement.....	44
6.6	Test case 6. SQL injection attack.....	45
6.6.1	Running the test 6.....	45
6.6.2	Analysis of the test 6 results.....	45
6.7	Test case 7 implementation. Cross site scripting attack test.....	46
6.7.1	Running the test 7.....	46
6.7.2	Analysis of the test 7 results.....	46
6.8	Test case 8 implementation. File upload test .....	46
6.8.1	Running the test 7.....	46
6.8.2	Analysis of the test 8 results.....	47

6.9	Common test's results and security audit outcomes.....	47
6.10	Testing outcomes.....	48
7	Security audit report 2 .....	49
7.1	Test case 1 implementation. Gathering information.....	49
7.1.1	Running the test 2.1 .....	49
7.1.2	Analysis of the test 2.1 results.....	50
7.1.3	Recommendations for security improvement.....	52
8	Conclusions .....	53
	References .....	54
	Appendix A. Description of database tables of the WordPress.....	57
	Appendix B. Description of core files and folders. ....	61

# 1 Glossary

**Browser, web browser** – software application for retrieving, presenting and traversing information resources on the World Wide Web.

**CMS** – content management system, an application that supports creation and modification of digital content.

**Cross-site scripting (XSS)** - a vulnerability, which enables attackers to inject client-side scripts into web pages to bypass access controls such as the same-origin policy.

**CSS** – cascading style sheets, style sheet language used for describing the presentation of a document written in a markup language.

**Domain Name System (DNS)** - a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network.

**Domain, domain name** – identification string that defines a realm of administrative autonomy authority or control within the Internet. Domain names are formed by the rules and procedures of the Domain Name System (DNS).

**DOM** - Document Object Model, is a programming interface for HTML and XML documents.

**IP address** – an identifier assigned to each computer connected to the network (including the Internet).

**GMT** – Greenwich Mean Time, the international civil time standard used as a basis for all time zones in the earth.

**Hosting** – service allows individuals and organizations to make their website or files accessible via World Wide Web.

**HTML** – hypertext markup language, standard markup language for creating web pages and web applications.

**HTML/CSS** – combination of two web oriented languages are commonly used for web design.

**HTTP** – hypertext transfer protocol, application protocol for exchange and transfer hypertext.

**HTTP request** – message sending to server contains requested address off webpage and set of parameters.

**Hypertext** – text with links (hyperlinks) to other text displayed on a computer display or other electronic devices. The reader can access immediately all links, because they can be activated by mouse, keyboard, sensor or other type of input devices.

**Linux** – operating system assembled under the model of free and open-source software development and distribution.

**OS** – operating system, software manages computer hardware, distribute its resources and provides common services for computer programs.

**PHP** – server-side scripting programming language.

**Ping time** – the length of time it takes for signal to be sent plus the length of time response have been received.

**Post (in WordPress)** – single webpage in WordPress.

**Relational database** – database, which has data organization based on relationships.

The idea of organization is presence of data in one or more tables. Every table has unique key which identifies each of row.

**Same-origin policy** - a concept in the web application security model. permits scripts contained in a first web page to access data in a second web page in the web browser.

**Security** – protection of website or computer system from the theft of damage to their information, software or hardware.

**Sidebar** – Vertical column located on the left side or on the right side of the page for displaying information other than the main content.

**SQL** – structured query language used in programming and designed for data management in relational database.

**Taxonomy** - a grouping mechanism for posts (pages) which helps to categorize them together using various names for better understanding and management by user.

**The Internet** - The Internet is a global system of interconnected computer networks that interchange data by packet switching using the standardized Internet Protocol Suite.

**Trackback time** – a way to notify legacy blog systems that you have linked to them.

**User interface** – a part of software that handles human-machine interaction.

**Vulnerability** - a weakness, which allows an attacker to hack website

**Web application, web app** – client-server software application which client part runs in browser.

**Web server** – system based on computer connected to the Internet, which processes HTTP requests. Main functions of web server are to store, process and deliver web pages to client sent request.

**Whols** – domain name lookup service to search information from database for domain name registration.

**Wordpress** – most popular CMS in the World today.

**World Wide Web, WWW, the Web** - is an information space where documents and other web resources are identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and can be accessed via the Internet.

## 2 Introduction

There are over 1.1 billion websites in the world wide *web today* (Total number of Websites, 2016). Website is the most simplest way to share information about your business, services, products, ideas. So it is very important and useful information tool today. Basically, all websites are using HTML code, but different ways of its creation can be used. Approximately half of websites are using CMS (content management system) today (Usage of content management systems for websites, n.d.). It allows to create new pages, posts, store images, media files and other content using simple and understandable tools. It allows to manage website without writing code (or minimum code required, create unified look and user's permission management. Most popular CMS today is Wordpress, it covers more than 27% websites which are using content management system (Usage of content management systems for websites, n.d.). It means that more than 100 million of websites run on WordPress platform. Half of this amount of websites are self-hosted and many of people who administrate their websites are not seriously concerned about security. Of course, WordPress is updating everytime and improves security from release to release, but, in fact, the problem is deeper than it seems.

The aim of my research is to create detailed effective guide contains set of tests with explanations to do security protection audit of WordPress-based. It should cover checking of all components of the website, vulnerabilities monitoring and, if possible, offering solution for to fix it.

One more objective is practical implementation of my guide have created. I plan to check one website have chosen for tests. It will help to check if all instructions, applications and commands are suitable for chosen purpose and well structured (running in right order), haven't errors. Guide should consist of set of tests that allows to do evaluation of security level and get information for improvement if it possible.

The methods and techniques used in this research are empirical, but it reference to various researches made by different authors to provide main ideas of security audit for website. The structure of the thesis is built to be easy for the reader to understand ideas without deep knowledge of testing technologies and applications.

The results of thesis can be used for security audit WordPress websites and protection improvement from attackers.



### 3 Environment

Environment is based on two computers have similar configuration: desktop (Intel core i5, 4Gb RAM) and laptop (Intel core i3, 4Gb RAM), which will be using for running test cases outside of my laboratory. Both computers are connecting to the Internet using LAN or WI-FI connection. Operating system Kali linux 2016.2 have installed on both machines as the main OS. I will use several applications for testing. Most of it have already preinstalled into the current version of Kali Linux.

Kali Linux is a Debian-based Linux distribution provides tools for penetration testing, ethical hacking and security audit. It contains several hundreded tools which are geared towards various information security tasks, such as Penetration Testing, Security research, Computer Forensics and Reverse Engineering. Kali Linux is developed, funded and maintained by Offensive Security, a leading information security training company (What is Kali Linux?, n.d.).

Firefox browser preinstalled in Kali Linux have been used for information gathering and when developer tools were needed.

I will use also online-based tools for testing. There are Whols service, search engines.

Target website have been used for testing is located on URL: [www.digitaluniversity.fi](http://www.digitaluniversity.fi). This website provides scalable platform for teaching and learning. It contains courses for learning offered by different schools and allows to manage it using one account only registered on this website. There is good target for testing security audit solution, because it contains registration and account management with different roles of users, extensive system of the pages (posts). Further, it is possible to use this platform for different courses – free of charge and payable. Payments need big attention to security. Finally, many of users will store a lot of personal information in the website database like names, e-mails, date of birth etc. Administration of website is responsible for keeping information hidden from others.

Finally, I want to notice, that chosen environment consists of components are accessible for everyone, it allows to test (or attack) website without expensive or unavailable equipment or software.

## 4 Theoretical framework

The nature of WordPress website is based on content management system which allows to simplify creation and management of its content. It allows to standardise structure of website using templates for different kinds of documents, manage multimedia content, users etc.

### 4.1 CMS

CMS – content management system is a computer application that supports the creation and modification of digital content (Content management system, 2017). It runs on the web-server where its components are installed. Most CMSs are using for Web-based content and named also web content management systems (WCMSs). The main purpose of WCMSs is to support management of the content of Web pages (Content management system, 2017). WCMS process HTTP requests into webpages can be accessed through web-browser. The main feature of CMS is ability to design websites easier and more effective, sometimes, even without special knowledge of computer language. It helps to create and manage webpages using understandable interface.

CMS consists of several components connected between each other. There are database, different kinds of content (texts, graphics, audio and video etc), application components are using for content management, content delivery and extension's management. Database is storing user accounts including their usernames, roles, passwords. It allows to implement user management tools into application accessible through web browser. It opens many features from multi-user administration of website to using user's authentication on the website pages. Database allows also technical features for data indexing and implementation of internal search engines for easy access to information and search webpages by keywords and attributes. Version control, a large number of extensions, sets of templates and themes, SEO-friendly URLs – this is not full list of features CMS offers.

CMSs can be divided into three types by the way of page generation:

**Page generation by query** – this type uses schema "edit module – database – presentation module". Presentation (generation) of webpage is based on current information taken from database. Page is generated by server when query have received. It makes this type of CMS flexible from the point of dynamic content changing, but it creates additional load on the server resources.

**Page generation after editing (creation)** – this type of CMS generates static webpages immediately after its editing or creation. It helps to improve server resources optimisation, but impair interaction between user and website content.

**Mixed type of page generation** - type of CMS combines previous two ideas. Realisation can be achieved using several ways – caching, saving blocks from pages and construction page from blocks after query etc.

Approximately half of total number of websites are using content management system. Most popular CMSs today are WordPress, Joomla, Drupal, Magento, Blogger, Shopify, TYPO3, Bitrix (Usage of content management systems for websites, n.d.).

But WordPress is the most commonly used CMS today (more than 50% of websites are using CMS), it means that 25% of all websites in the world powered by WordPress. (How to choose the right CMS for your Website, 2015). Approximately 409+ million people view more than 19.6 billion WordPress pages monthly (Kimi, 2016).

## **4.2 WordPress**

WordPress is definitely most popular CMS today. It powers more than 50% of websites are using CMS and approximately 25% of total number of websites in the World.

WordPress is not only the most popular CMS, it is also the fastest-growing system: Every 74 seconds a site within the top 10 million starts using WordPress. Compare this with Shopify, the second-fastest growing CMS, which is gaining a new site every 22 minutes. (WordPress Used On 25 Percent Of All Websites, 2015).

There are many reasons for success:

First of all, WordPress is Open Source and Free CMS based on PHP and SQL, PHP is used by 82,6% of all websites whose server-side programming language is known (Usage of server-side programming languages for websites, n.d.), and SQL is the most popular language used in computers to create and manage databases (Introduction to SQL, 2007).

Besides, WordPress doesn't require computer languages knowledge (like PHP, SQL and HTML required for many other CMSs). Moreover, a large number of plugins and themes available, most of them are free and can be accessible from control panel or official WordPress website.

Finally, this platform has the largest community of users, designers, plugin creators, webmasters, website developers and so on. It allows to discuss actual problems, get support, create new extensions, and everyone can access it.

Despite the large number of advantages, WordPress has several disadvantages:

First of all, a lot of modifications need to change the code, which require PHP or HTML/CSS knowledge. It can limit some possibilities for web design implementation on this platform.

Second of all plugins, themes and widgets for WordPress are designing by third-party developers, which makes usage of resources non-effective and can cause conflicts between extensions or decrease script's efficiency.

Third of all, WordPress has not well secured platform as other CMSs. Many of security issues have been uncovered in the software, but a list of possible vulnerabilities is rising over time. Generally, WordPress engine security improves from version to version by efforts of Development team and its community including volunteers who helps to test new releases. Despite good support from developers and volunteers, WordPress has security problems, especially caused by usage of plug-ins and other extensions developed by third-part developers.

### **4.3 WordPress History**

**2001-2003.** The history of WordPress started in 2001 from developing a tool named "cafelog" which purpose was to simplify creation and editing of webpages for news or blogs posting. In opinion of the author of "Cafelog" (Project page, 2001) main features were that pages were generated dynamically from the MySQL database, it allowed to use more search/display capabilities, and the ability to serve news/blogs in different 'templates'. Platform used MySQL database and PHP code runned on web server.

**2003.** At 2003 two users of "b2/cafelog" Matt Mullenweg and Mike Little, decided to build a new platform based on "cafelog", because project was discontinued. On April 1st, 2003, Matt created new branch of b2 on SourceForge, and, with the name coined by his friend Christine Tremoulet, called it WordPress (McKeown, 2017). On 27 May 2003, Matt announced the availability of the first version of WordPress (WP beginner 2017). It was based on b2 Cafelog with significant improvements. One of the most important things was to make platform simple and possible to use it without PHP and SQL skills. At the same time website wordpress.org have launched, it contained support, forum, starting documentation and development blog. It was useful for new users to create new websites and get involved. At the same year many bugs and security issues were fixed.

**2004.** Plugin Architecture released in version 1.2 allowed to extend platform functionality by writing plugins and sharing in community. WordPress have become more popular and, for example, the number of users increased from 8,000 in April to 19,000 in May

(WordPress News, 2013). Running of plugin functionality have caused first serious security issues, because first versions of it were not completely tested and proved.

**2005.** Theme system and static pages were introduced. A new backend user interface, persistent caching functionality and new user roles have been introduced during this year. In addition, detected some security issues have fixed.

**2006.** There are no new major releases of platform was released, but growing popularity of platform have attracted funding partners have joined WordPress team. At the August, the first ever WordCamp was held in San Francisco. It was an event that bring people related to WordPress together. Later it will become an annual event.

**2007.** Widgets, new user interface, autosave, spell checks, tagging, update notifications, pretty URLs system and new taxonomy system have been implemented. Adding function allowed to hide blog from search engines have done platform more securable. Moreover, development team revised plugins registered at the official website and all plugins that were inactive more than two years have hided from search results, because have considered as discontinued by their developers.

This year, founder of WordPress, Matt Mullenweg was named number 16 on the list of "The 50 Most Important People of the Web" due to his work on WordPress, angel investing and contributing to freedom of speech (The History of WordPress, 2012).

**2008.** New interface and administration panels were designed by "Happy Cog", a company outside of Wordpress community, which used innovative approach to web design. It allowed the main development team to focus on fixing issues and platform engine improvements. Rising popularity of Social media have brought "BuddyPress" plugin that added community and social media features to users of WordPress. There are many vulnerabilities were found in PHP, MySQL, JavaScript, server applications that WordPress used, but development team every time payed close attention and fixed every security hole they can find.

**2009.** Built-in theme installer, an improved widget user interface and API, image editor introduced. WordPress wins Best Open Source CMS Award. Popularity of platform makes it a prime target for hackers looking to blogs. Hackers are using advantage of the open-source software and area analyzing the source code to detect new vulnerabilities, but development team reacts very fast to new holes found and releases updates simultaneously. Regular updates become number one important tool to protect website from attack.

**2010.** Version 3.0 released and brought major updates: custom taxonomies, custom menu management, post formats and multisite management. At the same year "The WordPress

Foundation” has established as a non-profit organization with the goal of democratizing publishing through Open Source, GPL software (G.R., 2010).

**2011.** WordPress carried out a global survey, asking over 18000 people a basic questions about platform, which showed 14.7% of the internet running WordPress, and 22% of new domains running on WordPress (McKeown, What’s Going On In The WordPress Economy?, 2012). At March 3th, Matt Mullenweg said: ”Say hello to the **Distributed Denial of Service (DDoS)** attack. WordPress.com sustained “the largest and most sustained attack we’ve seen in our six year history,”. It is also important to note that the attack was neutralized the same day (The History of WordPress, 2011). At the June 16, a black box WordPress vulnerability scanner “WPScan” has launched. WPScan team have runned a vulnerability database contains all known detected WordPress core, and its extensions vulnerabilities (WPScan, 2017).

**2012.** Theme customizer and theme preview tool have launched. Several security updates have released that fixed vulnerabiocities and security issues included potential information disclosure, bugs affected nultisite installs with untrusted users.

**2013.** Support of audio and video released. Autosaving and post locking improved. Automatic updates for security and maintenance introduced. The number of websites are still using out-of-date version of platform decreased 20 times. Stronger password meter have added to platform. According to W3Techs research, WordPress is used by 59.9% of all the websites whose content management system they know. This is 22.1% of all websites (The History of WordPress, 2012). WordPress become the strongest leader among CMS-based websites.

**2014.** New version released, the media experience improved and live widget and header previews introduced. Platform was available in over 40 languages, several cross-site-scripting vulnerabilities which could enable anonymous users to comprompise a website have fixed.

**2015.** Emoji support, responsive images and extended character support added. Two big security updates released. One related to cross-site scripting vulnerability, which could allow users with the Contributor or Author role to compromise a site fixed. Second release addressed to six issues included XSS vulnerabilities and potential SQL injection that could be used to compromise a site.

**2016 – now.** WordPress is holding the leader position of website management platform capable for any type of website – from single page website to internet-shops. Security issues are fixing in short time. The WordPress Security Team consists of approximately

50 experts including lead developers and security researchers that consults with well-known and trusted security researches and hosting companies (Rosso, 2015).

As we can see, WordPress reacts to new vulnerabilities and security issues very fast. The platform evolve according to modern trends and gives a confidence to millions of users. Platform automatically informs administrators of websites about available upgrades or release announcements. If they used automatic updates, system informs them by e-mail when update have been completed.

#### **4.4 WordPress structure and installation**

WordPress provides user interface for administration of platform, tools for creation and edition of posts, plugins and themes management etc. Basically, WordPress-based website is the system of posts. The early-concept of platform was that it designed for blogs, and, this is a reason “post” term is using for webpages in WordPress. Technically, text part of posts and all links between them related to CMS are stored into the database, multimedia files are stored into the server and its links are using in webpages structure. Files are necessary for WordPress functionality are also stored in the server. Therefore, WordPress CMS has two main components: MySQL database and core set of files and folders. Combination of it allows to run on the server website based on WordPress.

WordPress platform can use additional components - plugins and themes There are third-part developed extensions which add functionality to the website.

Installation of Wordpress is quite easy – user needs to create empty database on the server, download installation package from official wordpress website, upload files and folders from package to the server or web hosting account (if hosting is using). Next step is accessing of website – WordPress runs and automatically forward to page which helps to create wp-config.php file contains information for connection into MySQL database. After this step user needs to fill the name of website (blog), username, password and e-mail (for the case, when password recovery is needed). After that WordPress is ready for use. During the years the process of WordPress installation have been named “the Famous 5 minute installation” Many web hosting providers offer this installation in automatic mode.

Let’s go further and discover main components of Wordpress more deeper.

#### 4.4.1 Database

WordPress needs MySQL database which links all content in website between each other, and stores all text information of it including users links comments etc. It consists of 12 tables showed on the schema below:

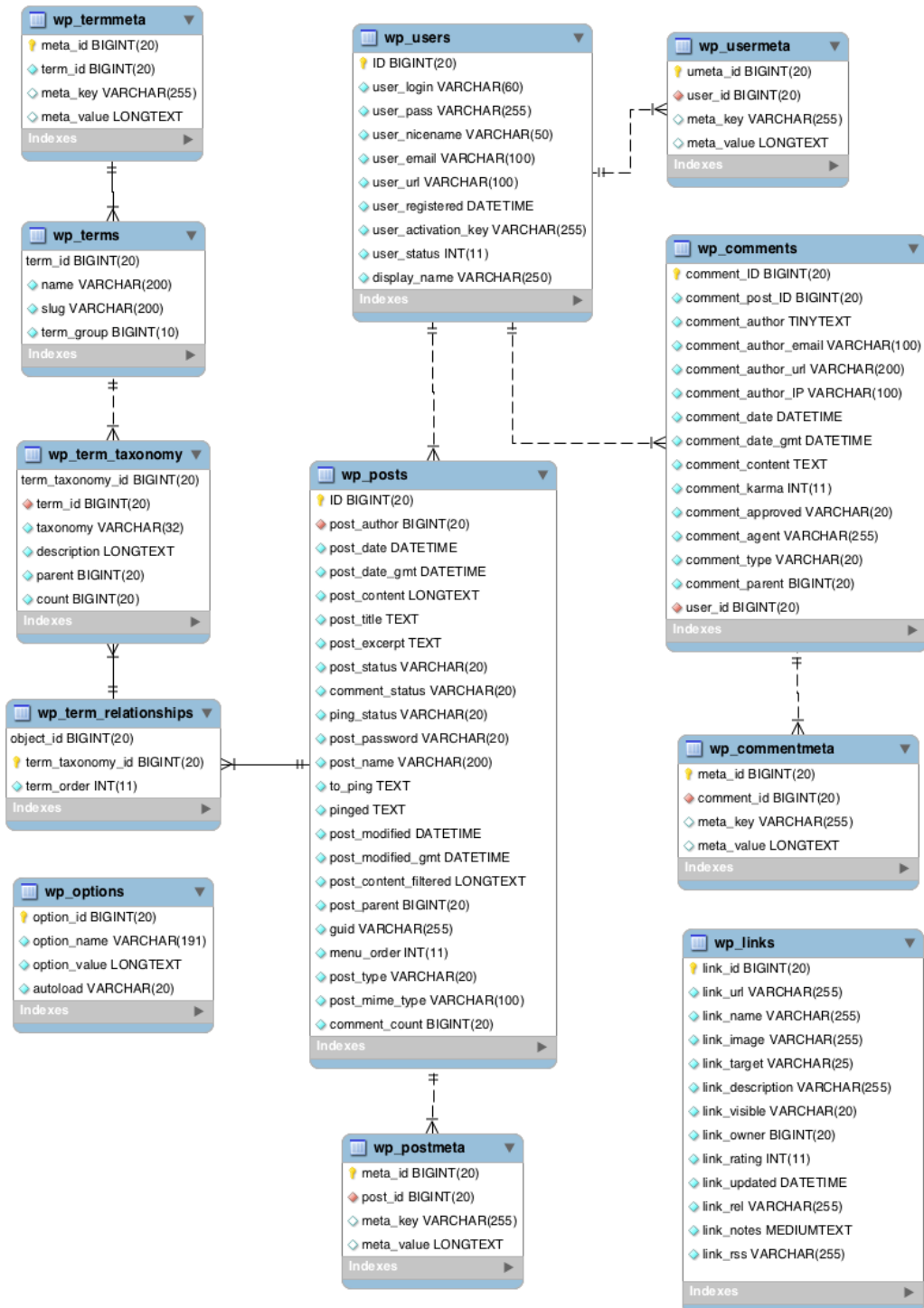


Figure 1. Wordpress information schema (Tour of the WordPress Database, 2016)



Each table has one main function it is responsible, for example, table wp\_users contains all information necessary for user management and linking their records to content.

Description of all tables and its records is available in Appendix A.

I would like to note that the table contains information about users is very important from the point of security protection. It should be accessible by authorized persons only, because the data inside contains usernames and passwords of all users and can be used for access by unauthorized person.

Therefore, database component plays very important role in WordPress website structure, and contains all text data from its content and connections between its elements, which are critical for the whole systems. It is also stores user's information, which needs to be secured from attackers.

#### **4.4.2 Set of files and folders with a code.**

Second important component of the WordPress core is the set of files contains PHP, CSS, JavaScript code required for website functionality. It runs the software for website management and processes all operations – interaction with database, user interface functionality, media elements loading and so on. Most of file's and folder's names start from "wp-" prefix. It helps to identify it, because other files and folders of website allowed to be stored at the same place.

Definitions of the main core files and folders presented in Appendix B.

I would like to take attention to files and folders that can interested for attackers.

**wp-config.php** should be protected from unauthorized access, because it contains critical data allows to get access to MySQL database.

**wp-load.php** has script running after authorization that can be changed to attacker's script, which can cause unpredictable behavior of application.

**admin.php** contains the code of authorisation page that can be replaced and authorisation form might be bypassed

**functions.php** is responsible for website functionality that can be broken or changed to unwanted behavior.

**robots.txt** – the file accessible from the Internet for anyone. It should not contain information that can be useful for attackers.

**.htaccess** – file accessible from the Internet for anyone. It should not contain the paths and files information about that can be useful for attackers.

### 4.4.3 WordPress extensions

**Plugins** – optional extensions for WordPress CMS developed by third part developers. Plugin contains a piece of software written in PHP programming language. It consists of set of functions necessary for extension of functionality and adding new features to WordPress website. Simple and user-friendly plugins management allows creation of additional functionality to website without additional programming knowledge. It makes WordPress CMS flexible and powering tool for website design.

Most of plugins are accessible from official WordPress website. There are more than 45 000 presented there at moment (Plugins, 2017). Administration and installation of plugins is possible from admin console. Installation is also possible by placing plugin's files into wp-content/plugins directory manually.

The main features of plugins are easy management and website functionality extension. Most of developers makes these features as the main target. No doubt, it is useful, but they cannot to consider one serious thing - possible security holes. Official WordPress development team does not control Plugin's developers. It makes plugins one of the most vulnerable components of WordPress. It means they needed periodically checking security issues.

**Themes** - optional extensions for WordPress CMS developed by third part developers. It contains a set of templates and CSS stylesheets used to define how to display content of WordPress website. Administration and installation of themes is possible from admin console. Installation is also possible by placing manually theme's files into wp-content/themes directory. Themes allows to change design of website according to design trends and purpose of content displaying (like website for photographers or business related). Changing of design is simple and does not make impact to website content. It means website design can be changed in several clicks. Themes need to check periodically for vulnerabilities as plugins, because they are producing by third-part developers.

**Widgets** – optional extensions for WordPress CMS looks like small blocks of webpage, which performs special functionality in website. It can be added to widget-ready areas like sidebars, footers/headers. Widgets can be also vulnerable and needed to check for security issues regularly.

#### 4.4.4 Dashboard

**WordPress Dashboard, Console** – a tool gives general overview and management of WordPress based website. This is interface for customize settings and properties of website, manage extensions, create update and delete pages, comments administration and so on. Generally, dashboard is the main control tool for website management.

Console contains many tools for many actions: add, edit or remove blogs (webpages), change themes, manage plugins and widgets, create or update menus, sidebars, footers and headers, manage comments.

Console can be accessed using user credentials – username and password.

#### 4.5 Possible ways to hack

There are many ways to hack WordPress website. They can be divided into several categories by target component or property used for hacking:

- Getting access to MySQL database
- Getting access to WP dashboard
- Getting access to known core files and folders of WordPress
- Using vulnerabilities.

##### 4.5.1 Getting access to SQL database

Database is the core component of WordPress website. It contains all information – from pages content to user management. Getting access to database allows to modify users records and permissions for them, get access to dashboard, change content of webpages get private information about user's profiles like e-mails, passwords etc.

There are several ways to get access to WP database:

- **Getting authorization data from wp-config.php file from WP directory**

File wp-config.php located in WordPress root directory, but it should not be available from the internet, because it contains all information required to connect directly to database. It is recommended to deny access to it or move the file to the level up – if WordPress will not find it, it will attempt to access the file at the level up on the path. Of course, it will be available by FTP client using credentials given by web-hosting provider, directly from hosting control panel or server OS (when dedicated server used). There is also effective way to protect this file from unauthorized access – to deny access using IP-address filtering.

- **Brute Force attack**

This method is classics of hacking. The main idea is to find combination of username and password to get unauthorized access to application used it for authorization. WordPress has several access points, which are using login and password: there are database, dashboard and FTP-server. Database and FTP-server are often managed by web-hosting provider and can be monitored from unauthorized attacks. For example, hosting provider can check IP address of computer requesting database and deny access from external IPs, or use internal interface for database access in web-hosting website. But WordPress dashboard is not protected from ability to try different usernames and passwords. Moreover, there are several ways how to get usernames of WordPress website. Brute-force attack uses a password's dictionary. Attackers are often using dictionaries contain most usable passwords. However it looks simple, this method has one serious disadvantage – if password length is long, a lot of time needed to check all possible passwords – from several days to several years depending the length of password. Most of brute-force attacks are implemented using automated tools working with external dictionaries.

- **SQL-injection**

Most popular common method of database-based website attack is SQL injection. It is using technique of sending SQL-queries, which have modified logic. It allows getting alternative behaviour of application code. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS) (SQL Injection Bypassing WAF, 2017). A simple example of an SQL injection is setting the password value to "password OR 1=1" in SQL query. SQL-injection is the second most common vulnerability found in WordPress (The WordPress Security Learning Center, 2017).

SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database. In 2013, SQLi was rated SQL injection the number one attack on the OWASP top ten (SQL injection, 2017).

- **Malware applications and cheating of authorized persons**

This kind of methods contains tools and techniques aimed to obtaining from user credentials without any interaction with WordPress components. I only mention this possibility because it is not directly related to my research.

#### **4.5.2 Getting access to WordPress dashboard**

Dashboard is a main control tool for website management. Access to this tool gives freedom for any modification of website including possibility to edit important code, for example, core file functions.php. Dashboard is accessible using WordPress user account. All

features are accessible using administrator account. User's accounts functionality decreases automatically according their roles. Administrator can set limitations in dashboard. It is necessary to use strong password for administrator(s) to prevent unauthorised access.

#### **4.5.3 Getting access to core files and folders of WordPress**

Core files contain the code required for website functionality. Access to it allows modifying functionality or deleting some files and breaking website. Access to it is possible via FTP, browser or directly from hosting account or server operating system. Prevention of attack consists of activities of setting up policies disallowing access to core files or allowing from specified IP address only. It is also important to use strong passwords for access to web hosting account and to not share it with persons who are not related to administration. In case when responsible person have been changed, it is necessary to change all passwords he/she used.

#### **4.5.4 Checking WordPress platform, plugins and themes vulnerabilities**

There are many potential vulnerabilities related to WordPress CMS have been detected since it have been developed. Nowadays many people involved to process of detection and fixing of possible treats – from developer's team to private volunteers. There is website, which offers full catalogue of detected vulnerabilities, its descriptions and recommendations for fixing (WPScan Vulnerability Database, 2017). Most of vulnerabilities are fixing by developers. It means, that versions of Wordpress, plugins and themes are needed to update periodically. I want to note, that WordPress CMS is updating regularly and well secured during the time, but extensions (plugins and themes) which are generally produced by third-party developers, have often many security holes and possibilities for attacking.

## 5 Empirical part

This chapter contains practical implementation of ideas and methods from theoretical part. It will be used for testing of the website that have chosen for implementation security audit. It consists from developing strategy of audit and design of tests. Test's design needed to specify its description, input data necessary for the test running, expected outputs. Test results should be clear to evaluate level of security protection for the tested website.

### 5.1 Strategy

My strategy of security audit designed according to the nature of WordPress platform and common techniques for web-application's attacks. It planned to create a number of tests that will implemented step-by-step. There is a plan of actions, which needed to be done:

#### **Information gathering:**

- WordPress version
- WP folder locations and its accessibility
- Plugins installed and its versions
- Themes installed and its versions
- Widgets installed and its versions
- Possible known vulnerabilities detection
- Pages contain forms or file upload blocks
- User authentication form behaviour
- List of users detection
- Open ports detection

#### **Attacks:**

- BruteForce attack
- Cross site scripting (XSS) attack
- SQL-injection attack
- File upload vulnerability attack

#### **Analysis of test results and reporting.**

### 5.2 Test cases design

Test cases design follows two main purposes: detect known vulnerabilities and try to break security barriers as attacker does. Tests have designed for implementation in order they presented, Order following needed to collect all necessary information before the mo-

ment test will start. Order can be changed, but be sure previous tests needed for implementation have been done. Each test contains one or more phases include description of testing actions, input information and expecting outputs.

Results are collecting and storing into test case forms that include to security audit report, which contains test results, its analysis and recommendations for security protection improvements.

### **5.2.1 Information gathering**

The very first step before designing test cases is getting information about target website, as more, as possible. It is necessary for several reasons – to detect possible ways for hacking, to detect known vulnerabilities and ways to fix it, to minimise resources (time and human resources) in audit process and make it more effective. There is a list of data to be collected:

- Version of WordPress, which can be detected by several ways – inspecting code of webpage, online scanning services, or special tools like WPScan I used to use.
- Installed plugins and themes, which can contain vulnerabilities. It is necessary to check Vulnerabilities database and descriptions for each of installed extensions for better understanding the nature of the whole system of website.
- Basic information about website owner, hosting provider, is using. WhoIs service provides a lot of information including our target.
- List of users registered in SQL database. It is the thing needed for checking their passwords for possibility to guess using machine tools (brute force attack test)
- Opened ports detection – it is useful information to find communication tunnels using by website.
- File upload components (flash based or something else)
- Search engines information, which can provide more useful information about website.

WPScan - is powerful software tool for information gathering of WordPress website. It allows to scan most known vulnerabilities of website, plugins, themes, and attempt to get usernames from database.

## Test case 1. Gathering information.

**Description:** Gathering information needed for security audit.

**Precondition:** Check website for possible known vulnerabilities and useful information for potential attack.

**Goal:** Version of WordPress, installed plugins and themes and its versions, list of WordPress users, possible known vulnerabilities, page's URLs with forms or file upload blocks, ports opened, critical behaviour of user authentication page, Whols information.

**Table 1.** Test case 1. Gathering information.

Step	Description	Inputs	Expected outputs
1	Version of WordPress	Website URL: www.digitaluniversity.fi	Version of WP
2	Theme(s) installed	Website URL: www.digitaluniversity.fi	Theme(s) name(s)
3	Plug-in(s) installed	Website URL: www.digitaluniversity.fi	Plug-in(s) name(s)
4	Hosting provider	Website URL: www.digitaluniversity.fi Whols online service	Hosting provider name, IP
5	List of users	Website URL: www.digitaluniversity.fi	List of users
6	Opened ports	Website URL: www.digitaluniversity.fi	Opened ports
7	File upload components	Website URL: www.digitaluniversity.fi	File upload URLs
8	Forms on the page(s)	Website URL: www.digitaluniversity.fi	URL contain forms
9	User password recovery page behaviour	Website URL: www.digitaluniversity.fi	Possibility of de- tection usernames (ex- ists or not)

This test collects a lot of information will used in further tests and uses to create better strategy for testing opportunity to hack of website. 1-3 steps from the Table 1 are using WPScan utility. It is common test of known vulnerabilities and security issues of WordPress-based website. The next step is checking website URL at Whols service available at <http://who.is/>. It shows information about website owner and hosting is using that helps



to identify owner person or organisation – in case when somebody asks to do security audit – we can check, if person or organisation is the same entity. Hosting provider information is useful for possible attack to the server of hosting provider. 5<sup>th</sup> step from the Table 1 is using WPScan utility again with “—enumerate u” argument that scans websites for usernames available to detect. This data needed for SQL injection attack. 6<sup>th</sup> step checks opened ports that scans effectiveness of ports routing. The number and values of opened ports depend from website functionality and administration settings. Steps 7-8 from the Table 1 needed to find webpages URLs suitable for cross scripting and file upload attacks. And 9<sup>th</sup> step checks the behaviour of the logics of login and password recovery forms. For example, when you fill username or e-mail for password recovery field system shows, if username or e-mail filled exists or not. It allows to attempt to guess usernames. Moreover, many of usernames are usually showing with standard URL for WordPress like <https://www.website.com/?author=1>, where 1 is the number of user registered. Often, user are using number one has administrator rights and username admin, because it is standard, preinstalled user with administrator role in WordPress.

The information collected from this test will be used also at all further tests.

## 5.2.2 WordPress folder locations and its accessibility

This test purpose is to detect accessible critical core files and folders. For this purpose several checks are needed: search engines scanning using several queries, analysis of data from file “robots.txt” stored in the core directory of website, and attempt to access core files and folders using non standard URLs.

### Test case 2. WordPress folder locations and its accessibility

**Description:** Checking accessibility of WP core folders

**Precondition:** Several methods are using for checking accessibility of WordPress core files and folders

**Goal:** Detection of WordPress core folders and files are accessible.

**Table 2.** Test case 2. WordPress core folders and files accessibility check.

Step	Description	Inputs	Expected outputs
1	Google	Website URL: www.digitaluniversity.fi	WP-core folders accessible
2	Robots.txt	www.digitaluniversity.fi/robots.txt	Allowed/disallowed paths
3	Using non-standard paths	Website URL: www.digitaluniversity.fi	Allowed/disallowed paths

Step 1 from the Table 2 shows search results using google (or similar search engine) from queries contains “wp-“ in the path of related website. Maybe core folders are not closed. Step 2 allows to check robots.txt which is located in the root directory of the website. This file purpose is for search engines, but it can contain useful information for potential attackers, for example directories contain critical information – paths or passwords.

Final step from the Table 2 is attempt to experiment with paths to try access to closed folders using knowledge of path-building and several ideas, for example: [www.wordpress-website.com/wp-admin/admin-ajax.php?action=revslider\\_show\\_image&img=./wp-admin](http://www.wordpress-website.com/wp-admin/admin-ajax.php?action=revslider_show_image&img=./wp-admin) where directory “wp-admin” can be accessible using special construction. Information from this is necessary to use in further tests.

### 5.2.3 Extensions vulnerabilities test

Here we need to check possible known vulnerabilities of platform can be used by attackers. Many of extensions are developing by third-part developers are not focus attention to security, which allows using its holes to discover ways for attack. Each extension – plugin, theme or widget needs to check for possible known vulnerabilities using scan utilities or search engines and discover its code for SQL-queries or paths to core-folders of WP.

#### Test case 3. Extensions vulnerabilities detection

**Description:** Attempt to find known vulnerabilities in themes, plugins or widgets are using.

**Precondition:** Check extensions for known vulnerabilities, inspect the code of plugins and themes, and try to find SQL queries or credentials using

**Goal:** Getting additional information for further attacks.

**Table 3.** Test case 3. Extension vulnerabilities detection.

Step	Description	Inputs	Expected outputs
1	Check known vulnerabilities for plugins or themes detected in search engines or scan utilities	Extension name	Possible vulnerability of component
2	Inspect the code of component, if possible	Code of component	SQL statements or credentials (id= or ...)
3	Analysis of found information	Found information	Test case for audit

Table 3 consists of three steps: The first one scans extensions found in the Test 1 for known vulnerabilities using WPScan utility ore search engine. Step 2 is optional and intended for experienced users – if the code of extension is available, it needs to check for critical elements could be accessible – SQL statements, user credentials and so on. Last step analyses collected information for using in further tests.

#### 5.2.4 Brute force user's attack test

Brute force is most popular attack today. It is based on attempts to guess login and password of users using machine-processing tools. WPscan utility can be used for that purpose. It needs URL of website, username and a dictionary contains password for guessing. I will use most popular and common dictionary "rockyou" can be downloaded from [wiki.skullsecurity.org](http://wiki.skullsecurity.org), but any other dictionary is possible to use.

#### Test case 4. Brute Force attack

**Description:** Attempt to guess name and password to access admin console using password-dictionaries.

**Precondition:** WPscan using usernames

**Goal:** Getting access to console.

**Table 4.** Test case 4. Brute force attack.

Step	Description	Inputs	Expected outputs
1	WPscan	admin	password
2	WPscan	user	password
3	WPscan	...	

Brute force attack test needs usernames for running. It attempts to find password from dictionary for specified username. List of usernames should be provided from test case 1 "Gathering information". Process can take a lot of time, especially, if list of users and/or password dictionary are big. Table 4 consists of a number of steps – one check for each detected username.

This test helps also to check password's complexity for all users, because it is not recommended to use short or simple passwords. All passwords are checking for its possibility for fast hacking. But shorter way to achieve it – using internal WordPress password checker for all users. Data collected in this test is analysing and using for security report.

### 5.2.5 Discovering the scripts of Website.

Sometimes administrators are not consider importance of file permissions management, which allows to access the code from core files of WordPress. Website can contain un-used or unblocked functionality of common-used components with scripts, which can be used for attack. This test attempts to check website files are using scripts and core files.

#### Test case 5. Attempt to access files and scripts.

**Description:** Attempt to find scripts or paths, which have access to core elements

**Precondition:** Inspect the code and website paths for possible interaction with core elements.

**Goal:** Getting access to wp-directories or to database.

**Table 5.** Test case 5. Discover the scripts of website.

Step	Description	Inputs	Expected outputs
1	Check website URLs for wp-paths in search engine	Website URL, wp-path's Statements: /wp-content/ /wp-content/plugins/ /wp-content/themes/ /uploads/ /images/	Paths accessible detected
2	Inspect the code of scripts	Code of script	SQL statements or credentials (id= or user= or paths) +, no SQL statements credentials
3	Analysis of found information	Found information	Test case for audit

These tests are necessary for checking scripts or elements of code content can be used for XSS attacks or SQL injection attack. Step 1 of the Table 5 checks several URLs need to be checked for availability paths contain "wp-" folders where script files can be stored and contain id's or SQL-statements used. Scripts are vulnerable, because this way of attack comes from interaction between user and website. Any website shows in browser as html-document. Most of html documents today are using DOM structure, which provides a representation of the document as a structured group of nodes and objects that have

properties and methods. Essentially, it connects web pages to scripts or programming languages. The nature of the DOM allows modifying existing code and running it. It is a kind of XSS-attack (Introduction to the DOM, 2017). Inspection of scripts found running at Step 2 of the Table 5 allows to check the code for elements can be used for XSS attack or SQL injection attack. If elements have found, it needs to analyze it for possibility to use at the further tests.

## 5.2.6 SQL Injection and common SQL attack's test

This test checks the website for resistance to SQL attack. I need to consider that most of possibilities to attack website using SQL injection technique have been already fixed and last versions of WordPress platform are generally secured from that attack, but, maybe installed plugins or themes or some components will have holes to do that.

### Test case 6. SQL injection attack

**Description:** Attempt to get name and password to access admin console using SQL-injection method.

**Precondition:** Using traffic monitor and SQL injection tool to get username and password

**Goal:** Getting access to console.

**Table 6.** Test case 6. SQL injection attack

Step	Description	Inputs	Expected outputs
1	Check actions for request/response can be used for injection	Website URLs with forms or scripts	Data for injection
2	Try to get database content via SQL injection tools	Data for injection	Access to database
3	Get user's credentials	Access to database	User's credentials

SQL attack test checks website for possibility to execute malicious SQL statements for getting access to database or records contain user's credentials. First step from the Table 5 needs information collected from test cases №1 and №5 necessary for running this step. The 2<sup>nd</sup> step implements SQL injection attack using special software for this purpose. I plan to use powerful tool for SQL injection testing named "sqlmap" preinstalled in Kali Linux that can be used for this test with a couple of commands. My method is using URLs contain arguments can be queried from database. For example:

```
sqlmap -u http://www.example.com/?post=123
```

URL is using in example attempts to query post number 123 from the SQL database.

If injection is succeed, software will show database have accessed and will allow to do the 3<sup>rd</sup> step of the Table 3 that returns users credentials contain usernames and password's hashes. It allow to get unauthorized access to WordPress console. Results of this test case will be reflected in final report.

### 5.2.7 XSS attack test

Cross Site Scripting attacks (also named XSS-attacks) are using most common vulnerabilities found in WordPress plugins by a significant margin (The WordPress Security Learning Center, 2017). The idea of this attack is using the code of webpage script can be modified using malicious scripts injection and runned from browser side script. It allows to access website bypassing access controls policies.

#### Test case 7. XSS attack test

**Description:** Attempt to get access to Admin console or database using XSS attack.

**Precondition:** Try to implement XSS attack, if possibility to interact with WordPress elements found core

**Goal:** Getting access to console or database.

**Table 7.** Test case 7. XSS attack.

Step	Description	Inputs	Expected outputs
1	Inspect the code in browser, find potential elements for attack	Website URL	Elements for attack
2	Modify code for attack	Elements for attack	Modified code
3	Run modified code	Modified code	Access to console or database

XSS attack needs a code for its modification, so we will use “uniscan” utility preinstalled in Kali Linux, which scans website files with scripts for content availability for XSS attack at step 1 of the Table 7. Information collected from test cases №1 and №5 can be used also. If vulnerabilities will found, we can use it at the steps 2 and 3 of the Table 7. XSS attack is the most common vulnerability found in the WordPress (How to Prevent Cross Site Scripting Attacks, 2017) It means, this kind of attack needs to pay special attention.



### 5.2.8 File upload test

File upload vulnerability us using to get access to the WP-core folders and files. The idea is to upload file with malicious code via this functionality, and, after that, run it from browser.

#### Test case 8. File upload test.

**Description:** Attempt to get users credentials via file upload components by uploading script

**Precondition:** Attack website using file upload vulnerability

**Goal:** Getting access to console or database.

**Table 8.** Test case 8. File upload attack.

Step	Description	Inputs	Expected outputs
1	Inspect the website, find file upload components	Website URL	File upload components
2	Try to upload simple script	Upload component	Success
3	Try to upload malicious script	Upload component	Success
4	Try to run script from browser	Upload URL	Credentials for access

This test needs information collected from Step 7 of test case 1. If information collected, we can use it at steps 2-4 of Table 8. This test has similar logics like XSS attack, but the script injected into the file uploaded.

### 5.3 Preparing to testing.

Design of test have done. At this chapter I have tried to analyse information collected in theoretical part and use it for the set of tests. Of course, it can not guarantee 100% confidence of my solution, but we need to consider, there is nothing is ideal in the World and every solution needs to be updated every time, like, for example, WordPress platform.

The next phase will implement set of designed tests on the real website. It will black box testing method, because I don't know anything about developing of website have chosen for the test. Results will be placed into security report. I will discuss about my feelings and observations during test running.

## 6 Security audit report 1

Here will be showing tests were running and test results and its analysis. All recommendations of security protection improvements presented at the end of this chapter.

### 6.1 Test case 1 implementation. Gathering information.

#### 6.1.1 Running the test 1

Test date: 08.05.2017

**Table 9.** Test case 1 implementation. Gathering Information.

Step	Description	Inputs	Outputs
1	Version of Wordpress	Website URL: www.digitaluniversity.fi	4.7
2	Theme(s) installed	Website URL: www.digitaluniversity.fi	Eduma http://digitaluniversity.fi/wp-content/themes/eduma/
3	Plug-in(s) installed	Website URL: www.digitaluniversity.fi	Plug-ins detected
4	Hosting provider	Website URL: www.digitaluniversity.fi WhoIS	AinaCom Oy 84.34.147.34
5	List of users	Website URL: www.digitaluniversity.fi	Several users detected
6	Opened ports	Website URL: www.digitaluniversity.fi	checked
7	File upload components	Website URL: www.digitaluniversity.fi	Not found
8	Forms on the page(s)	Website URL: www.digitaluniversity.fi	Found (password reset, authentication and search field)
9	User password recovery page behaviour	Website URL: www.digitaluniversity.fi	Username existence can be detected

First tests are collecting information about website, which can be used for known vulnerabilities detection and attempts to attack. There are many ways to find holes in website security and results can be different according to knowledge, experience of attacker. My scanning has been started from WPScan utility. It is a black box WordPress vulnerability scanner preinstalled on Kali Linux distribution. Utility is using a command line interface with different arguments. More information is available on the official website (WPScan, 2017).

First of all, I've downloaded the configuration file from the official GitHub repository of the utility and placed it into the configuration folder of my Linux, because it didn't exist. Besides, I've updated the database using `wpscan --update` command. After that, I've started my testing.

First command allows to get common information about website:

```
wpscan --url www.digitaluniversity.fi
```

```
root@sqvaioow:~# wpscan digitaluniversity.fi
```

---

```

  _____
 /         \
|   WPSCAN   |
 \         /
  _____

WordPress Security Scanner by the WPScan Team
      Version 2.9.1
Sponsored by Sucuri - https://sucuri.net
@_WPScan_, @ethicalhack3r, @erwan_lr, pvd1, @_FireFart_

```

---

```
[i] The remote host tried to redirect to: https://www.digitaluniversity.fi/?cache-flush=1494313390.339
[?] Do you want follow the redirection ? [Y]es [N]o [A]bort, default: [N]Y
[+] URL: https://www.digitaluniversity.fi/?cache-flush=1494313390.339/
[+] Started: Tue Jul 26 11:09:43 2016

[+] robots.txt available under: 'https://www.digitaluniversity.fi/robots.txt'
[+] Interesting entry from robots.txt: https://www.digitaluniversity.fi/wp-admin/admin-ajax.php?cache-flush=1494313390.339/
[!] The WordPress 'https://www.digitaluniversity.fi/readme.html' file exists exposing a version number
[+] Interesting header: SERVER: Apache/2
[+] XML-RPC Interface available under: https://www.digitaluniversity.fi/xmlrpc.php

[+] WordPress version 4.7 identified from readme (Released on 2016-12-06)
[!] 18 vulnerabilities identified from the version number

[!] Title: WordPress 4.3-4.7 - Potential Remote Command Execution (RCE) in PHPMailer
Reference: https://wpvulndb.com/vulnerabilities/8714
Reference: https://www.wordfence.com/blog/2016/12/phpmailer-vulnerability/
Reference: https://github.com/PHPMailer/PHPMailer/wiki/About-the-CVE-2016-10033-and-CVE-2016-10045-vulnerabilities
Reference: https://wordpress.org/news/2017/01/wordpress-4-7-1-security-and-maintenance-release/
[!] Fixed in: 4.7.1
```

Figure 2. Information gathering using wp-scan utility

Figure 2 reflects utility output. As we can see, version of WordPress has been detected. It is not the latest, but close to it, which means website updates occasionally.

Utility showed also installed plugins, theme and detected vulnerabilities. Detailed review and analysis is below. Now we have information for 1-3 steps of the Table 9.

We will return to WPScan utility later, but now is the time for the 4<sup>th</sup> step – gathering information using who.is service, which provides information about website holder, DNS records, date of creation/expiration of website and so on.

Usage is very simple and intuitive – user needs to fill URL into the field on the page and press the search button, which runs the process of scanning.

All information requested from WHOIS database is showing into the webpage. User can switch between 5 tabs as showed on the Figure 3.

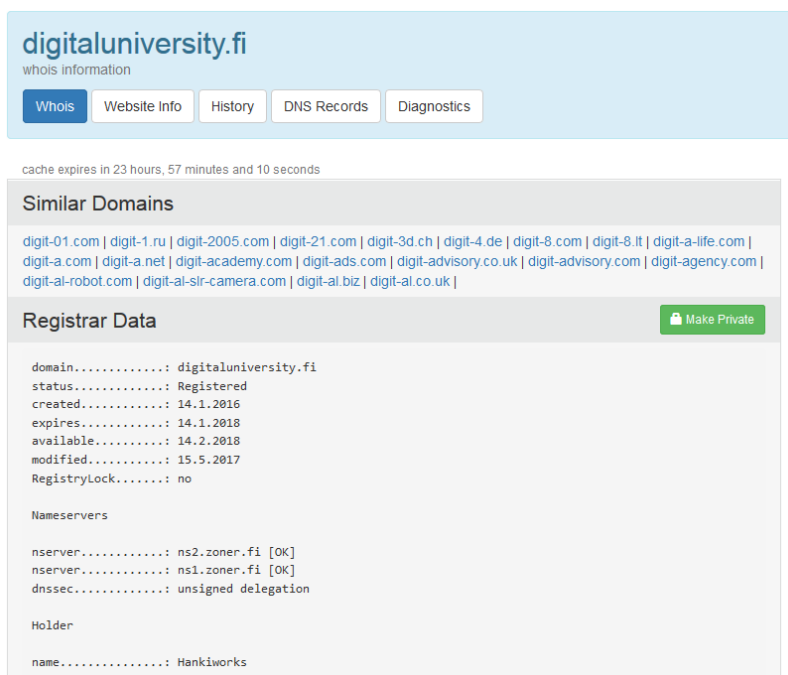


Figure 3. Whois information service

It is necessary tool to make sure, that real website owner ordered security research and test website according the legislation. It is interesting fact, that website located at the hosting account in another country, it should be covered by legislation of country it is located.

Step 5 from the Table 9 is attempt to get a list of usernames of the website. For that purpose, I have used command:

```
wpscan --url www.digitaluniversity.fi --enumerate u of WPScan utility.
```

```
[+] Enumerating usernames ...  
[+] Identified the following 5 user/s:  
+-----+-----+-----+  
| Id | Login | Name |  
+-----+-----+-----+  
| 6 | jukkapekkavuorinen | jukkapekkavuorinen |  
| 7 | haollahd | haollahd |  
| 8 | buse | buse |  
| 9 | heimo88 | heimo88 |  
| 10 | henri-heikkinencaverion-com | Henri |  
+-----+-----+-----+
```

Figure 4. Usernames identification

Figure 4 is showing that 5 usernames have been detected. It is strange, because usually utility shows full list of users or nothing showed. It is recommended to discover properties

of users listed on the Figure X to make them hidden. Maybe it is not critical to show their names, but it is necessary to check their roles in user hierarchy, if somebody of them is administrator, successful brute force attack can give full access to website dashboard.

6th step of the Table 9 is opened ports scanning. For that purpose, I have used “nmap: utility. Nmap is free and open source utility for network discovery and security auditing (Nmap, 2017). It have preinstalled in my Kali Linux distributive. Utility is using command line interface. I used command:

```
nmap www.digitaluniversity.fi
```

```
root@sqvaioiw:~# nmap digitaluniversity.fi
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2016-07-26 04:46 EDT
Nmap scan report for digitaluniversity.fi (84.34.147.34)
Host is up (0.13s latency).
rDNS record for 84.34.147.34: www14.zoner.fi
Not shown: 988 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
110/tcp   open  pop3
143/tcp   open  imap
443/tcp   open  https
465/tcp   open  smtps
587/tcp   open  submission
993/tcp   open  imaps
995/tcp   open  pop3s
2222/tcp  open  EtherNetIP-1
35500/tcp closed unknown
Nmap done: 1 IP address (1 host up) scanned in 12.86 seconds
```

Figure 3. Scanning opened ports using nmap utility

Opened ports have been detected. Figure 5 shows full list of it. It is not critical, because this information cannot be hidden, but website administrator should regularly check opened ports and its frequent scanning attempts from outside need to be detected, because somebody can use it for attack. In that case it is recommended to block detected IP using special security plugins or other solutions. In case, when ports are opened but are not using, it is recommended to close unusable ports.

For the last steps 7-8-9 of the table 9, I did not use any utilities and discovered website manually. I found some interesting things. First, website offers registration for users without administrator approve. It allows using console for website management from the point of user role. From the attacker point, this kind of audit helps to find possibilities for attack hidden without registration, and use user credentials for brute force testing and audit of password recovery components. Finally, I have checked password recovery form, which asks e-mail or username and I have tried to fill different usernames or e-mails – in case when the name doesn't exist it says about that fact, it allows to check existence of the name in database which can be used for attempt to guess name and password, or use

vulnerability of exchange e-mail for password recovery link to another. This vulnerability have been detected by WPScan utility and included into the list of detected vulnerabilities. It may be dangerous, because vulnerability have not fixed yet.

### 6.1.2 Analysis of the test 1 results

There is a list of detected vulnerabilities in WordPress platform:

**Vulnerability:** php version info is accessible in readme.html

**Description:** File readme.html is accessible via HTTP on the default location and contains detailed php version info.

**Recommendation:** Hide information contains versions.

**Vulnerability:** WordPress 4.3-4.7 - Potential Remote Command Execution (RCE) in PHP-Mailer

**Description:** In the vulnerable version of PHPMailer, the sender email address is passed unescaped to a shell command. An attacker could include shell commands in the sender email that execute malicious code on a target machine or website (Critical Vulnerability in PHPMailer. Affects WP Core., 2016).

**Fixed in:** 4.7.1

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 4.7 - User Information Disclosure via REST API

**Proof:** <https://digitaluniversity.fi/wp-json/wp/v2/>

**Description:** The new WordPress REST API allows anonymous access. One of the functions that it provides is that anyone can list the users on a WordPress website without registering or having an account (WordPress 4.7.1 Security and Maintenance Release, 2011). To see this function in action you can check hyperlink showed above.

**Fixed in:** 4.7.1

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 2.9-4.7 - Authenticated Cross-Site scripting (XSS) in update-core.php

**Description:** Multiple cross-site scripting (XSS) vulnerabilities in wp-admin/update-core.php in WordPress before 4.7.1 allow remote attackers to inject arbitrary web script or HTML via the (1) name or (2) version header of a plugin (CVE-2017-5488, 2017).

**Fixed in:** 4.7.1

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 4.7 - Cross-Site Request Forgery (CSRF) via Flash Upload

**Description:** Cross-site request forgery (CSRF) vulnerability in WordPress before 4.7.1 allows remote attackers to hijack the authentication of unspecified victims via vectors involving a Flash file upload (CVE-2017-5489, 2017).

**Fixed in:** 4.7.1

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 3.4-4.7 - Stored Cross-Site Scripting (XSS) via Theme Name fallback

**Description:** Cross-site scripting (XSS) vulnerability in the theme-name fallback functionality in wp-includes/class-wp-theme.php in WordPress before 4.7.1 allows remote attackers to inject arbitrary web script or HTML via a crafted directory name of a theme, related to wp-admin/includes/class-theme-installer-skin.php (CVE-2017-5490, 2017).

**Fixed in:** 4.7.1

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress <= 4.7 - Post via Email Checks mail.example.com by Default

**Description:** wp-mail.php in WordPress before 4.7.1 might allow remote attackers to bypass intended posting restrictions via a spoofed mail server with the mail.example.com name (CVE-2017-5491, 2017).

**Fixed in:** 4.7.1

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 2.8-4.7 - Accessibility Mode Cross-Site Request Forgery (CSRF)

**Description:** Cross-site request forgery (CSRF) vulnerability in the widget-editing accessibility-mode feature in WordPress before 4.7.1 allows remote attackers to hijack the authentication of unspecified victims for requests that perform a widgets-access action, related to wp-admin/includes/class-wp-screen.php and wp-admin/widgets.php (CVE-2017-5492, 2017).

**Fixed in:** 4.7.1

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 3.0-4.7 - Cryptographically Weak Pseudo-Random Number Generator (PRNG)

**Description:** wp-includes/ms-functions.php in the Multisite WordPress API in WordPress before 4.7.1 does not properly choose random numbers for keys, which makes it easier for remote attackers to bypass intended access restrictions via a crafted (1) site signup or (2) user signup (CVE-2017-5493, 2017).

**Fixed in:** 4.7.1

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 4.2.0-4.7.1 - Press This UI Available to Unauthorised Users

**Description:** wp-admin/includes/class-wp-press-this.php in Press This in WordPress before 4.7.2 does not properly restrict visibility of a taxonomy-assignment user interface, which allows remote attackers to bypass intended access restrictions by reading terms (CVE-2017-5610, 2017).

**Fixed in:** 4.7.2

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 3.5-4.7.1 - WP\_Query SQL Injection

**Description:** SQL injection vulnerability in wp-includes/class-wp-query.php in WP\_Query in WordPress before 4.7.2 allows remote attackers to execute arbitrary SQL commands by leveraging the presence of an affected plugin or theme that mishandles a crafted post type name (CVE-2017-5611, 2017).

**Fixed in:** 4.7.2

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 4.3.0-4.7.1 - Cross-Site Scripting (XSS) in posts list table

**Description:** Cross-site scripting (XSS) vulnerability in wp-admin/includes/class-wp-posts-list-table.php in the posts list table in WordPress before 4.7.2 allows remote attackers to inject arbitrary web script or HTML via a crafted excerpt (CVE-2017-5612, 2017).

**Fixed in:** 4.7.2

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 4.7.0-4.7.1 - Unauthenticated Page/Post Content Modification via REST API

**Description:** This privilege escalation vulnerability affects the WordPress REST API that was recently added and enabled by default on WordPress 4.7.0. One of these REST endpoints allows access (via the API) to view, edit, delete and create posts. Within this particular endpoint, a subtle bug allows visitors to edit any post on the site (Content Injection Vulnerability in WordPress, 2017).

**Fixed in:** 4.7.2

**Recommendation:** Update WordPress to the latest version



**Vulnerability:** WordPress 3.6.0-4.7.2 - Authenticated Cross-Site Scripting (XSS) via

**Description:** In WordPress before 4.7.3, there is authenticated Cross-Site Scripting (XSS) via Media File Metadata. This is demonstrated by both (1) mishandling of the playlist shortcode in the wp\_playlist\_shortcode function in wp-includes/media.php and (2) mishandling of meta information in the renderTracks function in wp-includes/js/mediaelement/wp-playlist.js (CVE-2017-6814, 2017).

**Fixed in:** 4.7.3

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 2.8.1-4.7.2 - Control Characters in Redirect URL Validation

**Description:** In WordPress before 4.7.3 (wp-includes/pluggable.php), control characters can trick redirect URL validation (CVE-2017-6815, 2017).

**Fixed in:** 4.7.3

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 4.7.0-4.7.2 - Authenticated Unintended File Deletion in Plugin Delete

**Description:** in WordPress before 4.7.3 (wp-admin/plugins.php), administrators using the plugin deletion functionality can delete unintended files (CVE-2017-6816, 2017).

**Fixed in:** 4.7.3

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 4.0-4.7.2 - Authenticated Stored Cross-Site Scripting (XSS) in YouTube URL Embeds

**Description:** In WordPress before 4.7.3 (wp-includes/embed.php), there is authenticated Cross-Site Scripting (XSS) in YouTube URL Embeds (CVE-2017-6817, 2017).

**Fixed in:** 4.7.3

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 4.7-4.7.2 - Cross-Site Scripting (XSS) via Taxonomy Term Names

**Description:** In WordPress before 4.7.3 (wp-admin/js/tags-box.js), there is cross-site scripting (XSS) via taxonomy term names (CVE-2017-6818, 2017).

**Fixed in:** 4.7.3

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 4.2-4.7.2 - Press This CSRF DoS

**Description:** In WordPress before 4.7.3, there is cross-site request forgery (CSRF) in Press This (wp-admin/includes/class-wp-press-this.php), leading to excessive use of server resources. The CSRF can trigger an outbound HTTP request for a large file that is then parsed by Press This (CVE-2017-6819, 2017).

**Fixed in:** 4.7.3

**Recommendation:** Update WordPress to the latest version

**Vulnerability:** WordPress 2.3-4.7.5 - Host Header Injection in Password Reset

**Description:** The vulnerability stems from WordPress using untrusted data by default when creating a password reset e-mail that is supposed to be delivered only to the e-mail associated with the owner's account. This could possibly allow the attacker to intercept the email containing the password reset link in some cases requiring user interaction as well as without user interaction (WordPress-Exploit-4-7-Unauth-Password-Reset-0day-CVE-2017-8295, 2017).

**Attention:** Vulnerability is not fixed at moment

**Recommendation:** Change logic of password recovery form to exclude possibility of user existence check and hide users detected in user enumeration part of information gathering test to exclude possibility of using existing usernames.

### **6.1.3 Recommendations for security protection improvement**

Hide usernames identified in the username's enumeration test if necessary.

Website needed to update CMS to the latest version for better level of protection.

It is recommended to repeat vulnerabilities scanning after updating to be sure vulnerabilities listed have been destroyed.

Change the logic of behavior of the password recovery form.

## 6.2 Test case 2 implementation. WP-folders location and its accessibility test

### 6.2.1 Running the test 2

Test date: 09.05.2017

Table 10. Test case 2 implementation. WP

Step	Description	Inputs	Outputs
1	Google	Website URL: www.digitaluniversity.fi	Flash element found
2	Robots.txt	-	See Figure 7 below
3	Using non-standard paths	Website URL: www.digitaluniversity.fi	Failed

Search engines become very important and useful tool and its importance is growing every time. For second test I have used google search engine to try detect core directories of the website which can be accessible. The idea is search engines indexing data at all website pages can be accessible. This process contains, collecting, parsing and storing data to facilitate fast and accurate information retrieval (Search engine indexing, 2017). It allows to get a lot of useful information for potential attacker using several queries. The language of queries is clear and intuitive, for example, argument "inurl:abcd" checks pages only which have "abcd"-sequence of characters in URL.

The objective of the 1st step from the table 10 is to check accessible core folders of website. I have used command: `site: www.digitaluniversity.fi inurl:wp-*` which searches all accessible folders starts from "wp-" of website we are testing.

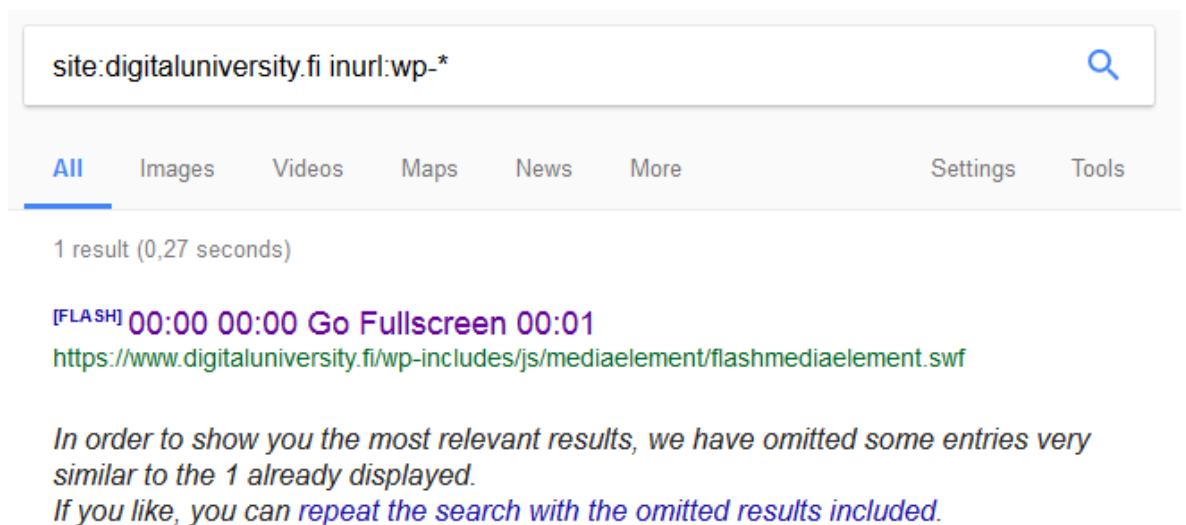


Figure 4. Searching "wp-" folders using Google

As we can see on the figure 6, one result only have been presented, but, maybe it will be useful later.

I've also checked several another search engines like Bing, DuckDuckGo, Yandex for similar queries, but not valuable results have been found.

2nd step of the table 10 is checking content of "robots.txt" file, which tells to search engines which files or folders are allowed/dissalowed to be indexed. File is always located in the root folder of website.

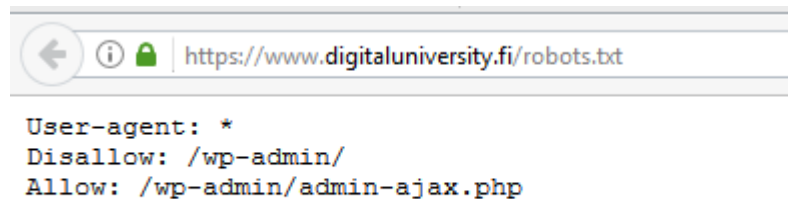


Figure 5. File robots.txt

Figure 7 shoes that "wp-admin" directory is dissalowed for indexing, but "admin-ajax.php" file located inside this directory needs to be indexed.

Final step checking website for access core directory using non-standard paths. I've used paths are using format:

[www.digitaluniversity.fi/wp-admin/admin-ajax.php?action=revslider\\_show\\_image&img=../wp-admin](http://www.digitaluniversity.fi/wp-admin/admin-ajax.php?action=revslider_show_image&img=../wp-admin) but specified paths had not accessed.

### 6.2.2 Analysis of the test 2 results

Gathered information will attempt to use in further tests.

### 6.3 Test case 3. Extensions vulnerabilities detection

#### 6.3.1 Running the test 3

Test date: 12.05.2017

Table 11. Test case 3 implementation. Extensions vulnerabilities detection.

Step	Description	Inputs	Outputs
1	Check known vulnerabilities for plugin or theme in google	Extension name	See the list of vulnerable plug-ins below
2	Check if solution already exists	Vulnerability name	See the list of vulnerable plug-ins below
3	Inspect the code of component if solution not found (optional)	Code of component	No vulnerable plug-ins without solution found
4	Analysis of found information (optional)	Found information	No vulnerable plug-ins without solution found

Themes and plug-ins have detected at the Test case 1 "Information gathering". WPScan utility shows information about all detected plugins including vulnerabilities if exist. It allowed me to check vulnerable extensions only from Test case 1 "Information gathering" results. I have checked information related to each vulnerability and found, that all vulnerabilities can be destroyed by updating current WordPress and plug-ins versions to the latest, but it is necessary to check again updated version of website.

#### 6.3.2 Analysis of the test 3 results

Vulnerabilities have detected in plug-ins:

**Vulnerability:** Connections Business Directory <= 0.7.9.3 - Pagination URL H&ling XSS

**Description:** This plugin is prone to a Pagination URL H&ling XSS vulnerability.

**Fixed in:** 0.7.9.4

**Recommendation:** Update plug-in to the latest version

**Vulnerability:** Connections <= 0.7.1.5 - Unspecified Security

**Description:** Unspecified vulnerability in the Connections plugin before 0.7.1.6 for WordPress has unknown impact and attack vectors (CVE-2011-5254, 2013).

**Fixed in:** 0.7.1.5

**Recommendation:** Update plug-in to the latest version

**Vulnerability:** Connections <= 8.5.8 - Reflected Cross-Site Scripting (XSS)

**Description:** Cross-site scripting (XSS) vulnerability in includes/admin/pages/manage.php in the Connections Business Directory plugin before 8.5.9 for WordPress allows remote attackers to inject arbitrary web script or HTML via the s variable (CVE-2016-0770, 2015).

**Fixed in:** 8.5.9

**Recommendation:** Update plug-in to the latest version

**Vulnerability:** WordPress Slider Revolution Local File Disclosure

**Description:** Directory traversal vulnerability in the Elegant Themes Divi theme for WordPress allows remote attackers to read arbitrary files via a .. (dot dot) in the img parameter in a revslider\_show\_image action to wp-admin/admin-ajax.php. NOTE: this vulnerability may be a duplicate of CVE-2014-9734 (CVE-2015-1579, 2015).

**Fixed in:** 4.1.5

**Recommendation:** Update plug-in to the latest version

**Vulnerability:** WordPress Slider Revolution Shell Upload

**Description:** Slider Revolution and Showbiz Pro fail to check authentication in revslider\_admin.php/showbiz\_admin.php allowing an unauthenticated attacker to abuse administrative features (WordPress Plugin Slider REvolution 3.0.95 / Showbiz Pro 1.7.1 - Arbitrary File Upload, 2014).

**Fixed in:** 3.0.96

**Recommendations:** Version of plug-in have not detected. Check version and update to latest if necessary.

**Vulnerability:**

sitepress-multilingual-cms - Full Path Disclosure

**Description:** Plugin full path disclosure vulnerability can be used for attack

**Fixed in:** 3.1.7.2

**Recommendations:** Version of plug-in was not detected. Check version and update to latest if necessary.

**Vulnerability:** WPML <= 3.1.7.2 - Multiple Vulnerabilities (Including SQLi)

**Description:** SQL injection vulnerability in the WPML plugin before 3.1.9 for WordPress allows remote attackers to execute arbitrary SQL commands via the lang parameter in the HTTP Referer header in a wp-link-ajax action to comments/feed (CVE-2015-2314, 2015). The "menu sync" function in the WPML plugin before 3.1.9 for WordPress allows remote attackers to delete arbitrary posts, pages, and menus via a crafted request to sitepress-multilingual-cms/menu/menus-sync.php (CVE-2015-2791, 2015). The WPML plugin before 3.1.9 for WordPress does not properly handle multiple actions in a request, which allows remote attackers to bypass nonce checks and perform arbitrary actions via a request containing an action POST parameter, an action GET parameter, and a valid nonce for the action GET parameter (Common Vulnerabilities and Exposures, 2015).

**Fixed in:** 3.1.9

**Recommendation:** Update plug-in to the latest version

**Vulnerability:** WPML 2.9.3-3.2.6 - Cross-Site Scripting (XSS) in Accept-Language Header

**Description:** Vulnerability in sitepress-multilingual-cms plugin allows to do XSS attack using Accept-language header code.

**Fixed in:** 3.2.7

**Recommendation:** Update plug-in to the latest version

### 6.3.3 Recommendations for security improvement

Update plugins to the latest version.

Attention. Plugins are creating by third-part developers, and, sometimes, plugins could be not compatible with current WordPress version after updating, start conflict with another extensions or break their functionality. Please check explanations at the plugins database on official WordPress website.

## 6.4 Test case 4 implementation. Brute Force attack

### 6.4.1 Running the test 4

Test date: 10.05.2017

**Table 12.** Test case 4 implementation. Brute Force attack.

Step	Description	Inputs	Outputs
1	WPscan	test_sergey	Unknown response
2	WPscan	admin	Unknown response
3	WPscan	... (if possible)	impossible

Brute force attack have been done using WPScan utility. For that purpose, I have downloaded a dictionary contains a lot of most common used passwords (Passwords, 2015).

Brute force test on the Figure 8 shows, that mechanism of user authentication have been modified and brute-force attack cannot be implemented using traditional methods.

```
[+] Starting the password brute forcer
Brute Forcing 'test_sergey' Time: 00:00:00 <> (0 / 14344392) 0.00% ETA: ??? Brute Forcing
'test_sergey' Time: 00:00:04 <> (1 / 14344392) 0.00% ETA: ??? [!] ERROR: We received an unknown
response for princess...
Brute Forcing 'test_sergey' Time: 00:00:04 <> (1 / 14344392) 0.00% ETA: ??? Brute Forcing
'test_sergey' Time: 00:00:04 <> (2 / 14344392) 0.00% ETA: ??? [!] ERROR: We received an unknown
response for nicole...
Brute Forcing 'test_sergey' Time: 00:00:04 <> (2 / 14344392) 0.00% ETA: ??? Brute Forcing
'test_sergey' Time: 00:00:05 <> (3 / 14344392) 0.00% ETA: ??? [!] ERROR: We received an unknown
response for 1234567...
Brute Forcing 'test_sergey' Time: 00:00:05 <> (3 / 14344392) 0.00% ETA: ??? [!] ERROR: We received
an unknown response for qwerty...
Brute Forcing 'test_sergey' Time: 00:00:05 <> (4 / 14344392) 0.00% ETA: ??? Brute Forcing
'test_sergey' Time: 00:00:05 <> (5 / 14344392) 0.00% ETA: ??? [!] ERROR: We received an unknown
response for ashley...
Brute Forcing 'test_sergey' Time: 00:00:05 <> (5 / 14344392) 0.00% ETA: ??? [!] ERROR: We received
an unknown response for rockyou...
Brute Forcing 'test_sergey' Time: 00:00:05 <> (6 / 14344392) 0.00% ETA: ??? [!] ERROR: We received
an unknown response for 123456789...
Brute Forcing 'test_sergey' Time: 00:00:05 <> (7 / 14344392) 0.00% ETA: ??? Brute Forcing
'test_sergey' Time: 00:00:05 <> (8 / 14344392) 0.00% ETA: ??? [!] ERROR: We received an unknown
response for michael...
Brute Forcing 'test_sergey' Time: 00:00:05 <> (8 / 14344392) 0.00% ETA: ??? Brute Forcing
'test_sergey' Time: 00:00:05 <> (9 / 14344392) 0.00% ETA: ??? [!] ERROR: We received an unknown
response for 123456...
Brute Forcing 'test_sergey' Time: 00:00:05 <> (10 / 14344392) 0.00% ETA: ??? Brute Forcing
```

Figure 6. Attempt of brute force attack.

### 6.4.2 Analysis of the test results 4

Test failed. I suppose, website contains solution protecting it from brute force attack, because utility returns text “We received an unknown response for...” displayed for all scanned values.



## 6.5 Test case 5. Discovering the scripts of Website.

### 6.5.1 Running the test 5

Test date: 11.05.2017

Table 13. Test case 5 implementation. Discovering scripts of Website

Step	Description	Inputs	Output
1	Check website for wp-paths in search engine	Website URL, wp-path's Statements: /wp-content/ /wp-content/plugins/ /wp-content/themes/ /uploads/ /images/	No access
2	Inspect the code of scripts	Code of scripts (Developer tools in browser used)	No SQL statements, user's credentials or paths found
3	Analysis of found information	Found information	No information for audit.

This test I've done manually, because no automation tools have been found. Step 1 contained attempts to check several URLs that were no succeed. At the 2<sup>nd</sup> step I have checked scripts from the browser development tools manually, but no any scripts or SQL-queries have been found. But I remembered, that in further tests will check the code for scripts automatically also.

### 6.5.2 Analysis of the test 5 results

No vulnerabilities found.

One strange thing have found, but I cannot find the way for attack.

I've attempted to get access to "wp-admin" folder hidden in robots.txt using URL:

[https://www.digitaluniversity.fi/wp-admin/admin-ajax.php?action=revslider\\_show\\_image&img=../wp-admin](https://www.digitaluniversity.fi/wp-admin/admin-ajax.php?action=revslider_show_image&img=../wp-admin)

Browser returns "0" on the page. Sometimes, it can be used for attack, but I can't evaluate is it dangerous or not.

### 6.5.3 Recommendations for security improvement.

Check the problem found. Evaluate the risk for security and fix issue, if it necessary.

## 6.6 Test case 6. SQL injection attack

### 6.6.1 Running the test 6

Test date: 12.05.2017

**Table 14.** Test case 6 implementation. SQL injection attack.

Step	Description	Inputs	Outputs
1	Check actions for re-quest/response can be used for injection	Website URLs with forms scripts	Search form Password recovery form User login form
2	Try to get database content via SQL injection tools	Data for injection	Failed
3	Get user's credentials	Access to database	Failed

For the 1<sup>st</sup> step I have tried to use two different tools for SQL injection attack pre-installed in Kali Linux. There are sqlmap and jSQLi applications.

First I have tried to make injection using command in terminal:

```
sqlmap -u https://www.digitaluniversity.fi/?cache-flush=1495499187.632
```

It showed that no vulnerabilities have found. I have checked also URLs for user login form and contained number of user that are showing results in browser, for example URL:

<https://www.digitaluniversity.fi/?author=100>

SQL injection have not been succeed.

Besides I've opened a jSQL injection tool using command:

```
Jsql
```

This application has GUI and I've tried to use the same URLs for it. Unfortunately, all attempts have not been succeed

### 6.6.2 Analysis of the test 6 results

No vulnerabilities found. All attempts to SQL injection have been failed that means the website is well protected from SQL injection attacks.

## 6.7 Test case 7 implementation. Cross site scripting attack test

### 6.7.1 Running the test 7

Test date: 12.05.2017

**Table 15.** Test case 7 implementation. XSS attack test

Step	Description	Inputs	Outputs
1	Inspect the code in browser, find potential elements for attack	Website URL	No elements found
2	Modify code for attack	Elements for attack	Failed
3	Run modified code	Modified code	Failed

This test is generally based on information given from Test case 5 “Attempt to access files and scripts” which collects information from scripts. Besides XSS attack scanning have been done using “uniscan” utility. It tests all files accessible for XSS vulnerabilities inside its content. I have used command:

```
uniscan -u www.digitaluniversity.fi -qd
```

which runs scanning for different kinds of vulnerabilities including XSS vulnerabilities and dynamic checks that scan all possible directories.

### 6.7.2 Analysis of the test 7 results

No vulnerabilities found.

## 6.8 Test case 8 implementation. File upload test

### 6.8.1 Running the test 7

Test date: 12.05.2017

**Table 16.** Test case 8 implementation. File upload test.

Step	Description	Inputs	Outputs
1	Inspect the website, find file upload components	Website URL	No possibility found
2	Try to upload simple script	Upload component	Failed
3	Try to upload malicious script	Upload component	Failed
4	Try to run script from browser	Upload URL	Failed

Website have been inspected manually and automatically for uploading forms availability, but nothing have been found.

### 6.8.2 Analysis of the test 8 results

No vulnerabilities found

## 6.9 Common test's results and security audit outcomes.

Table 17. Common security tests results.

<b>Test case</b>	<b>Result (passed /failed)</b>	<b>Severity if failed (critical / normal / low)</b>	<b>Notes (test case step that failed, deviation from expected outputs and other relevant information)</b>
1	passed		The list of users is not full. Recover password behaviour allows to check if the user exists. All opened ports are detected. It is possible to check if the user exists using password recovery form
2	passed		Wp-... folders are not accessible
3	passed	Critical	Vulnerabilities found. Site needs to be updated
4	failed	Normal	Errors during test caused by unusual security settings
5	failed	Normal	Scripts or code are using credentials or queries are not found
6	failed	Normal	Website has a high level of security against SQL-injection attack
7	failed	Normal	No possibilities to implement XSS attack were found
8	failed	Normal	File upload attack is impossible, upload component unavailable

## **6.10 Security audit 1 outcomes.**

Generally, website digitaluniversity.fi has a good level of protection. Core folders were hidden for access. One critical vulnerability founded. Updating WordPress to the latest version and update of several plug-ins needed. The number of vulnerabilities detected shows the importance of regular updates for all components of the website, because most of vulnerabilities became known at the last three months.

It is necessary to modify password protection form that should not to show any signs tells about username existence in database. It is important to reduce ability of username guessing that can be used for attacks.

All recommendations have been sent to the website owner. Second set of testing cases needs to run for checking if all vulnerabilities have destroyed after update.

## 7 Security audit report 2

Website owner informed me that updates have been installed and I can test it again to be sure all vulnerabilities found in previous testing have destroyed. I have decided to run second set of tests are checking previously detected vulnerabilities only. It will contain checking WordPress platform, its extensions, usernames are available to view, checking logics of login and recovery password forms. I have collected all necessary actions into the one test.

### 7.1 Test case 1 implementation. Gathering information.

#### 7.1.1 Running the test 2.1

Test date: 17.05.2017

Table 18. Test case 2.1 implementation. Gathering Information.

Step	Description	Inputs	Outputs
1	Version of Wordpress	Website URL: www.digitaluniversity.fi	4.7.5 (have been updated)
2	Theme(s) vulnerabilities detection	Website URL: www.digitaluniversity.fi	No vulnerabilities detected
3	Plug-in(s) vulnerabilities detection	Website URL: www.digitaluniversity.fi	Plugin's vulnerabilities detected
4	List of users	Website URL: www.digitaluniversity.fi	5 usernames detected
5	User password recovery page behaviour	Website URL: www.digitaluniversity.fi	Username existence can be detected

Steps 1-4 from the Table 18 are using WPScan utility.

First, WPScan utility needs to update its database, because, probably, new vulnerabilities have been found and have added to it since the last using. I am using command `wpscan --update` for updating the vulnerabilities database.

Next command runs scanning of website for platform and its extensions vulnerabilities:

```
wpscan --url www.digitaluniversity.fi
```

Results contain information for the steps 1-3 of the current test.

Step 4 purpose is attempt to get a list of usernames of the website. For that purpose, I have used command:

```
wpscan --url www.digitaluniversity.fi --enumerate u
```

Test showed that WordPress version is 4.7.5 that means it have been updated to the latest version, which is actual for the test date. One vulnerability of platform and several issues in plugins have detected.

### 7.1.2 Analysis of the test 2.1 results

There is a one vulnerability detected in WordPress platform. This is the same vulnerability have been presented in previous security audit report without any solution for its fixing:

**Vulnerability:** WordPress 2.3-4.7.5 - Host Header Injection in Password Reset

**Description:** The vulnerability stems from WordPress using untrusted data by default when creating a password reset e-mail that is supposed to be delivered only to the e-mail associated with the owner's account. This could possibly allow the attacker to intercept the email containing the password reset link in some cases requiring user interaction as well as without user interaction (WordPress-Exploit-4-7-Unauth-Password-Reset-0day-CVE-2017-8295, 2017).

**Attention:** Vulnerability is not fixed at moment

**Recommendation:** Change logic of password recovery form to exclude possibility of user existence check and hide users detected in user enumeration part of information gathering test to exclude possibility of using existing usernames.

There is a list of detected vulnerabilities in plugins:

**Vulnerability:** WordPress Slider Revolution Local File Disclosure

**Description:** Directory traversal vulnerability in the Elegant Themes Divi theme for WordPress allows remote attackers to read arbitrary files via a .. (dot dot) in the img parameter in a revslider\_show\_image action to wp-admin/admin-ajax.php. NOTE: this vulnerability may be a duplicate of CVE-2014-9734 (CVE-2015-1579, 2015).

**Fixed in:** 4.1.5

**Recommendation:** Update plug-in to the latest version

**Vulnerability:** WordPress Slider Revolution Shell Upload

**Description:** Slider Revolution and Showbiz Pro fail to check authentication in revslider\_admin.php/showbiz\_admin.php allowing an unauthenticated attacker to abuse administrative features (WordPress Plugin Slider REvolution 3.0.95 / Showbiz Pro 1.7.1 - Arbitrary File Upload, 2014).

**Fixed in:** 3.0.96

**Recommendations:** Version of plug-in have not detected. Check version and update to latest if necessary.

**Vulnerability:** sitepress-multilingual-cms - Full Path Disclosure

**Description:** Plugin full pass disclosure vulnerability can be used for attack

**Fixed in:** 3.1.7.2

**Recommendations:** Version of plug-in was not detected. Check version and update to latest if necessary.

**Vulnerability:** WPML <= 3.1.7.2 - Multiple Vulnerabilities (Including SQLi)

**Description:** SQL injection vulnerability in the WPML plugin before 3.1.9 for WordPress allows remote attackers to execute arbitrary SQL commands via the lang parameter in the HTTP Referer header in a wp-link-ajax action to comments/feed (CVE-2015-2314, 2015). The "menu sync" function in the WPML plugin before 3.1.9 for WordPress allows remote attackers to delete arbitrary posts, pages, and menus via a crafted request to sitepress-multilingual-cms/menu/menus-sync.php (CVE-2015-2791, 2015). The WPML plugin before 3.1.9 for WordPress does not properly handle multiple actions in a request, which allows remote attackers to bypass nonce checks and perform arbitrary actions via a request containing an action POST parameter, an action GET parameter, and a valid nonce for the action GET parameter (Common Vulnerabilities and Exposures, 2015).

**Fixed in:** 3.1.9

**Recommendation:** Update plug-in to the latest version

**Vulnerability:** WPML 2.9.3-3.2.6 - Cross-Site Scripting (XSS) in Accept-Language Header

**Description:** Vulnerability in sitepress-multilingual-cms plugin allows to do XSS attack using Accept-language header code.

**Fixed in:** 3.2.7

**Recommendation:** Update plug-in to the latest version

I have checked password recovery form again. It has similar behavior like in test case 1 from security audit report 1.



### **7.1.3 Recommendations for security improvement.**

Several extensions have versions that are out of date and recommended to be updated, but it is not critical for security protection, because no vulnerabilities detected in current versions.

Enumerating usernames test showed that 5 usernames were identified. Maybe it is not critical to show their names, but it is necessary to check their roles in user hierarchy and to hide usernames, if it is needed.

The logics of password recovery should be changed to not show any signs tell to the user if filled username exists or not. It will allow to prevent attempts to guess usernames.

## 8 Conclusion

Amount of WordPress websites is growing fast. Simplicity of website creation and powerful extensions attracts many people but distracts from importance of security and administration of it. Many of enthusiasts are working on vulnerabilities search and platform improvement every time, but it but that is not the reason to calm down and do nothing. Any WordPress based website needs regular updates and security audit.

During the work on this report, I found many different ways to identifying security issues. My set of test cases have designed from scratch. It is useful solution that can be used by anyone who needs to check WordPress based website for security issues. Test results showed that my method is effective and helps to identify known vulnerabilities and get information how to fix it. When I wrote this thesis I has not experience in software security audit or web application testing, but I have achieved all required goals and got expected results. I suppose, If I were more experienced in web application security, I could find more threats and vulnerabilities, but considering global experience used by WordPress users, my solution covers most possible security issues.

I would like to recommend three important actions allow avoiding most known vulnerabilities and security issues for all WordPress users: First, there are regular updates of WordPress platform and its extensions. Second, it is necessary to use common recommendations for user's credentials – close usernames from others and use strong passwords, disallow access to critical files and folders. Finally, I recommend doing regular security audit. The frequency of security audit can differ depends from many factors like website content, its purpose, value from the business point and so on, but recommended period of security audit is 2-4 months.

I hope my thesis work will be useful for people who needs to do security audit. Tests presented in my report can be used for any WordPress website. Moreover, there are several plugins could be installed for permanent monitoring of website security.

Many sources have been used during the work on thesis. Most of it are security-oriented websites supported by official WordPress development team or being its partner.

Thesis advisor, Olavi Korhonen has supported my work. We regularly discussed current status of the project on the meetings or by e-mail. Advisor's suggestions and comments were useful to achieve better results.

My further research could discover testing of known solutions for regular automatic monitoring of security issues of WordPress-based websites, which required to protect most of websites in the World to possible attacks.

## References

- Content management system.* (2017). Retrieved March 27, 2017, from Wikipedia:  
[https://en.wikipedia.org/wiki/Content\\_management\\_system](https://en.wikipedia.org/wiki/Content_management_system)
- G.R., A. (2010, January 22). *WordPress Foundation Launches to Protect Open Source Projects.* Retrieved May 25, 2017, from The Blog Herald:  
<http://www.blogherald.com/2010/01/22/wordpress-foundation-launches-to-protect-open-source-projects/>
- How to choose the right CMS for your Website.* (2015, April 11). Retrieved May 17, 2017, from Access Desires: <http://www.accessdesires.com/blog/how-to-choose-the-right-cms-for-your-website/>
- Introduction to SQL.* (2007). Retrieved May 04, 2017, from Function X:  
<http://www.functionx.com/sql/Lesson01.htm>
- Introduction to the DOM.* (2017, April 18). Retrieved May 09, 2017, from Mozilla Developer network: [https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model/Introduction](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction)
- Kimi. (2016, November 16). *Why is WordPress the Most Popular CMS for Web Design.* Retrieved May 17, 2017, from Web Marketing Studio:  
<http://webandmarketingstudio.com/wordpress-popular-cms-web-design/>
- McKeown, S. (2012, April 12). *What's Going On In The WordPress Economy?* Retrieved May 25, 2017, from Smashing Magazine:  
<https://www.smashingmagazine.com/2012/04/wordpress-economy-part-1/>
- McKeown, S. (2017). On Forking Wordpress, Forks in General, Early Wordpress and the Community. In S. McKeown, *WordPress. Freedom, COmmunity and the Business of Open Source.* Retrieved May 22, 2017, from Wordpress:  
<https://wordpress.org/about/history/chapter3.pdf>
- Nmap.* (2017). Retrieved May 17, 2017, from Nmap.org: <https://nmap.org/>
- Passwords.* (2015, May 18). Retrieved May 02, 2017, from Skull Security:  
<https://wiki.skullsecurity.org/Passwords>
- Plugins.* (2017). Retrieved April 04, 2017, from Wordpress: <https://wordpress.org/plugins/>
- Project page.* (2001). Retrieved May 20, 2017, from b2 cafelog: <http://cafelog.com/>
- Search engine indexing.* (2017, May 12). Retrieved May 18, 2017, from Wikipedia:  
[https://en.wikipedia.org/wiki/Search\\_engine\\_indexing](https://en.wikipedia.org/wiki/Search_engine_indexing)

*SQL injection*. (2017, April). Retrieved April 27, 2017, from Wikipedia:  
[https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)

*SQL Injection Bypassing WAF*. (2017, 01 20). Retrieved 04 11, 2017, from OWASP:  
[https://www.owasp.org/index.php/SQL\\_Injection\\_Bypassing\\_WAF](https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF)

*The History of WordPress*. (2011, October 28). Retrieved May 2017, 24, from SEO-Alien:  
<http://www.seo-alien.com/articles/history-wordpress/>

*The History of WordPress*. (2012, October 14). Retrieved May 25, 2017, from Lorelle teachers:  
<https://lorelleteaches.com/2012/10/14/the-history-of-wordpress/>

The WordPress Security Learning Center. (2017, January 04). *How to Prevent Cross Site Scripting Attacks*. Retrieved April 28, 2017, from Wordfence:  
<https://www.wordfence.com/learn/how-to-prevent-cross-site-scripting-attacks/>

The WordPress Security Learning Center. (2017, January 04). *Understanding SQL Injection Attacks*. Retrieved April 07, 2017, from Wordfence:  
<https://www.wordfence.com/learn/how-to-prevent-sql-injection-attacks/>

*Total number of Websites*. (2016). Retrieved March 27, 2017, from Internet live stats:  
<http://www.internetlivestats.com/total-number-of-websites/>

*Tour of the WordPress Database*. (2016, April 18). Retrieved March 28, 2017, from Delicious Brains:  
[https://deliciousbrains.com/tour-wordpress-database/#wp\\_terms](https://deliciousbrains.com/tour-wordpress-database/#wp_terms)

*Usage of content management systems for websites*. (n.d.). Retrieved March 29, 2017, from W3Techs:  
[https://w3techs.com/technologies/overview/content\\_management/all](https://w3techs.com/technologies/overview/content_management/all)

*Usage of server-side programming languages for websites*. (n.d.). Retrieved May 04, 2017, from W3Techs:  
[https://w3techs.com/technologies/overview/programming\\_language/all](https://w3techs.com/technologies/overview/programming_language/all)

*What is Kali Linux?* (n.d.). Retrieved May 19, 2017, from Kali Linux official documentation:  
<http://docs.kali.org/introduction/what-is-kali-linux>

WordPress News. (2013, December 31). *History of WordPress: The Good, The Bad & The Ugly*. Retrieved May 25, 2017, from WPExplorer:  
<http://www.wpexplorer.com/history-wordpress/>

*WordPress Used On 25 Percent Of All Websites*. (2015, November 09). Retrieved May 24, 2017, from Martech: <https://martechtoday.com/wordpress-used-on-25-percent-of-all-websites-report-151115>

*WPScan*. (2017). Retrieved 05 2017, 2017, from WPScan: <https://wpscan.org/>

*WPScan*. (2017, May 25). Retrieved from About WPScan: <https://wpvulndb.com/about>

*WPScan Vulnerability Database*. (2017, 03 07). Retrieved 04 04, 2017, from WPScan Vulnerability Database: [www.wpvulndb.com](http://www.wpvulndb.com)

## Appendix A. Description of database tables of the WordPress.

1. **wp\_term\_relationships** – links posts (pages) with taxonomies and their order:  
object\_id – the ID of the post object (linked to wp\_posts table)  
term\_taxonomy\_id – the ID of the term (linked to wp\_term\_taxonomy table)  
term\_order – ability to use order of terms for an object
2. **wp\_term\_taxonomy** – contains taxonomies with their properties:  
term\_taxonomy\_id – unique number of taxonomy  
term\_id - the ID of related to taxonomy term  
taxonomy – the slug of taxonomy  
description – description of the taxonomy  
parent – ID of a parent item. Used for hierarchical taxonomies  
count – number of posts objects which assigned to the taxonomy
3. **wp\_termmeta** – table contains metadata for taxonomies. Attributes:  
meta\_id – unique number for each row of the table  
term\_id - ID of related term (links to wp\_terms table)  
meta\_key – identifying key for the piece of data  
meta\_value – actual piece of data
4. **wp\_terms** – contains terms – items of taxonomy and their properties:  
id – unique number of term  
name – the name of term  
slug - URL friendly name of term  
term\_group – property allows to group terms together (usually using by plugins and themes)
5. **wp\_options** – contains website settings and configuration data. This table used for themes, plugins and widgets and temporary cached data.  
option\_id – unique number of option  
option\_name – the name for an option (piece of data for configuration)  
option\_value – value of option  
autoload – automatical loading control for option

6. **wp\_users** – contains user records required for user management:
  - id – unique number for the user
  - user\_login – unique username for the user
  - user\_pass – user’s password transformed to hash for security reasons
  - user\_nicename – name for the user showing on website
  - user\_email – e-mail address of the user
  - user\_url – website address of the user
  - user\_registered – date and time user registered
  - user\_activation\_key – activation key using for password reset
  - user\_status – old value used in versions of WordPress earlier 3.0 to indicate spam users.
  - display\_name – name to be used publicly in the website
  
7. **wp\_usermeta** – contains information related to users and their individual settings (like design settings, profile settings etc.), which is accessible from dashboard.
  - umeta\_id – unique number for each row of the table
  - user\_id – ID of related user (links to wp\_users table)
  - meta\_key – identifying key for the piece of data
  - meta\_value – actual piece of data
  
8. **wp\_links** – special table responsible for external links data storage contains attributes:
  - link\_id – unique number for each link
  - link\_url – URL of the link
  - link\_name – name of the link
  - link\_image – image URL related to the link
  - link\_target – target frame for the link (\_blank, \_top, \_none)
  - link\_description – description of the link
  - link\_visible – control visibility of the link (public/private)
  - link\_owner – ID of user who created the link (links to the wp\_users table)
  - link\_rating – rating for the link (from 1 to 10)
  - link\_updated – date and time link has been updated
  - link\_rel – relationship of the link
  - link\_notes – notes related to the link
  - link\_rss – RSS address for the link

9. **wp\_posts** – most important table in WordPress database. It contains data related to website content – posts, pages, menu items etc.

id – unique number of post

post\_author – ID of user created the post

post\_date – date and time of post creation

post\_date\_gmt – GMT time of post creation (necessary for flexible time zone management)

post\_content – the content for post (HTML, code and other content)

post\_title – title of the post

post\_excerpt – intro, a short version of post content

post\_status – status of the post (draft, private, publish...)

comment\_status – control comments allowance

ping\_status – control ping and trackbacks allowance

post\_password – password to view the post (optional)

post\_name – URL friendly slug for the post title

to\_ping – a list of URLs should send pingbacks to when updated

pinged – a list of URLs has sent pingbacks to when updated

post\_modified – time and date of last modification of the post

post\_modified\_gmt – GMT time and date of last modification of the post

post\_content\_filtered – used by plugins to cache version of post passed through the “the content” filter.

post\_parent – the parent post which current post is related. This parameter is used to create hierarchical relationship between posts

guid – Global Unique Identifier, permanent URL to the post, not the permalink version

menu\_order – used by sorting in special order

post\_type – the content type identifier

post\_mime\_type – MIME type of the attachment (uploaded file)

comment\_count – counter of the total number of comments related to the post

10. **wp\_postmeta** – contains extra information about individual posts not included to wp\_posts table. Used mainly by plugins and themes.

meta\_id – unique number of each row of the table

post\_id – the ID of the post related to (links to wp\_posts table)

meta\_key – identifying key of the piece of data

meta\_value – the actual piece of data.



11. **wp\_comments** – table responsible for storing comments data. Attributes:

- comment\_ID – unique number of the comment
- comment\_post\_ID – ID of the post related to the comment (links to wp\_posts table)
- comment\_author – name of the comment author
- comment\_author\_email – e-mail of the comment author
- comment\_author\_url – URL of the comment author website
- comment\_author\_IP – IP Address of the comment author
- comment\_date – date and time comment was posted
- comment\_date\_gmt – GMT date and time comment was posted
- comment\_content – the actual text of the comment
- comment\_karma – attribute used by plugins for comments management purpose
- comment\_approved – control comment approval
- comment\_agent – data contains technical information for posted comment (operation system, browser etc.)
- comment\_type – type of comment (comment/pingback/trackback)
- comment\_parent – parent comment which current comment is related to (used for hierarchy for replies)
- user\_id – ID of comment author if the author is registered user (links to wp\_users table)

12. **wp\_commentmeta** – table stores additional comment's data. Attributes:

- meta\_id – unique number assigned for each row
- comment\_id – the ID of the related comment (links to wp\_comments table)
- meta\_key – an identifying key for the piece of data
- meta\_value – the actual piece of data

## Appendix B. Description of core files and folders.

### Folders:

**wp-admin** – folder, contains files required to link database with current installation, tools for dashboard and key-functions related to user management and administration.

**wp-content** – stores themes and plugins are using in website. This folder doesn't overwrite when WordPress updates to a new version.

**wp-includes** – folder contains all files needed for website functionality but didn't included to wp\_admin and wp\_content folders. There are a lot of files of WordPress core, certificates, widgets, javascript files.

**wc-logs** – folder where logs can be stored

### Files:

**index.php** – the file loads and initializes all WordPress files when it requested. Usually it has requested automatically when website is requested.

**.htaccess** – the file of server configuration manages permalinks, redirects and access restrictions. It is not directly related to WordPress, but it needs to be stored inside root folder of website.

**wp-config.php** – configuration file of WordPress manages database connection and global settings for website. This file is very important from the point of security, because it contains important data required for access to database.

**robots.txt** – file contains instructions for search engines. This file is not a part of WordPress, but the policy of interaction with search engines needs to place it inside the root folder of the website.

**functions.php** – file stored in "wp-includes" folder is responsible for behavior, features and functionality of pages of the website.

**admin.php** - file stored in "wp-admin" folder contains code is showing authorisation page for website

**wp-load.php** – file stored in "wp-admin" folder contains script runs after verification which runs wp-config.php after executing

**wp-config.php** file stored in root folder of website responsible for general management of database and website. Contains data needed for authentication and connection to MySQL database.