AN AUTOMATED VULNERABILITY DETECTION FOR WEBSITE SECURITY

Primož Cigoj

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

Supervisor: Prof. Borka Jerman Blažič, Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia

Evaluation Board:
Assist. Prof. Tomaž Klobučar, Chair, Jožef Stefan Institute, Ljubljana, Slovenia
Prof. Tomaž Turk, Member, School of Economics and Business, University of Ljubljana, Ljubljana, Slovenia
Prof. Oliver Popov, Member, Department of Computer and Systems Sciences, Stockholm University, Kista, Sweden

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Primož Cigoj

AN AUTOMATED VULNERABILITY DETECTION FOR WEBSITE SECURITY

Doctoral Dissertation

AVTOMATSKO ZAZNAVANJE RANLJIVOSTI ZA VARNOST SPLETNIH STRANI

Doktorska disertacija

Supervisor: Prof. Borka Jerman Blažič

Ljubljana, Slovenia, September 2021

To my parents
Mojim staršem

Acknowledgments

I would like to thank my supervisor Borka Jerman Blažič for giving me the opportunity to work at the Laboratory for Open Systems and Networks at the Jožef Stefan Institute, for providing support and general directions for my research, and for her meticulous corrections of my papers.

I would like to thank Tomaž Klobučar for also giving me the opportunity to work at the Laboratory for Open Systems and Networks at the Jožef Stefan Institute, for general lessons, for all the time he spent on me, and for other opportunities he gave me.

I would like to thank the master of statistics Živa Stepančič for her response times and for various thoughts and lessons.

Finally, I would like to thank the evaluation board – Tomaž Klobučar, Tomaž Turk and Oliver Popov – for their time and the corrections that improved my thesis.

Thanks to all of you.

Abstract

Although large research efforts on web application security have been invested for more than a decade, the security of web applications is still a challenging problem. The main focus of the cybersecurity community has been to make operating systems and communication networks more secure and harder for attackers to penetrate. This also applies to all applications on the Internet. However, there is a big difference among them. The most frequently used web applications and user websites today are developed with the Web Content Management System (WCMS), because it allows user-friendly access, and easy development and operation. WCMSs are present all over the world in many different environments. Malware that can penetrate the WCMS can significantly affect the system itself and can cause a misconfiguration or other serious damage of the service offered. The security and stability of these systems are important for reducing the risks and consequences of attacks or malfunctioning of websites caused by malware or other methods used by intruders.

This thesis presents a newly developed method for identifying vulnerable Internet websites built using WCMS applications. The research study carried out within this thesis was focused on investigating and developing methods to collect metadata about vulnerable web applications across the Internet. The main desired properties pursued in the research were an acceptable speed of collecting vulnerable WCMS data on a large scale and applying an ethical vulnerability search method. The key feature of the developed method is the ability to perform automated, fast, and dynamic vulnerability scans of the websites built with WCMS and the attached plug-ins on a large scale.

This thesis explores the state of WP website security based on identified vulnerabilities on the global scale, with focus on the web space of 30 European countries. The study is based on a study of the web space where the websites are built with WordPress Content Management Systems (WPCMS), which is the most popular WCMS on the Internet. WPCMS websites represent a part of the websites in the whole world web space. The main part of the thesis presents a newly developed methodology for provision of information about the vulnerability of the WPCMS and the attached website plug-ins. The methodology is implemented as a newly developed tool that was used for large-scale scanning of the Internet. The collected data are applied in a system that computes the security score of websites. The collected data are then analysed in relation to several parameters that were identified as impacting the overall level of web space security in the countries studied.

In the first part of the thesis, a background of the studied field of application is given, which provides an overview of the previous research studies and compares the current vulnerability scanning methods. The overview provides a clear understanding about the applicability of the presented methods and their benefits and drawbacks. Based on these findings, the development of the new method is presented together with the main feature applied to enable fast scanning, data gathering and security scoring. Each property of the tool is compared with the existing similar tool properties. Advantages of the tool are presented as well.

The purpose of the study was to show whether websites built with WPCMS differ in the percentage of secure websites in different application sectors. No significant difference in the percentage of insecure websites belonging to different application sectors was found, with the exception of a lower average percentage of insecure websites found in the news sector. Also, the hosting of several websites on the same server does not appear to impact the occurrence of a higher insecurity. The percentage of insecure core versions of websites built with WPCMS in two time periods of large-scale scanning have shown that the percentage of detected insecure websites is lower. However, the presence of plug-ins remained the same and it was found that plug-ins are a positive risk factor for insecurity.

The impact of two parameters on the appearance of the higher level of security are presented by the end of the thesis. A research study within the European web space has shown that the level of the Digital Skills (DS) of the country and, indirectly, the low cost of fixed access to Internet normalized by the country's Gross National Income (GNI) are correlated with the percentage of web security and consequently the security of websites built with WPCMS is also higher.

Povzetek

Varnost spletnih aplikacij je še vedno izziv, čeprav že več kot desetletje potekajo obsežni raziskovalni napori na tem področju. Skupnost za kibernetsko varnost si prizadeva operacijske sisteme in komunikacijska omrežja narediti bolj varne, napadalcem pa otežiti napade. K enakemu cilju stremi tudi, ko gre za spletne aplikacije. Vendar je treba opozoriti tudi na veliko razliko med njimi.

Najbolj priljubljene spletne aplikacije in uporabniška spletna mesta razvijajo v sistemu za upravljanje spletnih vsebin (WCMS), saj uporabniku omogočajo prijazen dostop, enostaven razvoj in upravljanje. WCMS je razširjen po vsem svetu. Če zlonamerna programska oprema prodre vanj, lahko pomembno vpliva na sistem in povzroči napačno konfiguracijo ali drugače resno poškoduje ponujene storitve. Zato sta varnost in stabilnost teh sistemov ključni za zmanjšanje tveganj in posledic napadov ali motenj v delovanju spletnih mest zaradi zlonamerne programske opreme ali drugih vsiljivcev.

Doktorska disertacija predstavlja novo razvito metodo za prepoznavanje ranljivih spletnih mest, zgrajenih v WCMS. Študija, ki je potekala med nastajanjem doktorske disertacije, je bila osredotočena na raziskovanje in razvoj metod za zbiranje metapodatkov o ranljivih spletnih aplikacijah. Osrednja namena raziskave sta bila doseči sprejemljivo hitrost zbiranja ranljivih podatkov WCMS v velikem obsegu in pri iskanju ranljivosti slediti etičnim načelom. Ključna značilnost razvite metode je zmožnost obsežnega izvajanja avtomatiziranih, hitrih in dinamičnih pregledov ranljivosti spletnih mest, zgrajenih v WCMS, in priloženih vtičnikov.

Doktorska disertacija raziskuje, kakšna je po odkritih ranljivostih dejanska varnost spletne strani na platformi WordPress (WP). Raziskava temelji na študiji spletnega prostora po vsem svetu, s poudarkom na 30 evropskih državah, kjer so spletna mesta zgrajena z WordPress Content Management Systems (WPCMS), najbolj priljubljenem WCMS.

Spletne strani WPCMS predstavljajo del spletnih mest v celotnem svetovnem spletnem prostoru. Osrednji namen doktorske disertacije je predstaviti novo metodologijo za zagotavljanje informacij o ranljivosti WPCMS in usklajenih vtičnikih za spletno stran. Metodologija je bila kot orodje uporabljena za obsežno skeniranje interneta. Zbrani podatki se uporabljajo za izračun varnostne ocene spletnih mest. Podatki se nato analizirajo glede na več parametrov, za katere je bilo ugotovljeno, da vplivajo na splošno raven varnosti spletnega prostora v preučevanih državah.

V doktorski disertaciji je v prvem delu podano ozadje področja uporabe spletnih aplikacij in WCMS. Pregledno so predstavljene dosedanje raziskovalne študije in primerjave trenutnih metod skeniranja ranljivosti. Pregled omogoča jasno razumevanje uporabnosti predstavljenih metod, njihovih prednosti in slabosti. Na podlagi teh ugotovitev je predstavljen razvoj nove metode in njegova glavna izboljšava, ki omogoča hitro skeniranje, zbiranje podatkov in varnostno točkovanje. Vsako lastnost orodja primerjamo z obstoječimi podobnimi lastnostmi orodij. Predstavljene so tudi prednosti novega orodja.

Namen študije je bil ugotoviti, ali se spletna mesta, zgrajena z WPCMS, razlikujejo glede na delež varnih spletnih mest v različnih sektorjih uporabe. V odstotku ranljivih

spletnih mest, ki pripadajo različnim sektorjem, ni bilo ugotovljene pomembne razlike, z izjemo spletnih mest v sektorju novic. Prav tako je videti, da gostovanje več spletnih mest na istem strežniku ne povzroča večje ranljivosti. Delež ranljivih osnovnih različic spletnih mest, zgrajenih z WPCMS, je po dveh skeniranjih v večjem obsegu pokazal, da je delež najdenih nezaščitenih spletnih mest nižji. Kljub temu je prisotnost vtičnikov ostala enaka. Ugotovljeno je bilo tudi, da so vtičniki pozitiven dejavnik tveganja za ranljivost.

Raziskovalna študija evropskega spletnega prostora je pokazala, da je varnost spletnih mest, zgrajenih v WPCMS, boljša tam, kjer so višji tudi družbeni in digitalni standardi (DS), kjer so torej nižji stroški fiksnega dostopa do interneta in višja raven digitalnih spretnosti.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

ACK ... Acknowledgement
AFRINIC ... African Network Information Centre
AIC ... Akaike Information Criterion
API ... Application Programming Interface
APNIC ... Asia Pacific Network Information Centre
ARIN ... American Registry for Internet Numbers
AS ... Autonomous System
ASN ... Autonomous System Number
BIC ... Bayesian information criterion
ccTLD ... Country Code Top-Level Domain
CERN ... European Organization for Nuclear Research
CERT ... Computer Emergency Response Team
CLI ... Command-line Interface
CNAME ... Canonical Name record
CPE ... Common Platform Enumeration
CPU ... Central Processing Unit
CSRF ... Cross-site Request Forgery
CSS ... Cascading Style Sheets
CVE ... Common Vulnerabilities and Exposures
CVSS ... Common Vulnerability Assessment
DB ... Database
DDoS ... Distributed Denial-of-Service
DNS ... Domain Name System
DoS ... Denial-of-Service
DS ... Digital Skills
FoIU ... Frequency of Internet Use
FTP ... File Transfer Protocol
GDPR ... General Data Protection Regulation
GIL ... Global Interpreter Lock
GNI ... Gross National Income
GPL ... General Public License
GUI ... Graphical User Interface
gTLD ... Generic Top-level Domain
HH ... Household
HTML ... Hyper Text Mark-up Language
HTTP ... Hypertext Transfer Protocol
ICS ... Industrial control system
ICSSS ... International Cyber Security Summer School
ICT ... Information and Communication Technology
IDS ... Intrusion Detection System
IIS ... Internet Information Service

IoT	... Internet of Things
IP	... Internet Protocol
IPv4	... Internet Protocol v4
IPv4	... Internet Protocol v6
ITU	... International Telecommunication Union
IU	... Internet Use
JS	... JavaScript
JSON	... JavaScript Object Notation
LACNIC	... Latin America and Caribbean Network Information Centre
LDAP	... Lightweight Directory Access Protocol
MFA	... Multi-factor Authentication
MySQL	... My Structured Query Language
MS SQL	... Microsoft Structured Query Language
NIST	... National Institute of Standards and Technology
NoSQL	... Not only SQL
NSO	... National Statistical Offices
NVD	... National Vulnerability Database
OECD	... Organisation for Economic Co-operation and Development
OS	... Operating System
OSS	... Open-Source Software
OWASP	... Open Web Application Security Project
P2P	... Peer-to-Peer
PHP	... Hypertext Preprocessor
RFC	... Request for Comments
RIPE	... Réseaux IP Européens
RIR	... Regional Internet Registry
RMSE	... Root Mean Square Error
RST	... Reset
RWHOIS	... WHOIS Referral
SCAP	... Security Content Automation Protocol
SEO	... Search Engine Optimization
SMS	... Short Message Service
SOA	... Start of Authority
SQL	... Structured Query Language
SSH	... Secure Shell
SSL	... Secure Sockets Layer
SYN	... Synchronize
TCP	... Transmission Control Protocol
TLD	... Top-Level Domain
TLS	... Transport Layer Security
URI	... Uniform Resource Identifier
URL	... Uniform Resource Locator
W3C	... World Wide Web Consortium
WCMS	... Web Content Management System
WP	... WordPress
WPCMS	... WordPress Content Management System
WWW	... World Wide Web
XSS	... Cross-site Scripting
XXE	... XML External Entities
ZAP	Zed Attack Proxy

Chapter 1

Introduction

The World Wide Web (WWW) has changed dramatically over the years. Tim Berners-Lee invented the World Wide Web in 1989 while working at European Organization for Nuclear Research (CERN), as he needed a way for researchers in different parts of the world to collaborate and share information between them (Gloor, 2006). In the initial period of the web development, web application developers developed very simple static websites that contained only a few images and text. As users demanded more interactive web applications, there was a need for new technologies and concepts for easier development of the desired applications. However, within the advancement of the WWW technology, vulnerabilities that open the door to cybercrime have arisen and a need for tools that detect the vulnerabilities appeared.

A common methodology for finding web vulnerabilities is scanning the websites with a web scanner. There are different types of web vulnerability scanner models, in which a large number of features has some undesirable consequences. Scanning tools that support large-scale scans for inspecting security features of web server ports are one type of the web vulnerability scanners; however, they do not conduct a deep review of the content within the websites. The port information collected using these tools does not include information about the vulnerability of the website content and associated plug-ins as the tools operate with the Internet Protocol (IP) address of the web server and do not crawl links within the website's content.

Another type of tools are security scanning tools that support detailed scans. These tools are used to perform a vulnerability identification and other scanning activities, but the results are not always shared with website owners. In addition to these tools, several researchers have created scanners that provide information whether the website is vulnerable (Goethem, Chen, Nikiforkais, Desmet, & Joosen, 2014), (Stock, Pellegrino, Li, Backes, & Rossow, 2018), (Vasek, Wadleigh, & Moore, 2015), (Schagen, Koning, Bos, & Giuffrida, 2018). The major drawbacks of most of these known scanners are that they require a lot of time and the type of information found as they use simulated attack types on the web server to discover the vulnerability, which is not welcomed by web owners. Nowadays, the use of exploit techniques is considered illegal if a web browsing permission is not granted by web owners.

Large-scale vulnerability testing provides information about the overall security of the web space in a particular region or country. An analysis of different factors for measuring the digital inequality usually addressed as "digital divide" compared with the identified level of vulnerability provides information about how regions or countries differ in their digital security assurance, how the political and regulatory efforts in a particular country affect the security of its digital market and how the differences in the level of the digital development impact the overall web space security. From this perspective, the detected

inequality among different regions or countries can trigger actions for improving the awareness about the risks and exploits to which internet users are exposed every day.

This thesis explores the web space of the Internet and the level of the web-space security for 30 European countries in terms of their level of digital development. To do so, a new, innovative scanning platform was designed and implemented for exploring the most frequently applied Web Content Management System (WCMS) known as WordPress (WP). WordPress's popularity lies in the ease of setting up the website, its low cost, the friendless of its use and its easy maintenance. The abundance of plug-ins for developing almost any different type of service and scenario, such as blogs, social network applications, banking, e-commerce, and educational services is an additional reason of its popularity. Currently, the estimated number of plug-ins that can be applied in WP websites is close to 56,000 and the total number of downloads of these plug-ins is close to 900 million. Web applications handle very sensitive information, ranging from banking to health directories, as well as personal images and photographs that are of interest to criminals and attackers. Even applications that do not handle sensitive or valuable data have become interesting targets for criminals, as they are used for various unlawful purposes, such as serving malicious content or exploiting knowledge for nefarious purposes, motivated by economic, political or religious reasons. According to a survey carried out by W3Tech Consortium, about 52.9% of Internet websites use some kind of WCMS (W3Tech, 2019). The most popular open-source WCMSs are the already mentioned WordPress, Drupal and Joomla (Hassan, Sarker, Biswas, & Sharif, 2017). WCMSs allow users, even without any in-depth knowledge of web technology, to deploy and offer web system content and services to users. Due to the popularity of these systems, they have become an interesting target for malicious attackers, and this has triggered additional importance of security features for web site protection and the identification of the vulnerabilities. This especially holds true in cases where the web content owners do not possess the necessary knowledge and understanding of the possible threats to the system and the availability of data protection systems. Any malware that can penetrate a website can significantly affect the web system itself and can cause a misconfiguration and data breaches. The security and stability of these systems are important for reducing the risks and consequences of attacks or disfunctions. The knowledge about website vulnerabilities enables the owners to remove the presence of bugs either by themselves or by replacing the compromised website core and plug-ins with a new cleaned version. The web owners need to be notified about the existing lack of security as they usually are not aware of the vulnerability present. This is the reason why a vast majority of web systems on the Internet remain unpatched due to ignoring the system outdate information or due to the low awareness about the potential damage that can happen by vulnerability exploits. Therefore, informing users about the presence of vulnerabilities on a website is essential for the whole Internet space. As a widely accepted system for notifying users about a WCMS website vulnerability was not yet developed, the activity to develop and provide such a system is still in progress.

The need to solve all these problems reported above was explicitly stated at the Europol C3 "The International Cyber Security Summer School (ICSSS)'' workshop (Delta, 2018) in 2018, where experts suggested that a fast, reliable and applicable tool for identifying web vulnerabilities across the whole Internet is required. Most of the problems mentioned above can be avoided by a platform that helps web owners and web users to improve their website security and to enable a safe development of the future Internet. Knowing a system's vulnerabilities represents the most vital and precious information for malicious parties and for that reason the removal of vulnerabilities increases the resilience of the underlying system.

1.1 The Purpose of this Dissertation

Vulnerable software poses a threat to both individuals and organizations. The vulnerability of smaller applications on the Internet can act as a springboard for malicious attackers to gain access to data or the entire network to which the website is connected.

Popular applications such as WordPress offer users a wide range of popular plug-ins. Within a week, month, or year, multiple security patches are usually available that fix serious vulnerabilities in the application or plug-in. Repairing applications or plug-ins is usually not automated and requires human interaction with the system. For individual users of such applications, this means that some applications are left unpatched for a long time due to the lack of knowledge and awareness and are freely accessible to attackers. The statistics collected about vulnerable web applications can provide an overview of how regularly users update their web applications.

The purpose of this dissertation is the intention to find solutions for the problem of the lack of web security by reducing the number of vulnerable websites with a fast, efficient, and powerful tool that can detect vulnerabilities and alert the website owners to improve their website hygiene. To achieve this, an effective tool capable of identifying vulnerabilities of the majority of Internet websites with WCMS applications and informing the concerned users is needed. The main purpose of the research work carried out in the context of this dissertation is focused on providing such a tool and on the studies of the results collected by the tool with assessment of their relevance. The key feature that was specified for the tool was the ability to perform automated, very fast and dynamic vulnerability scans of WCMS-based websites and the attached plug-ins in the live Internet while respecting the ethics requests of the web owners and users. Another purpose of this dissertation was to obtain a clear picture of the web security status of the Internet space that belongs to European registered domains. By identifying the status, possibilities were opened to search for the basic enablers that impact the spaces to be more secure and better protected from various cyber-criminal attacks.

1.2 Goals and Hypotheses

When planning the research and the study, the following goals were defined:

Goal 1. Design a novel scan engine including a sophisticated method for providing a real measure of the vulnerability of WP websites.

Goal 2. Provide an analysis of the current status of vulnerabilities of WP websites.

Goal 3. Discover the enablers that improve the web security within the European WordPress web space.

Goal 4. Compare the proposed method and the developed tool with other existing solutions and expose its advantages.

Goal 5: Investigate the dependencies between the number of insecure websites and the other potential influential parameters that impact the insecurity degree of the WPCMS web space, such as the internet use (IU) in a particular country, the frequency of internet use (FoIU), households with access to internet (HH), the cost of fixed Internet cost access normalised with Gross National Income (GNI), and the Digital Skill (DS) index within European countries.

Goal 6. Establish the security status and the differences among 6 main website areas of application in the web space of European countries: different types of societies, healthcare, finance, news, education, research institutions.

Goal 7. Identify the trends of the web security situation based on the vulnerability data study in two-time terms.

Goal 8. Design a notification service for web owners that will be capable to inform the web owner about its website vulnerability situation based on the owner's request and by assuring non-disclosure of the vulnerability data found.

Closely related to the above goals, the following hypotheses were formulated and investigated (attempting to confirm or disprove):

Hypothesis 1. It is possible to develop a tool for fast, reliable discovering of WPCMS websites' vulnerabilities on an ethical basis in decent time within the large Internet dynamic web space.

Hypothesis 2. The vulnerabilities found within the large Internet web space can provide information about risk factors for a higher presence of vulnerabilities.

Hypothesis 3. There are no significant differences in the vulnerability presence within different types of web services deployed with WPCMS applications.

Hypothesis 4. Running a WPCMS website on a shared hosting web server is a positive risk factor for compromising the running applications.

Hypothesis 5. The trend of the lack of security of websites with WPCMS applications decrease over time and this is considered an information that the awareness about cyber security increases over time.

1.3 Methodology

In order to achieve the goals of the thesis and to prove the hypotheses, the following steps were developed.

First, the problem identification was provided with an exhaustive background overview about WCMS systems. Properties of existing scanners were studied to identify their capacity, efficiency and the success ratio in identifying web vulnerabilities in a dynamic Internet space. Existing different methods of vulnerability identification and the existing scoring systems for security assessment were assessed as well.

In the second step, the problem was defined, and exact objectives of the PhD research were specified. Those results contributed to the specification of the problem relevance.

Third, the design and development step were then focused on the shaping of new methods for accessing and querying websites for obtaining data about the web system and the presence of possible vulnerabilities.

In the fourth step, the analysis of the data collection method was selected. The validity and the accuracy of the data collection was evaluated as well. Known statistical methods for identifying the enablers for lower web security in the selected web space regions were applied.

1.3.1 Development of Novel Vulnerability Detection and Assessment Methods

Several groups of methods were developed. The first group of methods are websites scanning methods based on scanning algorithms such as Google. Here, an appropriate adaptation of the search algorithm for building the start list of Uniform Resource Locators (URL) data for the web targets was developed. The focus was on specifying the queries and getting responses from the website. Different options in the queries were set up for identifying the correct version of the WPCMS core and plug-ins, as they correspond to many different versions used by WCMS during the system development.

The second group of methods was defined for collecting data and building a local vulnerability database that was later used in the scanning that continued after the URL start list was created.

The third group of methods was developed to calculate the vulnerability score. The score for each WPCMS core version and plug-in version were provided based on the found vulnerability. The scores were applied for evaluation of the overall insecurity.

All methods combined are implemented as algorithms and incorporated into the tool. They are publicly available.

1.4 Scientific Contributions

The scientific contributions of the dissertation are summarized as follows:

- Advanced analysis and property evaluation of the current web vulnerability scanning tools for large Internet scanning and the quality and appropriateness of these properties for different types of scanning. Identification of missing features.

- A novel method and developed algorithms for scanning the Internet by following the ethical norms for vulnerability identification in large-scale scans.

- Design and implementation of a score system for quantifying website vulnerability levels.

- Results and findings from the web security study based on the analysis of the data collected with the new scanning tool. The results of this scientific research and study are manifold:

    - Risk factor analysis of plug-ins for WP websites.

    - No significant difference in percentages of insecure WP websites across different sectors of web services was found with the exception of the news sector, which differs from the others.

    - The hosting of several WP websites on the same web server with different URLs has no influence on the occurrence of a higher insecurity among the studied web spaces with WPCMS that host more URLs.

- The percentage of insecure web cores built with WPCMS becomes lower in the second large-scale scan run due to the replacement with the all-updated WP core versions. The presence of vulnerable plug-ins remains the same.

- A correlation exists between the percentage and the level of the calculated insecurity score among the web spaces of examined European countries with the studied parameters. The European websites with WPCMS revealed that a higher percentage of DS among the country population is reflected in a lower percentage of insecure websites from the analysed country's web space with WPCMS applications.

- The lower cost of the fixed Internet access normalized with GNI has a similar effect as the DS index. The percentage of websites that are insecure is lower within a country web space with WPCMS applications where the cost of Internet access is lower.

- The ITU parameters for European countries like internet use (IU), frequency of internet use (FoIU), are highly correlated with household Internet cost (HH) access normalised with GNI. Low-cost value of Internet access is reflected as low percentage of insecure WPCMS websites in the European countries.

The developed algorithms, methods and their evaluations and the case studies have been published as conference and journal publications that are listed in the Bibliography section at the end of this thesis.

1.5 Organization of the Thesis

The remainder of the thesis is organised as follows. Chapter 2 provides the necessary background information. This includes material on web application security covering common web applications, web content management system threats and their vulnerabilities. Chapter 3 provides related work and comparison of popular vulnerability scanning methods.

The main contributions of the thesis are described in detail in Chapters 4 and 0. Chapter 4 presents the design and construction of the methods. The applied methodology, scanning method, feature scoring mechanism and algorithms are presented in detail. The implementation of the proposed methods is presented too. Chapter 0 presents the evaluation of the collected data and the summary of statistical analyses. Finally, Chapter 0 discusses the limitations of the developed methods and their applications. It also discusses the limitation of the research, concludes the thesis and discusses some directions for further work.

Chapter 2

Background

Chapter 2 provides the necessary background information of the dissertation topics. The first section describes a general overview of web applications, open-source WCMSs and WordPress characteristics. The second section describes the web application security including common web application threats and vulnerabilities, followed by potential WCMS attacks and WordPress vulnerabilities. In the third section, web application security testing methods are presented. The fourth section describes crawling methods. The last section presents web vulnerability notification methods.

2.1 Web and Web Applications

Web applications appear in many forms, from small domestic to large-scale commercial services (Google, Twitter, Facebook). The number and the importance of web applications are growing exponentially fast in a world where web browsers are ubiquitous and used by everyone. In fact, millions of people rely on web applications for work and leisure and for accessing and managing sensitive information, such as financial and personal data. Although the web services provide a simple interface between the provider and the consumer, the service itself is supported by a complex software infrastructure which typically includes an application server, a content management system, and a set of external systems (e.g., databases, payment gateways and similar applications).

With the advent of the internet and online business, the development of Web Applications has gained in importance. The World Wide Web Consortium (W3C), the main international standards organization for the World Wide Web, clearly specifies the definition of Web application, which refers to a “Websites delivered over HTTP which use server-side or client-side processing (e.g., JavaScript) to provide an "application-like" experience within a Web browser” (W3C, Mobile Web Application Best Practices, 2021). Web applications generally consist of client-side and server-side programs. The client-side is sometimes referred to as front-end and the server-side as back-end. The client-side of a website refers to an application that is processed within the client user device, where everything is displayed including text, images and other UI components. The client-side code is sent to the client from the server-side, where the data and source code are stored. Validation, authorization, database, and controller logic are handled on the server-side of the application while the client-side can be used to examine the user’s form for errors before submitting it to the user input. When a user makes a request via a web browser, it transforms the Hyper Text Mark-up Language (HTML) and Cascading Style Sheets (CSS) and the browser interprets and renders the web content on the client-side (W3C, 2020).

Back-end is an inevitable part of website development and backend developers provide the largest part of creating the web application. Different programming languages are used for backend development (e.g., Hypertext Pre-processor (PHP), Python, etc.).

PHP is a server-side scripting language, and its main capability is to set up a dynamic website. PHP stands for "Hypertext preprocessor", which literally means fast text preprocessor (Welling & Thomson, 2003). In addition to being considered as a language that is relatively easy to master, its flexibility is also one of those features that make PHP one of the most popular scripting languages. The use of this language continues to increase for both business and individual purposes, as it is considered an alternative to Microsoft's software tool, ASP.NET. There are some popular PHP frameworks such as Laravel, Codeigniter, Zend, etc.

In general, developers write scripts in the Python programming language for back-end scanning, parsing logic, and web applications. Python is an object-oriented, high-level programming language with dynamic semantics (Lutz, 2013). Its data structure, combined with dynamic typing and dynamic binding, is the reason why Python is very popular for fast application development as well as for combining existing software components. Python is considered a programming language with a simple and easy-to-understand syntax, which gives it an advantage over some other languages when choosing the first programming language to learn.

A comparison of the differences between the client-side and the server-side is presented in Table 1.

Table 1: Comparison of the client-side and the server-side (Viva Differences, 2020).

Basis of Comparison	**Client-side scripting**	**Server-side scripting**
Use	Works on the frontend and is visible to users	Works in the backend, which cannot be seen at the client's end
Processing	Does not need interaction with the server	Requires server interaction
Supported Languages	HTML, CSS, JavaScript, etc.	PHP, Python, etc.
Running	It runs on the user's computer	It runs on the web server
Execution	The client-side scripting process is performed on the user's computer, so the response is comparatively faster compared to server-side scripting	The scripting process for the server-side code is performed on the remote server, so the response is slightly slower than the client-side scripting
Database Connection	No database connection	Connects to the databases that are already present on the server
Access to Files	No access to all files on the server	Access to all files on the server
Source Code	The source code is visible to users	The source code is not visible to users

Security	Less secure as the scripts are usually not hidden from the client end	Relatively secure, but more secure than client-side scripting as the server-side script is usually hidden from the client end

The database is also important in the development of the backend system. It is one of the main components in the software product. There are many choices of database types which a software developer can choose. In general, there are two types of databases (Sahatqija, Ajdari, Zenuni, Raufi, & Ismaili, 2018):

- Structured Query Language (SQL) Database is a relational database which is designed in a tabular form and contains schemata. Popular SQL databases are My Structured Query Language (MySQL), Oracle, Microsoft Structured Query Language (MS SQL) Server, Sybase, etc.

- Not only SQL (NoSQL) Database are self-describing and do not require schemata. Tabular structure is not included. Popular NoSQL databases are MongoDB, Redis and Casandra.

Database management usually requires knowledge of SQL (Server Query Language), which is a server query language. SQL enables communication with the server. For example, the use of PHP is most often present when working with MySQL, and ASP.NET with MS SQL Server.

2.1.1 Open-Source Web Content Management Systems

Over the last few decades, it become usual to acquire software by purchasing ownership licenses for web applications. However, in recent years, this habit is more and more often accompanied by a programming model characterized by software that comes with a compilable source code. This type of software is called "open-source software" (OSS) (Schryen, 2011).

Open-source software (OSS) is software whose source code is freely available and can be freely used, explored, modified and distributed, both the original and the amended copies; unlike the closed source software where all of the above is not enabled (De Laat, 2005). The terms of use of the open-source software are set out in various licenses that contain guidelines for using open-source software (Opensource, 2019). In the last decade, free open-source systems become the most recognizable in the field of web management.

Content Management Systems (CMS) are general software solutions that enable the creation, management, editing, publication and research of various content (Wikipedia, 2019). In addition to managing electronic documents, management capabilities also include various files like images, audio and video recordings. The content is stored, and the systems provide services such as time-dependent publishing, document flow, version control, access control through applications.

The Web Content Management System (WCMS) is a content management system that is implemented as a web application and offers the possibility of web content management. A WCMS is a system that manages and automates the process from preparation to publication of the content of a web presentation. WCMS's provide users with authoring tools that allow them to create and edit content without knowledge of programming (Wikipedia, 2019).

Prior to the introduction of WCMS, websites were mostly static or based on expensive systems made by the website owner. As a result, the website owner had to be highly educated or had to find an external experienced partner for this purpose. This method of

management is much slower, as the response time from preparation to publication of the website on the server could be from a few days to several weeks. According to a survey carried out by W3Tech, about 52.9% of Internet websites use some kind of WCMS (W3Tech, 2019).

The website frontend is separated from the content and is defined as template. This feature allows users to easily change the appearance without having to interfere with the stored content. Most of the systems use the database to store the content of the website, while some store the content in a file on a disk.

The most popular open-source web content management systems are WordPress, Drupal and Joomla (Hassan, Sarker, Biswas, & Sharif, 2017). However, customer support is especially important for business websites and therefore the use of free open-source WCMS can pose significant problems for companies. Despite the fact that users want ease of use and the cheapest offer, still WCMS providers need to provide support and appropriate updating and security of websites or plug-ins.

2.1.2 WordPress

WordPress appeared on the web in 2003, when it was introduced to the public as a simple blogging platform where users could write text, post images and link to other websites. Due to its ease of use, WordPress is one of the most widespread WCMSs. The WP platform was selected for this research due to the popularity of the software and its very high presence in the world web space. Its popularity lies in the easiness of setting up websites, its low cost, friendless of use and maintenance. The abundance of plug-ins for developing many different types of services and scenarios, such as blogs, social network applications, webmail services, banking, e-commerce, or educational services, contribute to its popularity. WordPress is the most widely used open-source WCMS and supports the creation of websites of companies like NBC, CNN, TED, New York Times, Forbes, eBay, Best Buy, Sonny, UPS, CBS Radio, TechCrunch and others (WordPress, 2019) (da Fonseca & Vieira, 2014).

WordPress is based on PHP scripting language and MySQL database. It is easy to learn and use. It allows users to create websites, even without knowing the HTML or PHP language, which means that it is also suitable for beginners. Although its basic functionality is extremely limited, it is widely used because of the availability of many plug-ins which extend its functionality according to the needs. Themes and plug-ins are individual pieces of software that act as add-ons to the basic WP platform. Users can install a theme or plug-in to access additional features. In order to display content, WordPress must have an activity that contains PHP functions with WordPress' own syntax, HTML code for structuring the page, and CSS code to determine the style of displaying content.

The open source and widespread use of the programming language used for its core functionality have contributed to the development of WCMS with the most diverse functionalities that are known today.

Although in most of cases plug-ins make the work easier and shorten the time to add certain functionalities to a page, users should be aware that plug-ins have some negative properties too. Since community plug-ins can be contributed and developed by anyone, this means that they can be unoptimized, which consequently slow down a website's performance. They may not be compatible with all other plug-ins, they could include a security issue, or they may no longer be supported when a newer version of the WordPress core is released. This means that users need to be careful when using plug-ins and consider whether they need them or not.

WordPress is licensed under the General Public License (GPLv2 or later) which provides four core freedoms and can be considered as the WordPress "bill of rights" (Rosso, et al., 2015)

2.2 Web Application Security

The rapid development of web applications allows users to utilize this benefit to a great extent, so they use the Internet to perform most of their daily activities such as online content reading, shopping, electricity bill payment, ticket booking, gaming, and other forms of entertainment. This phenomenon causes that users are connected to the internet for longer periods of time and increases the risk of exposure to internet abuse (Jang-Jaccard & Nepal, 2014). Attackers try to gain access through various social engineering attacks, such as phishing, voice phishing (vishing), SMS phishing (smishing), email phishing, etc. Social engineering attacks are either non-technical or they use low-tech methods to obtain information by using fraud or tricks (Salem, Hossain, & Kamala, 2010). Scammers are still successful in their phishing attacks in spite of the warnings given by the local Computer Emergency Response Centres (CERTs).

Web applications, especially those exposed to the Internet, are much more vulnerable to threats than desktop applications, as they can be accessed by many users and access is more difficult to control. Nowadays, attacks via the Internet are often almost completely automated and many tools allow people with even a small amount of technical knowledge to exploit vulnerabilities (Catakoglu, Balduzzi, & Balzarotti, 2017).

Both types of web applications, corporate and individual, handle very sensitive information that are of interest to cyber criminals and web attackers. Even applications which don't handle sensitive or valuable data have become interesting targets, as they can be used for different malicious gains such as serving malicious content. In that context, knowing and being aware of web vulnerabilities becomes an important value and task.

For small and medium size companies, even one day of unavailability of the website could bring a significant decrease in prosperity to their business, therefore they are the most endangered. Unfortunately, most of them see web-security more as a cost and not as an investment; therefore, these companies are the most vulnerable (Mansfield, 2020).

Modern web application development includes many challenges, and of those, security is both very important and often forgotten or under-emphasized. Writing web applications can be a complex task, and in addition, web applications are using libraries to make applications programming faster and easier. While techniques such as threat analysis are increasingly recognized as essential to any development process, there are also some basic practices that any developer could or should not take for granted.

Security vulnerabilities are flaws in the design or in the level of implementation of the web application. Flaws may not be detected for a long time before they are used for exploitation. Furthermore, the terms "bugs" and "flaws" appear frequently in literature. McGraw defines a bug as an implementation-level software problem ("fairly simple implementation problems"), whereas a flaw is a problem that is "certainly instantiated in software code but is also present on the design level" (McGraw, 2006). Examples of flaws include error-handling problems and broken or illogical access controls. Bugs and flaws create risks, which are the probability that a flaw or a bug will impact the software's purpose: risk = probability × impact. Software defects might exist for a long time before attackers actually exploit them (McGraw, 2006).

There is always a possibility of flaws in the application. A flaw in the operation of an information system, an error in the design or in the implementation can result in a vulnerability. From the beginning of the design process, both design and implementation

need to be coherent and take all the security principles into account to avoid the risk (Potter & McGraw, 2004). There is a growing need for web applications, given that several million domains are currently alive and the demand for development is high. As a result, companies or individuals are rapidly developing online solutions to meet customer requirements, leaving many errors open that can lead to serious problems. Code review is extremely important for identifying implementation errors considering the fact that implementation errors can be just as harmful as design errors.

The study of Bou-Harb et al. gives an example of hackers taking Sony's PlayStation Network offline for 24 days resulting in an estimated cost of $171 million for the company (Bou-Harb, Debbabi, & Assi, 2013). In addition, they pointed to the Citibank detection of a data breach that publicly exposed 360,000 North American credit card holders' account information, including their numbers, names, and email addresses. Moreover, according to the authors, companies as Twitter, Facebook and Google were targets of hacking attacks (Bou-Harb, Debbabi, & Assi, 2013). Summarizing all what has been mentioned above, we can highlight the necessity for progress in improving cyber security.

2.2.1 Open-Source Code Security

In information security, open-source software gives each individual the opportunity to control their own system. Each individual is responsible for the security of their own system. The open-source community comes in very handy, as it publicly warns about the faults and gives everyone the opportunity to contribute to the greater security of systems by developing their own patches and proposing solutions. This reduces the possibility of open-source software containing malicious code, for example a side door through which an unauthorized person can gain access to the system.

Table 2: A comparison of the properties of Open and Closed Source code.

	Open-Source Code	**Closed Source Code**
Use	free for use, distribution, permitted interventions in the code	use permitted under strict conditions
Ownership	Publicly owned	The owner is a corporate company
Detection of security errors	Greater possibility of detecting security faults: the code can be viewed by all users, which can lead to faster detection and error correction	Less chance of detecting security errors: security fixes are prepared only by programmers within the company that owns the code
Responsive time	Shorter time between identified security deficiencies and security corrections issued	Poorer responsiveness in case these errors are detected
Security	The code is open and anyone can exploit its vulnerabilities	The code is closed, and it is difficult to identify faults in the code that could be exploited by attackers
Development	Decentralized - users participate in the development, development takes place in communities	Centralized, closed and goal-oriented development
Reliability	Successful updating requires the continued involvement of the programming community. It happens that the lead programmer abandons the project	Product availability depends on commercial performance
Support	External support usually only against payment. It is difficult to find support for less widely spread products	Support included in the license price or enabled as an additional surcharge on an annual or monthly basis

Quality	No one is responsible for the quality of the code. However, the code is being constantly reviewed by users	More efficient management by one corporation, where a series of programmers make sure that the software is (supposed to be) error-free

Proponents of open-source software argue that the open-source community can help with detection of bugs in a code. In practice, this is not entirely the case, as most users today do not have programming skills, they do not know how to read the source code or identify the shortcomings in it. Therefore, as much as the open-source community advantage can be, it is also a disadvantage, as attackers can get to know the source code and its vulnerabilities, and exploit them in web attacks before they are detected and patched (Ransbotham, 2010).

Empirical results have shown that open source and closed source software do not differ significantly in terms of the severity of the vulnerability. Although open-source software development appears to prevent "extremely poor" patch behaviour, there is generally no empirical evidence that a particular type of software development is a major driver of insecurity. Therefore, the software vendors are most strongly encouraged to provide safe software and to avoid vulnerabilities from the outset (Schryen, 2011). A comparison of open-source software and proprietary software is given in Table 2.

2.2.2 Common Web Application Threats and Vulnerabilities

The web service usually includes several operations that are remotely invoked by a client. The multi-layered architectures of web-based systems and their sophisticated interactions with different types of subsystems increase the number of flaws that can be exploited by attackers with malicious intentions. Attackers can follow various paths through the digital infrastructure of an organization, with the aim to find security weaknesses to exploit, where severe consequences might result. That is why web services are, in general, subject to several security concerns. According to Symantec (Symantec, 2019) there was a global increase of 56 percent on web related attacks in 2018 and 23 percent were expected to have critical consequences. Sfakianakis et al. reported that 351,913,075 unique malicious URLs were detected in the second quarter of 2018 and due to the rising number of the vulnerabilities, website security was ranked first among serious cybersecurity problems of the whole internet (Sfakianakis, Douligeris, Marinos, Lourenço, & Raghimi, 2019). Krsul defines web vulnerability as a weakness in particular system that can be exploited to cause harm, although its presence does not cause harm by itself (Krsul, 1998). Notification or knowledge about a website's vulnerability is important as it enables the owners to remove the bug either by themselves or by replacing the compromised components with a new, cleaned version. The vulnerability of the web's popular servers is usually the result of the fact that programmers lack the required knowledge about secure coding and that the web server's owners lack the awareness of the need to protect the services they offer.

Today there are several established organizations working primarily on solving security problems. The Open Web Application Protection Project (OWASP) is a non-profit foundation that aims to highlight the importance of cyber security, web application security, and improving security in software development. Through open-source software programs, OWASP guides communities across hundreds of local chapters worldwide by providing training conferences trying to increase security awareness. The OWASP Top 10 is a standard awareness project for developers and web application security, which represents a broad consensus about the top ten most critical security risks. The most recent version of the top ten list was published in 2017 and is presented in Table 3 (OWASP, 2019).

OWASP prioritized the top ten vulnerabilities according to their prevalence and their relative exploitability, detectability, and impact. Companies or individuals should adopt such documents for better awareness and start the process of ensuring that their web applications minimize these risks. Using OWASP Top 10 as a reference for application security is probably the most effective first step in changing the software development to produce a more secure code.

The top ten vulnerabilities were considered in building the signature vulnerability database. A comparison of the OWASP Top 10 vulnerabilities from 2010, 2013 and 2017 is presented in Table 4. As visible SQL Injection remains first among the top ten vulnerabilities in the last ten years. The vulnerabilities A4 - XML External Entities (XXE) and A7 - Cross-Site Scripting XSS in the 2013 list were merged in 2017 list into a single vulnerability, A4 – Broken Access Control. The vulnerability A10 - Unvalidated Redirects and Forwards, dropped from the 2017 list, whereas two new vulnerabilities (A7 – Insufficient Attack Protection and A10 – unprotected Application programming interfaces (APIs)) were added to the 2017 Top 10 list. Based on the comparison presented in Table 4, it is evident that the top ten vulnerabilities have not changed significantly over the last ten years and web vulnerabilities remain a major problem for cyber-attacks.

Table 3: OWASP Top 10 Most Recent List.

A1	Injection
A2	Broken Authentication and Session Management
A3	Sensitive Data Exposure
A4	XML External Entities (XXE)
A5	Broken Access Control
A6	Security Misconfiguration
A7	Cross-Site Scripting (XSS)
A8	Insecure Deserialization
A9	Using Components with Known Vulnerabilities
A10	Insufficient Logging & Monitoring

The following list presents a summary of the Top 10 Application Security Risks, and the risk factors. The following factors were determined based on available OWASP statistics and experience (OWASP, 2019):

1. Injection means a lack of sanitation of user input data that allows injection to succeed. Injection flaws such as SQL, NoSQL, Operating System (OS) command, and Lightweight Directory Access Protocol (LDAP) injection are compromised by sending untrusted data to an interpreter as part of a command or query. Injection flaws are one of the most common hacking techniques. Using the technique of exploiting gives malicious hackers access to data in a database. The separation of code and data, escaping special characters and data validation can help prevent vulnerabilities.

2. Broken Authentication and Session Management is also one of the most common risks. These vulnerabilities are due to a corrupt web application design. Building custom application functionalities related to the authentication and session management are often implemented incorrectly. Attackers often take advantage of leaks in the authentication process or in session management to access sensitive data. Attackers can interrupt the authentication process by manual means and take advantage of it with automated tools with password

lists and dictionary attacks. Generally, these parts of applications have flaws allowing attackers to compromise passwords, keys, session tokens, or to exploit other implementation flaws. Good authentication libraries, multi-factor authentication (MFA), strong passwords and detection of brute force or stuffing can help prevent such faults.

3. Sensitive Data Exposure jumped from the sixth place in the 2013 OWASP report (OWASP, 2013) to the third place, meaning that such attacks are now more common. Some web applications and APIs do not protect sensitive data properly. According to the OWASP (OWASP, 2020), weak key generation and management, and weak algorithm, protocol and cipher are the most common cause for this flaw. Attackers may steal or manipulate such weakly protected data. To ensure that a website is compliant, only the users who need to use it can have access to it, and of course the data must be encrypted. The same goes for customer email addresses and news lists. Ideally, website owners should not store such information on their websites. Storing data on third-party services, such as Mailchimp, which is a more secure and reliable infrastructure, is an alternative solution to user's website.

4. XXE was recently added to the OWASP top 10 List. Any poorly configured XML processors evaluate external entity references within XML documents. Older XML processors by default allow the specification of an external entity Uniform Resource Identifier (URI), that is scheduled and evaluated during XML processing. Such flaws can be used to retrieve data, perform a remote request from the server, scan internal systems, perform a denial-of-service attack, and perform other attacks. Developers should avoid XML or at least use modern libraries and configure them well to prevent such faults.

5. Common Broken Access Control problem occurs when an application does not enforce the required restrictions for authenticated users. Vulnerability includes bypassing the access control check by modifying the URI. By changing primary key other users can access accounts, manipulate metadata, view sensitive files, modify other users' data, change access rights, etc. This type of problem can only be fixed by developers using proven code or libraries, failure logging and alerting.

6. Security Misconfiguration is the sixth item on the list and is also a common flaw related to web application security. Misconfiguration can happen anywhere in the application. This means that an attack can occur due to flaws in the platform, web server, framework, database or any custom code related to the application. A very small misconfiguration can put users' data at stake. Unpatched software and the exploitation of the default configuration are the two most common attacks on WCMS websites. To prevent such faults, unused services should be disabled and setting review tools can be used. WordPress, for example, no longer has a default administrator username, which has been the cause of many WordPress intrusions.

7. XSS, previously very high (3rd place) on the list, is now the seventh item on the OWASP Top 10 list. It is probably the most common fault that affects web applications. XSS generally happens when a website without proper validation or escaping user input allows attackers to execute scripts in the victim's browser.

In this attack, basically, the user's browser is fooled to perform unintended actions such as deface websites, or it redirects the user to malicious websites without the knowledge of the victim. Safely rendering all user-supplied data, appropriate encoding for the context and using template frameworks that assemble HTML safely can help to prevent such faults.

8. Insecure Decentralization is a new item that was recently added to the OWASP Top 10 list. Programming languages allow to turn a tree of objects into a string that can be sent to the browser. If untrusted data is deserialized, it may allow objects to be created, or code to be executed. Deserialization flaws can be used to perform attacks, including injection attacks, replay attacks, and privilege escalation attacks. To prevent such faults, serializing of objects should be avoided and signatures for detecting tampering can be used.

9. Using Components with Known Vulnerabilities is a very common problem, as almost all modern applications contain or depend on a lot of third-party libraries to speed up the development process. If vulnerable components such as libraries, software modules that run with the same privileges as the application, are exploited, this facilitates serious data losses, or a server takeover can happen. Also, attackers can enumerate the libraries used, and develop exploits. To reduce such flaws, users should reduce dependencies, scan for out-of-date components and use patch management.

10. Insufficient Logging and Monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. If there is no sufficient logging or monitoring, is not possible to react to attacks which are not known. Activity logs can improve website management and identify suspicious behaviour before it becomes a problem.

Table 4: A comparison of the OWASP Top 10 vulnerabilities from 2010, 2013 and 2017.

OWASP Top 10 - 2010		**OWASP Top 10 - 2013**		**OWASP Top 10 - 2017**	
A1	Injection	**A1**	Injection	**A1**	Injection
A2	XSS	**A2**	Broken Authentication and Session Management	**A2**	Broken Authentication and Session Management
A3	Broken Authentication and Session Management	**A3**	XSS	**A3**	XSS
A4	Insecure Direct Object References	**A4**	Insecure Direct Object References	**A4**	Broken Access Control (**Merged 2013-A4 and 2013-A7**)

A5	Cross-site request forgery (CSRF)	**A5**	Security Misconfiguration	**A5**	Security Misconfiguration
A6	Security Misconfiguration (NEW)	**A6**	Sensitive Data Exposure (**Merged 2010-A7 and 2010-A9**)	**A6**	Sensitive Data Exposure
A7	Insecure Cryptographic Storage	**A7**	Missing Function Level Access Control	**A7**	Insufficient Attack Protection (**NEW**)
A8	Failure to Restrict URL Access	**A8**	CSRF	**A8**	CSRF
A9	Insufficient Transport Layer Protection	**A9**	Using Known Vulnerable Components	**A9**	Using Components with Known Vulnerabilities
A10	Unvalidated Redirects and Forwards (**NEW**)	**A10**	Unvalidated Redirects and Forwards	**A10**	Unprotected APIs (**NEW**)

Information about system break in attacks should be monitored actively and reported to the development team to improve and update the web applications. Different web applications can be hosted within the internal company network or outside the company on shared hosting. Even if an application is within an organization which not properly separate different tiers, it often leaves an application vulnerable, enabling an attacker who has found a fault in one component to quickly compromise the entire application. In shared hosting environments, various processes occur where it is sometimes possible to exploit malicious code or a vulnerability in a single application that can compromise the entire environment and other applications running on it (Stuttard & Pinto, 2011). Web applications on shared hosting are one of the issues examined in this thesis.

2.2.3 Potential WCMS Attacks and Vulnerabilities

Due to the widespread use of WCMS, the potential damage from attacks is enormous. If WCMS is used as an online store, an attacker can obtain confidential customer information and credit card information. Also, if the WCMS has not been properly secured, an attacker may disclose confidential information and thereby cause claims for damages. In addition, a malicious attacker can collect user information, such as customer addresses or profiles, and sell it to competitors. WCMS can also be sabotaged by making it inaccessible, leading to a drop in sales. If the WCMS is used for business websites, there are also several dangers, as an attacker can use security vulnerabilities to load malicious code and damage the company's information infrastructure. An attacker can also change the content of a website, for example by adding questionable and suspicious content that has a negative effect on a company's reputation.

Even applications that do not handle sensitive or valuable data have become interesting targets. Motivation may vary among gathering or destroying confidential data, misuse of the web server for illegal activities (serving malicious content) or simply preventing users from accessing the application. There are various attack patterns that can be used to pose threats to web applications. Threats affect one or more security aspects. According to Huang et al. and Patel et. al, the threats of web applications include threats such as data manipulation, XSS, execution of code, and CSRF (Huang, Zhang, Cheng, & Shieh, 2017) (Patel, Rathod, & Prajapati, 2013).

Soska and Christin in their paper "Automatically Detecting Vulnerable Websites Before They Turn Malicious" also stress the problem of vulnerability of content-management systems. They pay attention to the fact that only websites with an active infection can be identified and the purpose of the paper was to find a solution that would prevent these attacks before the website was compromised (Soska & Christin, 2014).

From a security point of view, the websites have a system with weak passwords. Access to the website is gained either by guessing passwords (for example, the name of a famous person) or by a brute-force dictionary attack through dedicated software. Defending against such attacks is extremely simple, as it is sufficient to choose a secure password (at least 16 characters, three different characters, punctuation marks, and numbers).

Figure 1 presents the possible attacks that a WCMS might face. Web applications such as WCMS may encounter also database-level attacks with SQL Injection or web server-level attacks that attempt to manipulate web parameters, maliciously upload a file, elevate privilege, relay spam, bypass authentication, or hijack a session. The user can also become a victim of XSS or spam attacks.

WCMS applications are used by both professional and non-professional users. WCMS usually manage sensitive information and non-professional users often lack an in-depth technical and security expertise. The application has vulnerabilities that are easily exploited by attackers. Users should take precautions to reduce the risk. In particular, non-technical users should always use the latest available version of WCMS, and technically experienced users can use older versions, if they are aware of the potential security faults and regularly monitor vulnerability updates and countermeasures provided by the WCMS communities.

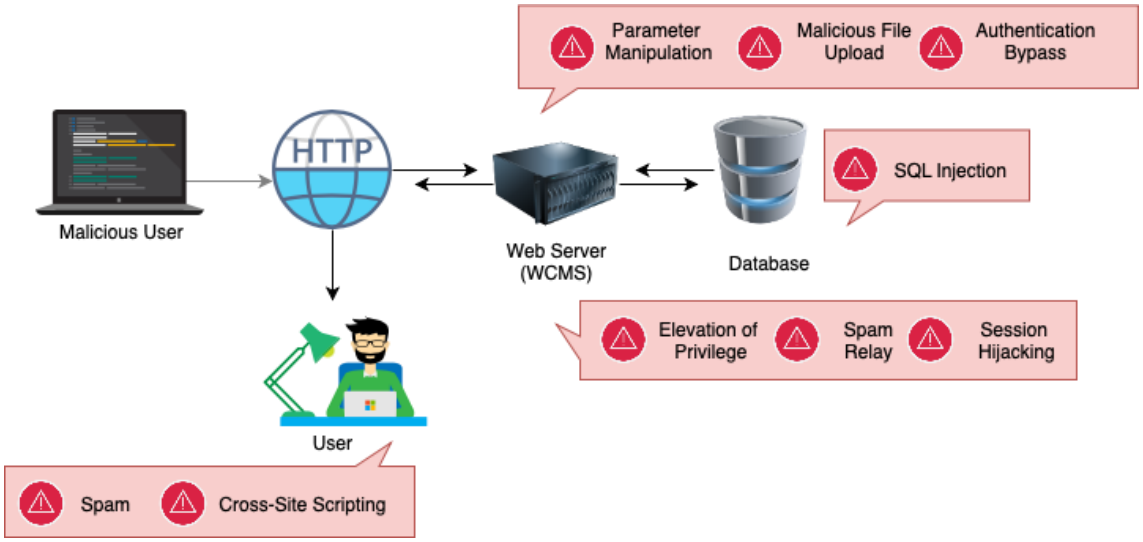

Figure 1: Potential WCMS attacks.

2.2.4 Types of Vulnerabilities of the WordPress application

Open-source content management systems such as WordPress are extremely popular among webmasters. Due to its popularity, WordPress is also very attractive for online attackers who are massively interested in websites that have not been updated for a long time. Open source WCMS are targeted by attackers not only because of their popularity, but also because they allow attackers to examine security flaws in the open-source code. Not only the owners of the mentioned vulnerable websites are at risk, but also the hosting providers and consequently all other co-hosted websites on the same server.

Generally, the target of the attacks are millions of outdated WP websites. The attackers upload their code on the servers through publicly known vulnerabilities. The vulnerable server then executes all of their malicious commands that are within the role rights of the assigned web server user.

WP websites have pingback enabled by default and therefore they can be used for Distributed Denial-of-Service (DDoS) attacks on other websites (Silaen & Lim, 2016). Due to a lack of security, WordPress was the target of an extremely large DDoS attack in 2011. The scale of the attack was several gigabits per second and tens of millions of packets per second. In addition, at that time they were already serving 18 million blog content publishers, including VIPs TED, CBS and TechCrunch, and were responsible for 10% of all websites in the world (Tsotsis, 2011). In January 2017, the WP community released WordPress 4.7.2 to fix three security faults: an SQL injection vulnerability in WP_Query, an XSS vulnerability in the posts list table, and an unauthenticated privilege escalation vulnerability in a REST API endpoint (Campbell, 2017).

According to the survey conducted by Vasek et al., web servers running WordPress and Joomla are more likely to be hacked than those not running any WCMS (Vasek, Wadleigh, & Moore, 2015). Additionally, it was found that servers running Apache and Nginx web server for WP applications are more vulnerable than those running Microsoft Internet Information Service (IIS). Furthermore, using a series of logistic regressions, they found that a CMS's market share is certainly interrelated with website compromise. The authors also presented the evidence that web servers running outdated software are less likely to be compromised than those running up-to date software such as the core WordPress software and many associated plug-ins (Vasek, Wadleigh, & Moore, 2015).

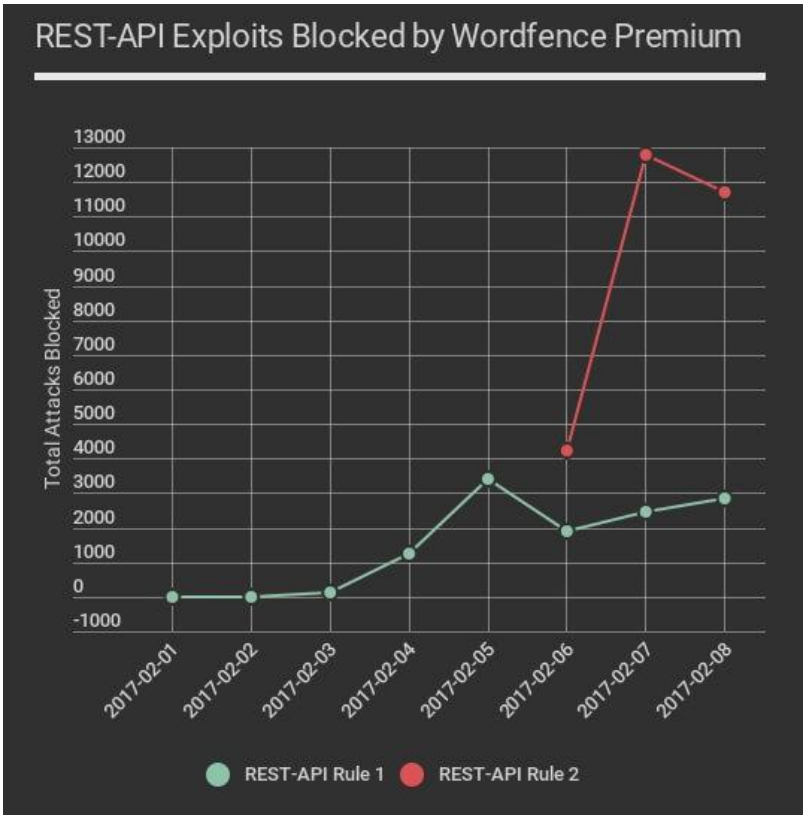

Figure 2: REST-API and Exploits Blocked by Wordfence Premium (Maunder, 2017).

There are also statistics about the attacks on websites running an outdated version of WordPress from February 2017. According to the Figure 2, the blocked attacks that tried

to exploit the bug began around 3 February. The attacks progressively increased in the next few days. On 6 February, around 4000 hacker activities were blocked, and one day later, 13,000. During those two days, more than 800,000 attacks happened across all WP websites (Maunder, WordFence, 2019).

According to Moen, 61.5% of the 1032 survey respondents did not know how the attackers compromised their WP website (Moen, 2016). Moreover, Moen states the difficulty of cleaning the website when the problem of occurrence is not known in the first place. Moen emphasizes the need for a tool which would predict the attack before its occurrence (Moen, 2016). Figure 3 presents the main reasons collected from respondents for compromised websites.

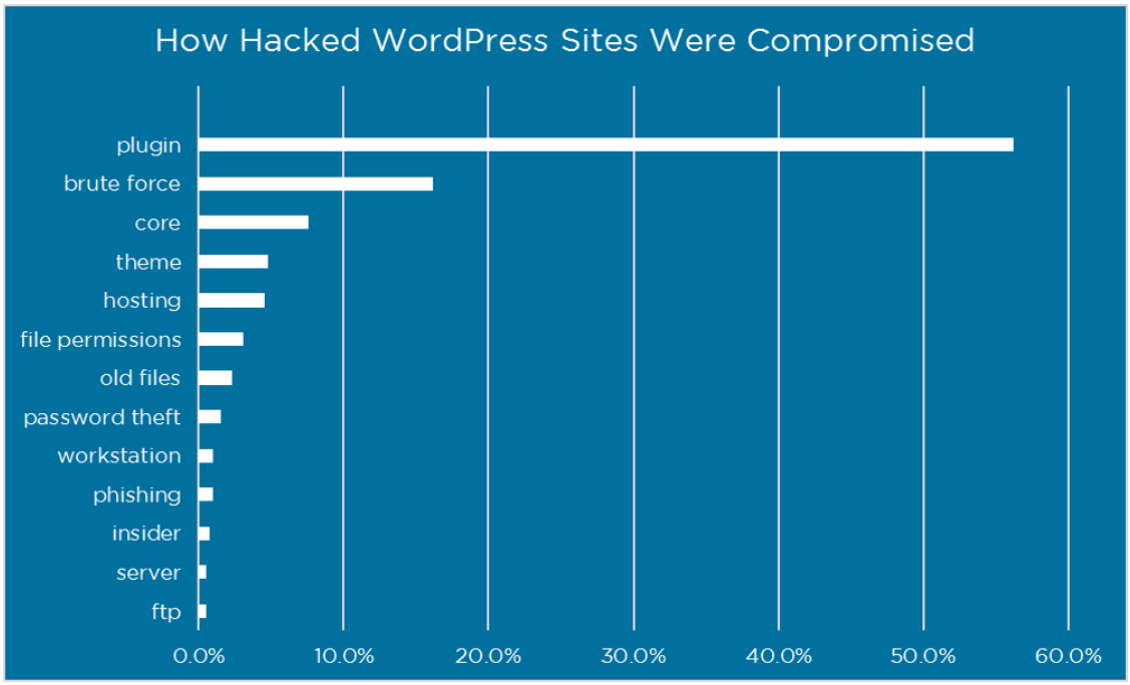

Figure 3: How hacked WP websites were compromised (Moen, 2016).

The graph in Figure 3 shows that in 2016, over 70% of the analysed WP websites being compromised were caused by plug-in vulnerabilities and brute force attacks; therefore, wise protection from these two weaknesses would improve the situation a lot.

To understand how the underlying security bugs can be discovered and exploited by attackers, Trunde and Weippl in their research analysed 199 publicly disclosed SQL injection exploits for WordPress and its plug-ins (Trunde & Weippl, 2015). They claim that previous research has shown that the majority of security bugs are caused by the same programming errors as 10 years ago and state that the complexity of finding them has not increased significantly. Furthermore, in their study they claim that although the complexity has not increased, automated tools still do not detect the majority of bugs. Based on the results of their study, they claim that to increase the informative value of their research it would be reasonable to test WordPress and its plug-ins with other well-known tools. Further they conclude that results of different crawlers would be interesting as crawling is the most important phase during testing (Trunde & Weippl, 2015).

WordPress servers are not fully protected and they are exploitable in most cases due to the use of third-party plug-ins. A study by Koskinen et al. discovered the presence of 860 vulnerabilities in a set of 127 plug-ins, belonging only to 322 WCMS websites (Koskinen, Ihantola, & Karavirta, 2012).

Having information about website vulnerability is a crucial data for website owners and maintainers. Since most of the WCMS applications are designed modularly, they are very flexible in terms of adding new functionalities and modifying existing ones according to the needs and requirements of the client or the website owner. With an individual module or template, website editors can edit certain content. However, the question arises how well the security, updating of the system and administrative tasks are taken care of.

2.3 Web Application Security Testing Methods

To facilitate the understanding of different approaches of web vulnerability testing in the continuation of the dissertation, the most common approaches for testing and detection are presented. The main purpose of software testing is to inspect the product or service for a review of the quality of the software. Software testing therefore shows an objective and independent view of the software and allows the developers to understand and appreciate the risk of software implementation (Kaner, 2006).

Software testing is independent of the state of the development process, as it can be performed at any stage. Different software development models provide testing at different stages of the software development state. In traditional development models, most testing is done after the requirements have been defined and the coding process has been completed. Newer models that rely on agile methodologies, prefer to use test-driven development, which requires testing for the developer in development process before the program code is completely written and sent to test groups (Hammond & Umphress, 2012).

Testing software consist of designing tests, the activity of running a series of dynamic executions of software programs and evaluation of the tests' output after the software source code has been developed. It is performed to uncover and correct as many potential errors as possible before production. It can be considered a risk management technique; a quality assurance technique. It represents the last defence to correct deviations and errors in the specification, design, or code (Lewis, 2017).

Figure 4 presents a life cycle model for testing. In the development phases, three opportunities arise for errors or mistakes to be made, resulting in faults that may propagate through the rest of the development process. The fault resolution step is another opportunity for mistakes and new faults. When a fix causes the previously properly executed software to malfunction, the fix is defective (Jorgensen P. C., 2018). To detect faults in the development process, the time of fault detection or the development phase in which the fault is detected is essential. If the fault is detected in the early stages of analysis or in the specification development, the cost of the detected fault is negligible. Faults discovered in the programming or testing phase can result in a cost that is ten or a hundred times higher than the initial one.

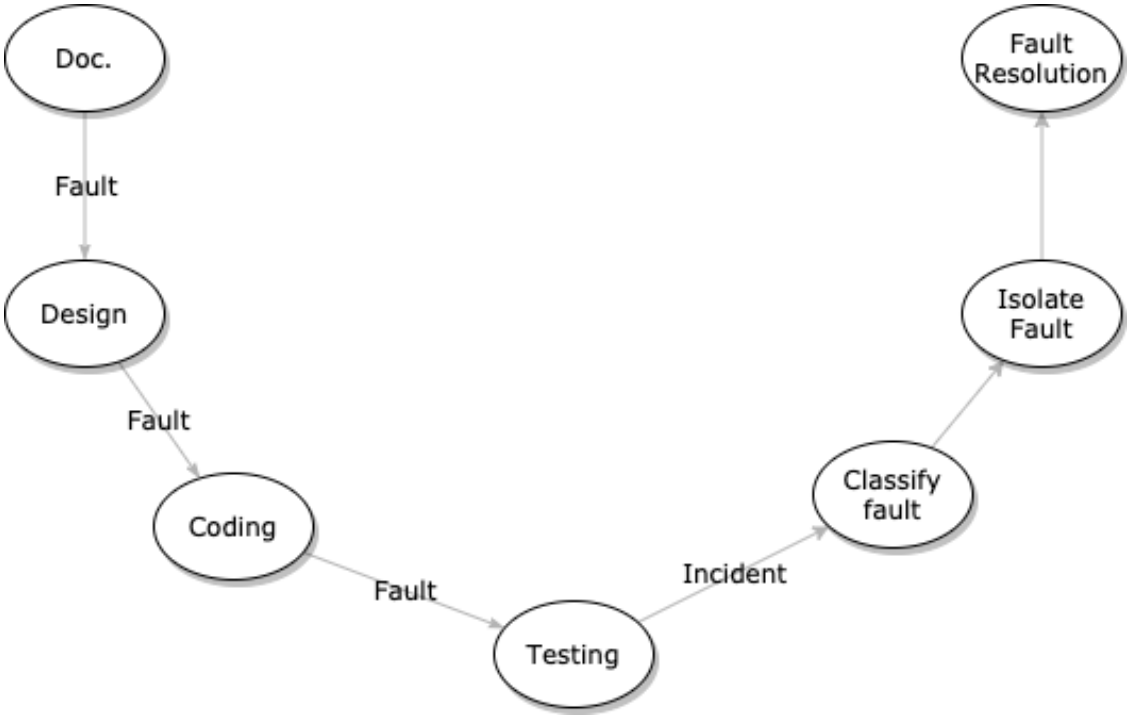

Figure 4: A testing life cycle (Jorgensen P. C., 2013).

One of the common mistakes that occur when planning testing is the silent assumption that many mistakes are not detected. The logical consequence of this error is an underestimation of the planned resources (people, test time periods), which is a widespread problem in most cases of the development process. In fact, the testing process often occurs only at the end of the development cycle. Another important limitation of software testing is that testing can show only the presence of failures, and not their absence (Ammann & Offutt, 2016).

As the protection of personal and other data becomes more and more important, some systems have special security tasks. Security testing is a process in which the task tries to trick the security control of a system with imaginary values. However, this kind of testing by trying to enter imaginary values is a difficult task and it only reveals that specific errors are not in the software under certain test conditions. These tests do not actually prove that the tested applications are immune to all possible attacks, as new faults can always be detected. Therefore, online applications require a greater emphasis on security testing when dealing with online commerce, banking or social applications.

Penetration testing is a combination of techniques to identify areas of the system or application that are vulnerable to intrusion by simulating an attack as if it had been carried out by a malicious unauthorized person. It is based on a structured procedure that gradually performs penetration testing. Penetration testers use static and dynamic analysis tools and fuzzers (an automated testing technique which involves invalid, unexpected, or random data as inputs to a program) (Chen, et al., 2018), (Amin, et al., 2019). The process involves actively reviewing the system for possible weaknesses, technical deficiencies, or vulnerabilities. The review is conducted from the point of view of a possible attacker and may involve exploiting the discovered system vulnerabilities.

Dynamic vulnerability testing is performed with web scanners that automate the process of examining the security of web applications and conduct large-scale security analyses on many different web applications (Alsaleh, Alomar, Alshreef, Alarifi, & Al-Salman, 2017). While static code analysis or personal interaction-based scanning techniques

are more effective for detecting security defects (e.g., SQL vulnerability), the use of these techniques is limited compared to the widespread application of automated dynamic security testing tools. One of the reasons that contributed to the limited adoption of static code analysis by software developers is that it cannot be performed without accessing the source code of the evaluated web-based application. That is why the dynamic testing for vulnerability is much more popular (Bird & Kim, 2014).

Testing methods are traditionally divided into testing by the black and white box method. These two methods actually differ in terms of the aspect that the software test engineer uses when designing test cases. Analysing a program by running it with different inputs and without using or knowing the source code is called black box testing. The white box approach, on the other hand, means analysing the source code and understanding the design of the program.

Generally, penetration testers for the black box approach only know the domain or IP address of the server. Before starting the analysis, first the location is determined. Black box testing tools are common for penetration testing where one wishes to gather more information about the examined web application by fuzzing the application parameters. Such information can help later in the vulnerability discovery process (Seitz, 2014). Different testing methods using the black box method are as follows (Desikan & Ramesh, 2007):

- Random Testing - faults can be recorded with the use of a set of random or incorrect input data.
- Research testing – the tester has a free hand in planning tests. The tester is expected to continually improve the tests by learning.
- Traceability Matrix - represents a document that checks whether the requirements are also reflected in the detailed architecture.
- All Pair Testing - the method for each pair of input parameters tests all possible combinations of these parameters.
- Boundary Value Analysis - a design technique for black box testing used to test boundary values, as the input values close to the boundary have a higher chance of error.
- Equivalence Partitioning - based on the division of input data into sections that can be used to derive tests which reduces the required time for the testing.

One of the advantages of black box testing is that the designer and tester work independently. The tester does not need to know any specific programming languages to test the reliability and functionality of an application. Another advantage is that tests are performed from the tester's point-of-view and not that of the designer. There are also some disadvantages, i.e., the testing of every possible input is not possible, results might be overestimated, and it cannot be used for testing complex segments of code.

White box testing is when testers fully know the infrastructure of the system they are testing. They have access to internal IP addresses, network diagrams, source code, and other useful information. This type of testing is used when time is essence and the budget and number of allowed hours is limited. Such testing is the least realistic demonstration of what a potential attacker can do. The testing methods according to the white box method are (Desikan & Ramesh, 2007):

- Static Testing - checking and validating the code and algorithms, especially checking for syntactic faults.

- Mutation Testing - in this method, a piece of code is intentionally changed and then the response of the entire system is checked. It is often used to discover the best coding techniques to be used when expanding application software.

- Fault Insertion - allows the improvement of the code coverage by introducing faults into code testing.

- Unit Testing – this is often the first type of testing done on an application by using tests that execute all sentences in the code at least once.

- API Testing - testing applications by using public or private API.

- Testing for Memory Leaks - memory leaks are the leading cause of applications running slowly. A tester who is experienced at detecting memory leaks is essential in cases where a slow running application is the tested system.

There are some advantages of using the white box approach. One of them is code optimization by finding hidden code errors. White box tests can be automated; code paths are usually covered, and testing can start early even if the Graphical User Interface (GUI) is not available. There are also some disadvantages. The white box can be complex and very expensive. White box testing is time consuming and requires professional resources with a good understanding of programming languages and implementation.

White box and black box testing are necessary for the successful software delivery even if a hundred percent testing is not possible in either of the cases. Both methods are successful in ensuring the proper functioning of the system, but only the white box method can confirm that the implementation process was correct. In some cases, it is possible to get the correct test results with the wrong process. This phenomenon is known as random correctness, which is not always detected by using the black box strategy.

In the past, some researchers – Schagen et al., Goethem et al. and Nappa et al. have undertaken to detect vulnerabilities with the black box approach which is considered illegal without the prior permission of the site owners (Schagen, Koning, Bos, & Giuffrida, 2018), (Goethem, Chen, Nikiforkais, Desmet, & Joosen, 2014), (Nappa, Rafique, Caballero, & Gu, 2014).

For a better understanding, the differences between both approaches, between black box and white box testing, are presented in Table 5.

Table 5: A comparison of black box and white box testing.

	Black Box Testing	**White Box Testing**
Base of Testing	Testing is based on the external environment; internal behaviour is unknown.	Internal functioning is known
Programming. Knowledge	Not required	Required
Implementation Knowledge	Not required	Complete understanding
Automation	Tough to automate	Easy to automate

Time	Not time consuming	Exhaustive and time-consuming
Algorithm Test	Not the best method for algorithm testing	Best suited for algorithm tests
Code Access	Not required	Requires code access
Skill Level	Low skilled testers can test	An expert tester with vast experience
Testing Method	Based on the trial-and-error method	Data domain and internal boundaries can be tested
Granularity	Low	High
Tested by	End user, developer, and tester	Usually done by tester and developers

2.4 Crawling Methods

Crawlers are Internet bots for scanning, viewing, indexing, and downloading various websites. They have become popular due to the rapid growth of the Internet and decentralized web design. A large amount of information is stored on the Internet, which can change daily or even more frequently, and at the same time is spread on many independent servers around the world. By far the most important and most valuable use of web crawlers is the web scanning through millions of websites for the needs of search engines. Another popular purpose of using web crawlers is data mining. Data mining is the process of finding information within a large amount of data. It is one of the analytical ways to find correlations between huge amounts of unorganized data, usually using special algorithms. However, increasingly popular are also web crawlers that scan for particular data (Khan, et al., 2014).

The crawler finds the links of every visited website and arranges them in a queue for later processing. Web crawlers can be developed in several ways with implemented different functionalities and properties. However, they can be divided into two main groups according to their functionality. The first group are recursive crawlers, where every time when a new link is discovered, the crawler immediately visits the link and returns to the initial link after visiting the rest of the links. Recursion is a software technique in which a program or method calls itself. In some cases, using the recursion method is a good choice, as the code implementation is much simpler than in the case of iterative implementation. It is suitable for less demanding crawlers and for detecting a small scale of websites. The problem is that the entire program is stored in a stack and each time a call is issued, the memory is filled if the website has a lot of web links to visit (Heaton, 2006).

Another group are the iterative crawlers, where the crawler stores newly discovered links in a list of links and visits them later. Implementing a crawler as iterative code eliminates the problem of recursive execution of the scan as the process is not limited to a recursive stack, and the program can run in parallel. This drastically speeds up the operation of the crawler as each website can be scanned in a separate process. In addition to the queue of pending links, the keeping of an error list was enabled; the list contains websites with errors arising from various reasons, such as server errors, temporary inaccessibility of websites, insufficient permissions, etc. By using a list of error links, repeated crawling can be avoided and websites that are not accessible can be revisited later (Heaton, 2006).

Web crawlers can send GET or POST HTTP requests. After receiving the request, the web server returns one of the predefined status codes according to which the crawler needs to respond appropriately. Responses are divided into the following groups (W3, 2019):

- 1xx - all codes starting with 1 are only informative statuses
- 2xx – the request was successfully processed. In response, an information about the page is received from the server and the entire content of the website, if GET or POST request were sent.
- 3xx - the website was not successfully downloaded because it was moved,
- 4xx - these errors represent various errors made by the client:
    - 400 – the request could not be understood, incorrect address,
    - 401 – client access without authorization,
    - 403 – the server understands the request but refuses access to the client,
    - 404 – the website does not exist,
    - 408 – the client did not send the entire request to the website in a timely manner,
    - 414 – the address of the requested page was too long,
    - 415 – unsupported files,
- 5xx - codes indicating server errors. The most common are:
    - 500 - internal server error,
    - 503 - server not available,
    - 505 - HTTP version not supported.

If the server returns the code from the 2xx group to the web crawler, the crawler continues to transfer, process and save the found website to the database. If an error occurs, then the current website is saved to the error list. Web links and all processed data are stored in an SQL database, which is designed to store and edit data in a type of relational database.

In addition to the recursive and iterative crawling methods, there are also so-called single web crawlers and multiple thread or multi process crawlers. A single thread crawler is the simplest web crawler method where the logic is executed on one computer. Considering its simplicity, it is also the most inefficient crawler, at least in terms of the page crawling speed. The flow of execution takes place in a single thread. It is especially useful for specific tasks that are limited to a small scale of websites, as website indexing is not used. The number of websites that are visited in a given amount of time is very small (Zaccone, 2015). Web crawling is increasingly utilized for many projects in different fields (e.g., scraping data for machine learning models, etc.). In most projects, Python programming language is used to help scrape data from websites.

Both single and multiple thread crawlers in Python run on a single computer. Threading can be used when an application has a network bound, or multiprocessing, when there is a Central Processing Unit (CPU) bound (Summerfield, 2013). Even though by nature Python is a linear language, the threading module can be useful when more processing power is needed. While threading in Python cannot be used for parallel CPU computation, it can be used for input or output (I/O) operations such as web crawling as the processor is sitting idle waiting for data. Threading is important since many scripts related to network input or output data spend the majority of time waiting for data from a remote source. When crawling scattered websites, the processor can download data from different websites in parallel and combine the result at the end. Certain parts of crawler code cannot be executed

in parallel, especially those sections of the program to which all threads access. These are usually the parameters used by the function within a single thread. Even though these functions can be performed in parallel, irregularities in the operation of the program can occur. Python programs have also trouble with system specifications because of the Global Interpreter Lock (GIL). The GIL is a lock that allows only one thread to hold control over the Python interpreter (Summerfield, 2013). In other words, only one thread can execute at any point in time. Python was not designed for computers that might have more than one core, therefore the lock is necessary as Python is not thread-safe and there is a globally enforced lock when accessing a Python object. Also, the lock is needed when there is a use or modification of variables between threads. Whenever a function wants to modify a variable, it locks that variable. Therefore, when another function wants to use the variable, it has to wait until that variable is unlocked. The lock ensures that one function can access the variable, perform calculations, and write back to that variable before another function can access the same variable. By over-synchronizing methods and using lock, a program can degrade performance. If the whole program is in a critical section, the use of multithreading would lose its significance, as only one thread can run. Therefore, multithreading may not always be the most useful Python functionality and may even result in worse performance depending on what should be achieved (Zaccone, 2015).

On the other hand, there is multiprocessing, which enables the functionality of programs that can be run in parallel (bypassing the GIL) and use the entirety of your CPU core (Zaccone, 2015). Though it is fundamentally different from the threading library, the syntax is quite similar as shown in Table 6.

Table 6: Threading and Multiprocessing comparison.

Python Multithreading	**Python Multiprocessing**

import thread
 	
# Create new threads 	
thread1 = myThread("Thread", 1)
thread2 = myThread("Thread", 2)
 	
# Start new Threads 	
thread1.start()
thread2.start()
 	
# Add threads to thread list 	
threads.append(thread1)
threads.append(thread2)
 	
# Wait for all threads to complete 	
for thread in threads:
  thread.join()
	from the multiprocessing import Pool
	
	p = Pool(64)
	p.map(scan, checkList)
	p.terminate()
	p.join()

Threads are created of a single process for increasing computing power	CPUs are added for increasing computing power
Threads of a process are executed simultaneously	Processes are executed simultaneously
A common address space is shared by all the threads	Every process owns a separate address space
Each thread runs parallel to each other	Multiprocessing improves the reliability of the system

System takes a moderate amount of time for job processing	System takes less time for job processing
The creation of a thread is economical in time and resource	The creation of a process is slow and resource-specific

A tool known as spider uses a scanning technique that detects web vulnerabilities by using multiple processes, including crawling web applications, downloading content, and discovering general vulnerabilities. There are two commonly used techniques for vulnerability assessment. The first is a passive vulnerability assessment which performs unobtrusive scans and simply scans a website to determine whether it is vulnerable. These techniques can be understood as bumping into a door, but not touching it to see if it is open or closed. On the other hand, passive vulnerability assessment techniques aim to cross-reference application characteristics such as specific software versions, plug-in versions, etc. with databases of known vulnerabilities (Samtani, Yu, Zhu, Patton, & Chen, 2016).

In contrast to passive vulnerability assessment, active scanning is a simulated attack on a website in order to access vulnerabilities. Examples of such techniques include port scanning, checking SQL injections and HTML injections, brute forcing, monitoring network traffic, and dropping malicious or exploitative payloads. Unlike the passive vulnerability assessment, various open source and business tools are available for active assessment. These tools generally focus on specific tasks (Samtani, Yu, Zhu, Patton, & Chen, 2016).

2.5 Web Vulnerability Notifications

Detecting vulnerabilities on websites has become more common in the recent years. For researchers and attackers, detecting such problems is often done by effectively looking for existing security issues based on a list of domains or hosts. For example, the zero-day vulnerability allows an attacker to upload malicious files on a site within a week of the vulnerability becoming publicly known. Thus, the security community must focus not only on detecting vulnerabilities, but also on identifying effective ways for informing the web owners of vulnerable websites. Therefore, a big challenge for researchers in revealing a web vulnerability is to establish contact with the appropriate person or administrator of the vulnerable website (Cimpanu, 2020). There are several ways that researchers generally use to obtain a direct communication channel that leads to the responsible contact:

- One option commonly used is to scan the frontpage while scanning the website and search for contact email addresses or phone numbers. Alternatively, searching the entire domain to retrieve email addresses is used.

- Another commonly used approach is standardized email aliases. Request for Comments (RFC) 2142 suggests redirection for each domain to the appropriate mailboxes and use to contact the organization's staff. The organization's domain name must be valid. For example, if the ISP's domain name is domain.com, the address <abuse@domain.com> must be valid and supported. Mailbox names must be recognized independently of uppercase and lowercase letters, for example POSTMASTER, postmaster, Postmaster, PostMaster and even PoStMaStEr must be treated the same, with delivery to the same mailbox. RFC also suggests the use of the remaining pseudonyms suitable for our goal. In addition to pseudonyms related to security@ and abuse@, they can be contacted by the service support mailbox that include the generic info@ pseudonym.

- WHOIS domain information can be also used to query for information about registered domain names. The WHOIS database stores personal and contact information of all domain owners, as well as information on the registrars through whom these domains were registered. The WHOIS record of an individual domain also contains information on when the domain was registered and when its registration expires. The record sometimes also contains administrative and technical contacts, which in most cases belong to the domain registrar.

- One of the notification options is also through CERT, the national cyber security response centre. It coordinates incident resolution, provides technical advice on intrusions, computer infections, alerts about network issues and raises the general public awareness about current threats on electronic networks.

Although emails are a convenient way to inform users about a vulnerability information, there is a privacy issue that can arise if the obtained information is used in an unethical manner, i.e., if a crawler collects user email addresses or other sensitive information from websites and adds them to the spam list. Due to abuses in the past, it is now rare for e-mail addresses to be listed on websites, as many spammers manage to obtain e-mail addresses by scanning the entire Internet and scrapping email addresses. However, if these contacts are listed, they are usually in protected form. As a result, this process is unreliable and error-prone, and complicated by anti-scratch mechanisms such as CAPTCHA or blurred email addresses. Therefore, such contacts are not feasible for extensive notifications and are no longer used. Contact forms are becoming more common, but unfortunately interacting with custom web forms in a large-scale scenario is also not a viable option.

Depending on the server provided, however, the structure and content of the information provided varies. WHOIS providers limit the number of requests and partially use CAPTCHA to protect contact addresses. Therefore, the WHOIS query is not feasible on a large-scale. There are otherwise online services that store WHOIS data of all registered domains on the Internet. Unfortunately, the prices of these databases are dizzying and almost inaccessible to researchers, so in most cases this method of notification is not exactly useful, especially if there are several million domains.

CERT can act as an intermediary to report software vulnerabilities to operators. They typically have the technical infrastructure and procedures to communicate vulnerability information to trustees within their remit. The countries in which vulnerable web applications are hosted can been selected for the selection of CERTs. This process can be time consuming and requires multiple connections between different organizations, which can cause noise and success effect.

Chapter 3

Overview of the Studies for Detection of Web Vulnerabilities

Chapter 3 provides a detailed overview of existing scanning tools and methods. It also provides an extensive overview of the work of other researchers. The chapter concludes with comparison of different vulnerability scanning methods.

3.1 Existing Web Scanning Tools and Methods

The rapid growth of the Internet in recent years has brought many benefits to modern society in terms of communication and information exchange. In addition, new and complex issues arise due to the flexibility, openness and network integration systems. Unfortunately, vulnerabilities on the Internet can be affected not only by virtual environments in a solitary way, but it can also have serious consequences in the real world. Therefore, identifying new system vulnerabilities represents important information for web server and website owners.

In this context, an overview of the various publicly available security scanning tools is provided. Specific security scanning tools, presented in Figure 5, are classified into two groups according to their functionality. This chapter of the thesis describes their main characteristics and thoroughly discusses their advantages and disadvantages. The first group consists of tools that allow security scanning supporting automatic large-scale scans. Nowadays, there are many tools available that enable the identification of system vulnerabilities by scanning the entire IPv4 address space. Well-known tools from this group are Shodan (Shodan, 2019), Censys (Censys, 2019), ZoomEye (ZoomEye, 2020) and IVRE (Lalet, 2020). Tools from the first group automatically scan the internet using port scanners such as Masscan, ZMap, Nmap or their own solutions and then publish their results. Their original purpose was well-intentioned, namely to detect open ports and vulnerabilities in systems for faster and easier troubleshooting. Although these tools can detect open ports on web servers and review the services running on those ports, unfortunately they are not designed to scan website vulnerabilities.

The second group consists of personal interaction-based scanning tools that support detailed vulnerability scans. The main examples of such tools are: Nessus (Nessus, 2019), Skipfish (Zalewski, 2020), WPScan (WPScan, 2019) and OWASP ZAP (OWASP ZAP, 2021). These tools are used to perform web vulnerability identification and other scanning activities with the black box approach. They carry out a recursive crawl and dictionary-based probes. Some of these tools are also specialized in obtaining services that run on open ports and list identified network vulnerabilities. A tool from this second group (IVRE)

uses the aforementioned tools (ZMap, Masscan and Nmap) for port scanning. However, they do not enable a web vulnerability collection at large-scale.

Given that some tools (Shodan, Censys, IVRE and Nessus) from both groups are port scanners, first the port scanning tools are presented. The main purpose of port scanning is to gather information about the services offered from network-connected hosts. This is done by sending probe messages to the target hosts and getting the subsequent response. As the name suggests, port scanning is focused on the transport layer port and is therefore mostly based on the Transmission Control Protocol (TCP).

The entire IPv4 address range means 4 billion IP addresses. Even though this seems huge, the entire IPv4 Internet can be completely scanned for one Transmission Control Protocol (TCP) port on a gigabit link (i.e., port 22 – Secure Shell (SSH)) in less than an hour (Durumeric, Wustrow, & Halderman, 2013). This kind of scanning is useful for collecting statistics on the technologies used in the world, or to estimate the percentage of services that are open to the outside. Prior to the appearance of network scanning tools, it took longer to inspect the entire Internet. Legitimate research of Internet was not available, while attackers were scanning through stolen network machines, e.g., larger botnets. Botnets are interconnected with logically connected Internet devices, such as computers, cell phones, or other IoT devices. Their security has been removed and is managed by a third party, an unauthorized person. Attackers, as well as researchers, are now embarking on these new tools, as they are used to carry out malicious scans, while researchers conduct legal academic research. Durumeric et al. have found that researchers and attackers are both taking advantage of new scanning tools (Durumeric, Bailey, & Halderman, 2014).

A publicly available tool is a well-known port scanning command-line utility called Nmap (Nmap, 2020) which is well-suited for scanning small ranges and selective hosts. It was originally developed by Lyon "Fyodor" Gordon in 1997, and with the increasing functional complexity, development was later taken over by the user community. Besides port scanning, it has an extensive set of features, including host discovery, detection of running services and operating systems, as well as vulnerability identification of the running services. Nmap has also its own custom scripting engine (Nmap Script Engine) and many pre-made scripts. Nmap is based on a combination of Synchronize (SYN) scanning and banner grabbing method to collect additional IP information. Generally, after discovered ports with SYN scan, a banner grab is used to obtain protocol-specific data. Nmap requires multiple machines and weeks to complete any horizontal scan of the public address space, making it rather slow.

In recent years, some improvement to Internet-wide scanning was achieved with tools such as ZMap and Masscan. ZMap was developed by the University of Michigan. It was released in August 2013 and, since then, has received considerable usage by other academic researchers.

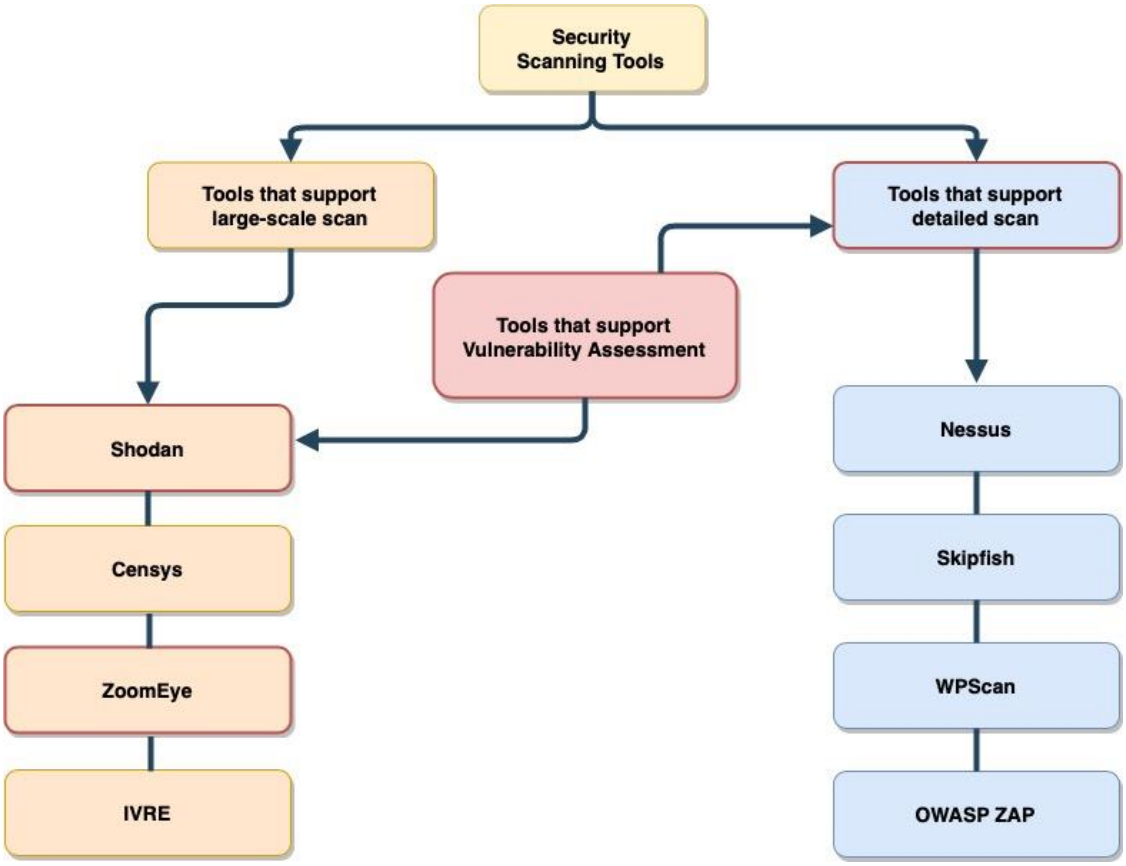

Figure 5: Security scanning tools.

The reason for this is that ZMap promises to be able to scan entire public IPv4 address range in under 45 minutes by using a mid-range machine and a gigabyte link. ZMap is SYN scan based where the SYN scan probe is sent to the target. It can be used to scan ports based on its data collection and uses distributed scan using multiple scanners and a scheduler. Since scan performance depends on the network bandwidth, probe randomization is used by ZMap in order to avoid bandwidth saturation. Using a random order drastically decreases the probability of simultaneously sending probes to hosts which belong to the same network. ZMap also does not keep the state in order to match responses to requests. Instead, sending and receiving packets happens independently, which in turn increases the overall performance. It also uses a well-known ping mechanism, which is applied in the detection step to scan the full range of IPv4 addresses.

Masscan on the other hand is an internet port scanner that can scan entire the IPv4 range in under 6 minutes and uses a custom TCP/IP stack in order to achieve this. It was developed by the security researcher Robert David Graham, with a release in 2013 (Graham & Johnson, 2014). It produces an output similar to Nmap and employs asynchronous transmission. Comparing to Nmap it is more flexible, by allowing arbitrary address ranges and port ranges. Masscan uses the same asynchronous model for sending and receiving between threads as ZMap does. Like ZMap, Masscan uses probe randomization, which aims to distribute IP addresses on the fly while scanning the entire web space.

The tools in the first group, designed for large-scale security scans, do not include web vulnerability assessments like the tools in the second group, with the exception of Shodan and ZoomEye, which allows at least a service vulnerability detection based on the version of the services listening on a particular port. A similar functionality is provided by Nmap using the extension which enables specific service vulnerability assessment. Unfortunately,

Nmap does not provide a score system for quantifying the website vulnerability level and is not capable to perform large-scale scans. On the other hand, the tools in the second group provide detailed web security analyses but are not designed for large-scale security scans. Therefore, there is no ideal tool which would cover both aspects.

3.1.1 Automatic Large-Scale Scanning

Smart devices, light bulbs, blinds, thermostats, voice assistants, and smart machines are used in households, businesses, and other industrial environments. The development of "smartness" in devices is often the victim; due to low-cost investments into the technology used it introduces new security risks and vulnerabilities.

Shodan is the most well-known search engine used on the Internet, created as an IoT search engine, launched in 2009 by John Materley. It collects information from about 500 million connected devices and services each month day and night, including traffic lights, cameras, home automation devices and other systems which are connected to the Internet. At first, Shodan inspected only ports such as port 21 File Transfer Protocol (FTP), port 22 (SSH), port 23 (Telnet) and port 80 (HTTP), but later started to cover more additional services. Shodan offers not only device indexing, but also provides an exploit database, a raw data visualization tool and an enumeration module built into the Metasploit exploitation framework (Bodenheim, 2014). Shodan is available as an online service that tries to identify which service is running on a specific port and thus providing specific information about the services, the status of the port, headers, operating systems, and so on. An example of a Shodan report for a specific IP is provided in Figure 6.

Shodan's workflow is based on a combination of SYN scanning and banner grabbing methods to collect additional IP information similar to other port scanning tools. In particular, Shodan uses their own algorithm (not publicly disclosed) that automatically reconstructs service names and versions from service banners returned by Shodan. The procedure employs mapping of all Common Platform Enumeration (CPE) names extracted from the National Vulnerability Database (NVD).

Shodan is a decent tool for discovering open ports of various services. One of the important functionalities commonly used by security researchers is that Shodan provides the vulnerability information of specific running services. It also helps researchers to discover open ports and find unprotected databases. There are also some disadvantages i.e., Shodan is not open source and its code implementation is not available. Given that the algorithm of the functioning of Shodan is not publicly disclosed, it is not known at what speed Shodan scans the entire Internet. Although Shodan scans open ports on web servers, it does not provide a scan for web applications; and due to its public availability, it can be misused by attackers. Despite the fact that Shodan provides a vulnerability assessment for a specific running service, it does not provide a score system for quantifying the website or web server vulnerability level.

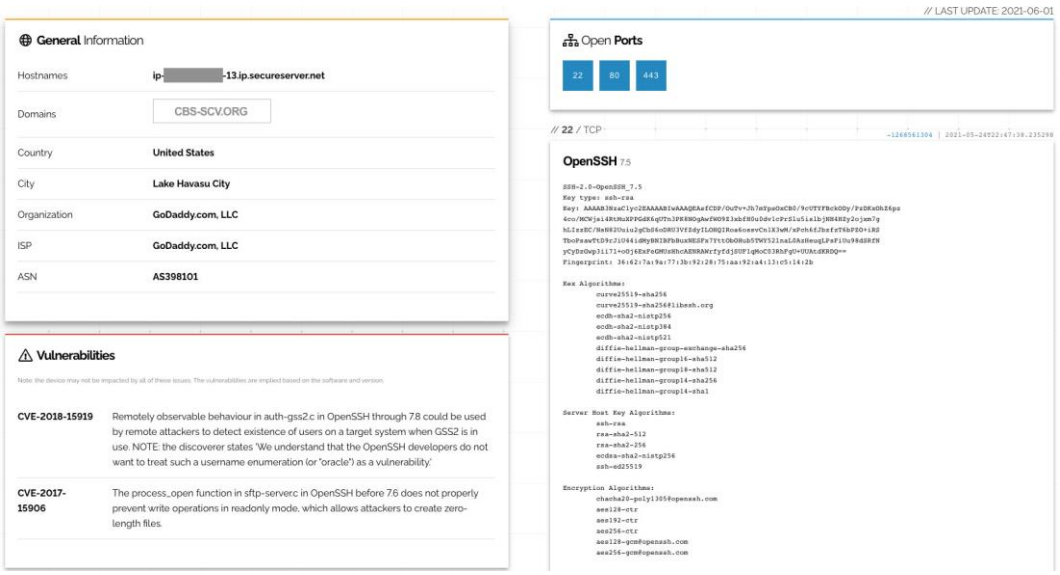

Figure 6: Vulnerability information of specific running services.

Censys is another search engine which can help to answer the question whether your device is exposed and accessible for someone else. It was designed by Zakir Durumeric in 2015 (Durumeric, Bailey, & Halderman, 2014). By means of the Censys tool, it is possible to obtain a description of devices, technical details, encryption, and certificates.

The engine collects data with ZMap by scanning more than 3 billion IPv4 addresses and allows scientists to analyse the data through scripts that can be requested with the help of a SQL engine (Arnaert, Bertrand, & Boudaoud, 2016).

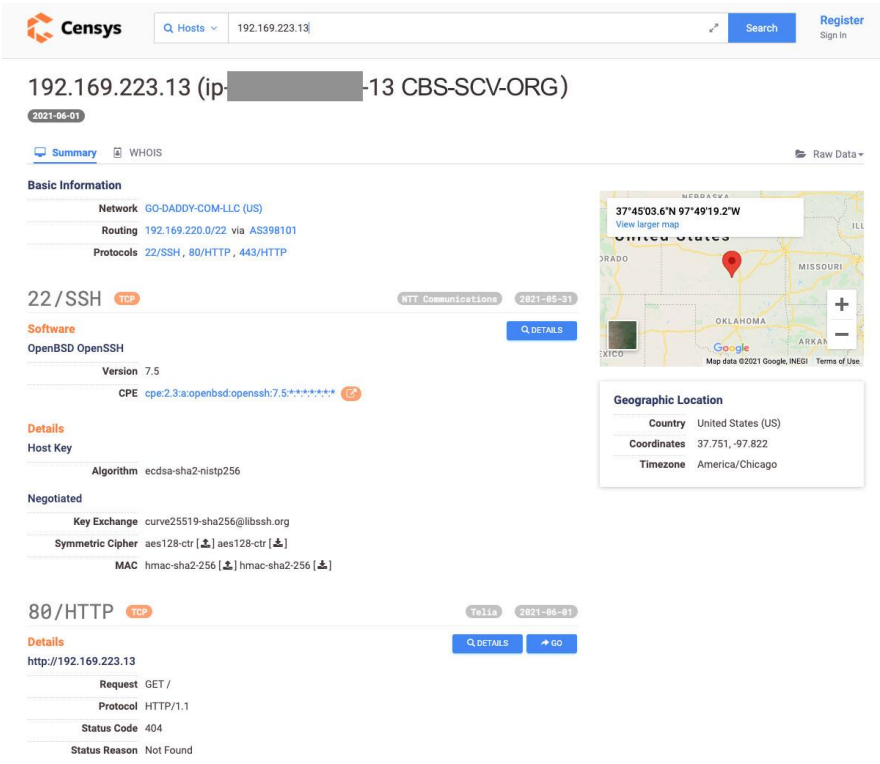

Figure 7: Information on specific running services by Censys.

Censys is similar to Shodan and allows users to retrieve information about devices and networks on the Internet. Similarly to Shodan, it collects data on hosts and websites through periodical and horizontal scans of the IPv4 address space with its own port scanning tools ZMap and Zgrab (Durumeric, Adrian, Mirian, Bailey, & Halderman). The Censys workflow is based on a combination of SYN scanning and banner grabbing methods to collect additional IP information similar to other port scanning tools. All collected data is stored in a database that is publicly accessible. An example of a Censys report for a specific IP is provided in Figure 7.

The research published by Durumeric et al. describes a detailed overview of the Censys search engine's abilities (Durumeric, Bailey, & Halderman, 2014). Moreover, the authors provide the information about its architecture and give a short comparison with the Shodan search engine, which is "the closest work to Censys", according to the authors (Durumeric, Bailey, & Halderman, 2014).

Like with Shodan, the advantage of Censys is in discovering the open ports of various services. It provides dedicated querying for hosts and certificates. Censys has some disadvantages too. It does not support the Internet Protocol v6 (IPv6). Unlike Shodan, Censys does not provide a vulnerability analysis of individual services running on open ports and does not provide a score system for quantifying website or webserver vulnerability levels. Although Censys scans open ports on web servers, it does not provide a vulnerability scan for the web applications; and due to public availability of data, they can be misused by the attackers.

ZoomEye is another IoT search engine for devices and vulnerabilities, created by the Chinese company Knowsec Inc in 2013. It is an alternative or complementary tool to the previously described Shodan and Censys. An example of a ZoomEye report for specific running services is shown in Figure 8.

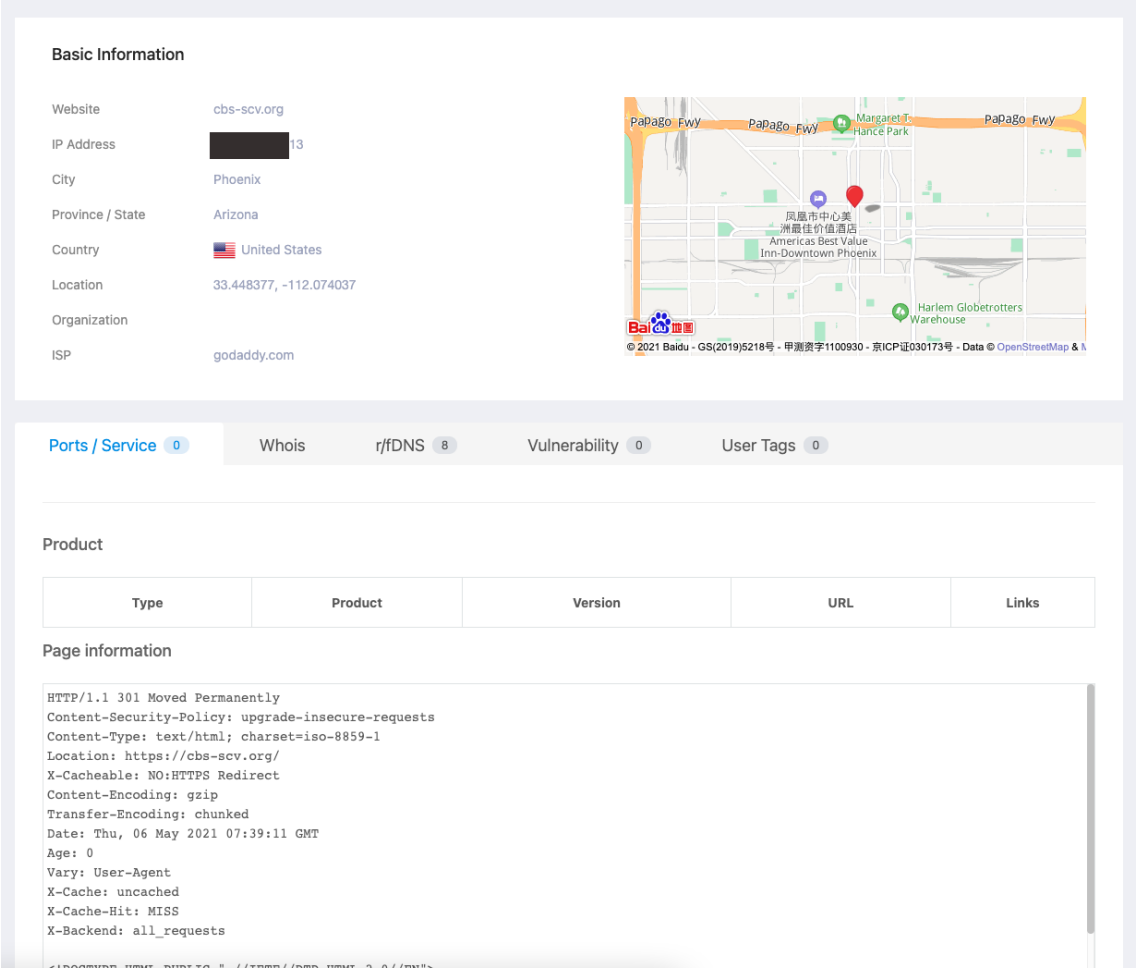

Figure 8: Vulnerability information on specific running services by ZoomEye.

ZoomEye uses its own developed tools - Xmap (host scanner) and Wmap (web scanner), where the search engine collects information from public devices and web services including fingerprint analysis. The search includes a large number of criteria (ports, services, OS, applications) with details for each host. ZoomEye's workflow is based on a combination of SYN scanning and banner grabbing methods to collect additional IP information similar to other port scanning tools. The procedure employs mapping of all CPE names extracted from NVD.

The search results also list possible identified vulnerabilities. Despite the fact that ZoomEye provides a vulnerability assessment for a specific running service, it does not provide a score system for quantifying the website or web server vulnerability level. Like Shodan and Censys, ZoomEye too does not provide a vulnerability scan for web applications; and due to public availability of data, they can be misused by attackers.

IVRE (Instrument de veille sur les réseaux extérieurs) or DRUNK (Dynamic Recon for UNKnown networks) is a network intelligence framework that contains tools for passive scanning (Lalet, 2020). It is an open-source framework and relies on port scanning tools like ZMap to output data about Internet-connected devices. IVRE also supports the ability to import XML data from Nmap and Masscan. Its main purpose is to recognise network traffic by performing Internet-wide scanning and collecting information to explore potential vulnerabilities.

The IVRE framework allows both active and passive data gathering by performing three main steps. IVRE's workflow is based on a combination of SYN scanning and banner

grabbing methods to collect additional IP information similar to other port scanning tools. The procedure employs mapping of all CPE names extracted from the NVD.

One of the advantages of IVRE is an open-source framework which provides a similar functionality as Shodan, Censys and ZoomEye. The disadvantage of IVRE is that it requires a manual setup installation and is very time consuming. Another disadvantage is its relying on external dependencies such as ZMap, Masscan, Nmap etc. Also, the speed may vary depending on the selected external scan tool. Although IVRE scans open ports on web servers same as Shodan, Censys, and ZoomEye, it does not provide a vulnerability scan for web applications. Despite the fact that IVRE provides a vulnerability assessment for a specific running service, it does not provide a vulnerability scan for web applications and a score system for quantifying the website or web server vulnerability level.

According to Arnaert et al., the following concerns arise while using these search engines: the results can be too numerous to be proficiently understood; they can be irrelevant (i.e., outdated, non-specific, incomplete, etc.) (Arnaert, Bertrand, & Boudaoud, 2016). Moreover, it might be a challenge for non-security experts to interpret the queries and results, as the understanding requires vital knowledge of different syntaxes in order to interact with the search engine (Arnaert, Bertrand, & Boudaoud, 2016).

3.1.2 Interaction-Based and Detailed Scan

To facilitate the work of specialists, there are programs that reduce the total time spent for searching the vulnerabilities, by assessing the security features of the systems. Such programs are called security scanners. Scanners detect security vulnerabilities on a specific remote target or local computer. Some of them are able to issue recommendations to eliminate the identified vulnerabilities.

Nessus is a popular vulnerability scanner from Tenable, Inc. It is intended for a comprehensive scan of local and network computers with possible vulnerabilities. During the scan, ports are scanned with a search for any shortcomings that could be exploited for intrusion. It allows testing for the system's resistance to Denial-of-Service (DoS) attacks and testing applications that use secure socket communication. Nessus also supports a web application testing to detect previously unknown web application vulnerabilities (Beale, Meer, van der Walt, & Deraison, 2004).

Nessus is based on a client-server architecture. The client is a user interface where the scan mode, plug-ins, and scan targets are configured. The client sends commands to the server and at the end of the scan it generates a report with the results where the identified security flaws are listed.

One of the advantages of Nessus is that it provides a vulnerability assessment of web applications and a score system for quantifying the website vulnerability level. In addition to the advantages, there are also disadvantages, i.e., network vulnerability scanners like Nessus serve an important purpose in a security testing program, however they do not provide a complete vulnerability scan, as they mostly focus on network security. Network level testing is definitely required, but it does not provide a deep analysis of the web application security. Nessus also uses black box methods to detect web application vulnerabilities such as SQL injection, XSS and remote file disclosure. Besides that, it is limited to target only scan and it cannot perform Internet-wide scans, as it has time consuming scanning methods.

In addition to all the aforementioned tools, **Skipfish** is another tool that differs from the others mainly in its design as it detects vulnerabilities in web applications. It is an active tool for security monitoring of web applications written by Michal Zalewski (Zalewski, 2020). With recursive crawling and dictionary-based probes, it prepares an interactive website map. The resulting map is marked with the output of a number of

active security checks. The final report generated by the tool serves as a basis for expert assessments of the security of web applications.

Similar to Nessus, Skipfish allows web developers to scan their web application for possible security flaws. Skipfish can be used to determine if a code is vulnerable to common attacks such as SQL injection, XSS, XML / XPath injection and blind vectors. Skipfish is fast and can manage many requests per second.

The advantage of Skipfish is its extensive reporting and the ability to scan web applications. It is free and open source. The disadvantage of Skipfish is its black box approach and its limitation to scan only one target –it cannot perform Internet-wide scans. Another disadvantage of Skipfish is that it does not provide a CPE/CVE vulnerability assessment, nor it does not provide a score system for quantifying the website vulnerability level.

WPScan is a semi free and open-source WP website vulnerability scanner written for security professionals and blog maintainers to test the security of their websites. It functions partly based on the black box approach (Uygur, 2014). WPScan allows users to examine installed plug-ins and themes on the specified target website. If a plug-in is found, then the internal WPScan database (DB) is checked for vulnerabilities. Additionally, the version of WordPress is determined and the vulnerabilities known for that version are displayed.

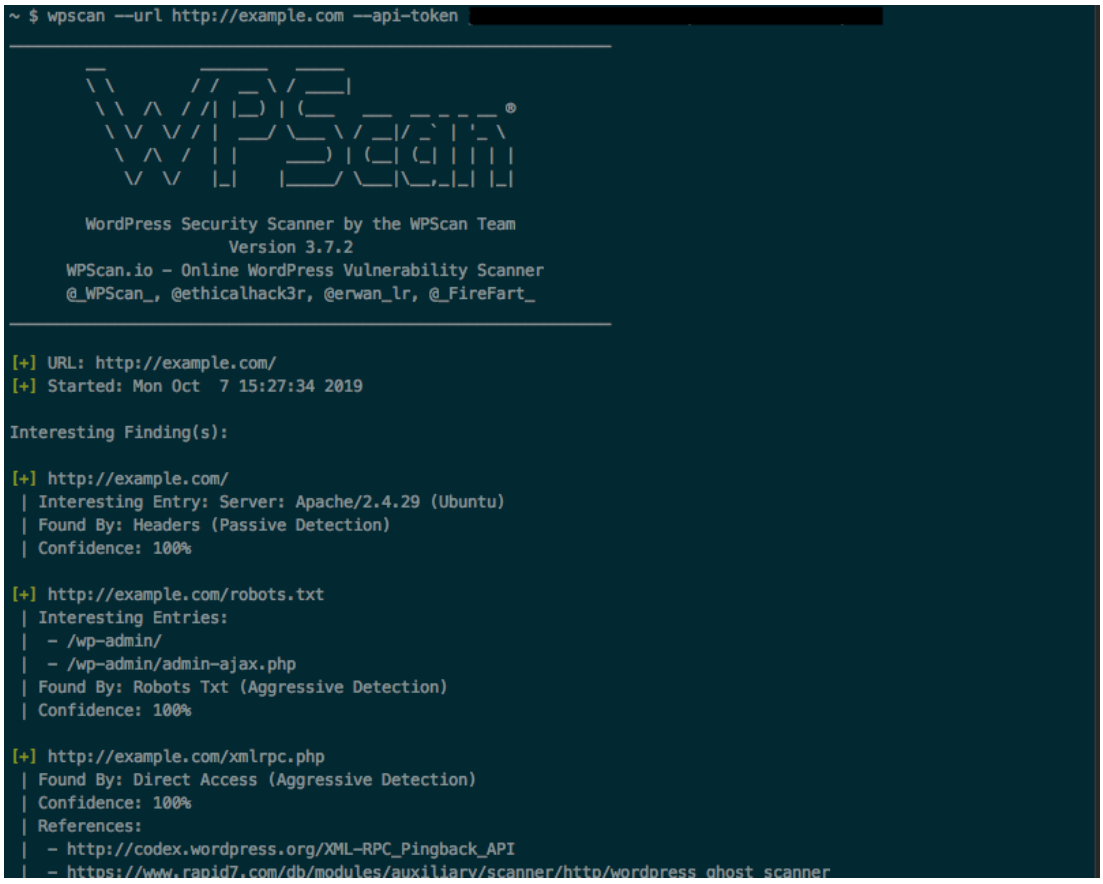

Figure 9: WPScan scan results.

The advantage of WPScan is definitely that it is made for a specific software and allows thorough reviews of the core of the website, as well as plug-ins and themes. Nevertheless, it also has some very important shortcomings. One of them is its scan limitation. Namely, it does not allow scanning of the entire Internet, but only one website at a time. In addition,

it is time consuming and has a limited number of scans per day. Although WPScan allows vulnerability detection, it does not allow risk assessment as it does not provide a vulnerability score for an individual vulnerability or a score system for quantifying the website vulnerability level. As shown in Figure 9, WPScan is accessible only the through command line and does not provide any web interface to examine the results.

OWASP ZAP (Zed Attack Proxy) or zaproxy is a web application vulnerability detection tool, part of the OWASP project (OWASP ZAP, 2021). It is considered as one of the better freely available test tools for detecting common vulnerabilities by using the black box approach. For advanced testing of the application, the plug-in needs to be loaded into the browser, which allows ZAP to record all requests and responses between client and server. It offers the functionality for a variety of competencies – from developers and testers to security testing professionals. ZAP provides an overview of the software code and looks for the flaws such as backdoors or other malicious code that gives attackers access to a website or sensitive data.

ZAP advantage provides extensive reporting and the ability to scan web applications. The disadvantage of ZAP is its black box approach and its limitation of scanning only one target – it cannot perform Internet-wide scans. Another disadvantage of OWASP ZAP is that it does not provide a CPE/CVE vulnerability assessment, nor it does provide a score system for quantifying the website vulnerability level.

3.1.3 Vulnerability Scanning in Past Studies

Several researchers have conducted various security studies by developing their own tools or by using and adapting existing ones to allow vulnerability scanning. Nevertheless, each of the existing or newly created tools faces a lack of features needed to cover all the necessary aspects. They can be divided into two groups.

The first group performs scanning of the space of IPv4 addresses for a specifically defined subject area, such as hosted services, Secure Sockets Layer (SSL)/Transport Layer Security (TLS), vulnerabilities or specific software or protocol vulnerabilities by using mass scan tools, such as ZMap, Nmap and Massscan. This technique is good as a fast TCP/IP stack-fingerprinting technique to identify the OS type, port range scan, and basic web, but not for a detailed overview of the online vulnerabilities. However, there are researchers who, in addition to scanning ports, use tools such as Zmap or Masscan to scan web application vulnerabilities (Schagen, Koning, Bos, & Giuffrida, 2018), (Nappa, Rafique, Caballero, & Gu, 2014), and (Kim, Kim, & Jang, 2018).

Even though automatic scanning tools from the first group can be customized to obtain the source code from a specific IP address, they do not allow users to scan co-hosted domain names. Admittedly, ZMap and Masscan can scan a huge number of IPv4 addresses, but access to the IPv4 address gives us direct access to only one website hosted on this IPv4 address and not the rest that could be located at this same IP address. Generally, there are several domains hosted on one IPv4 address, and these tools do not have a solution for this type of scanning. The original purpose of these tools is to scan open ports and not to collect web vulnerabilities at a large scale. Another major problem that occurs when using a ZMap or Masscan is that there is a high probability that there is some kind of barrier on the server visited that does not allow port checking. Even if allowed, there is a high probability of mass complaints to the abuse departments, as web server owners are not satisfied if their servers are inspected by such tools, which makes these tools unethical. Another disadvantage of these tools is that it does not provide a CPE/CVE vulnerability assessment, nor does it provide a score system for quantifying the website vulnerability levels.

The second group of studies involves the application of methods to identify vulnerabilities in more detail on a limited set of web servers and does not support scanning on a large scale (Goethem, Chen, Nikiforkais, Desmet, & Joosen, 2014), (Stock, Pellegrino, Li, Backes, & Rossow, 2018), (Vasek, Wadleigh, & Moore, 2015). The tools vary in their characteristics, such as types of inspected objects, types of detected vulnerabilities and crawler coverage.

Most often researchers from the second group look just to the selected websites from the Alexa database (Alexa, 2019), which ranks millions of websites in the order of popularity, with an Alexa Rank of 1 being the most popular (Doupé, Cavedon, Kruegel, & Vigna, 2012). They focus on the black box approach (unethical), determining whether a given input propagates, rather than efficiently finding the propagating inputs, for arbitrary vulnerabilities (Schagen, Koning, Bos, & Giuffrida, 2018).

An example of such an approach can be found in the research by the iMinds–Distrinet group of KU. They have reported the results collected after the analysis of a sample consisting of 22,000 websites from 28 European countries, but only 1,100 entries from the most popular websites were used in the study. For each EU member state, they have selected the top 1000 websites ending with the corresponding Country Code Top-Level Domain (ccTLD) from Alexa's list of the top 1 million websites. They have obtained up to 200 websites which were then visited by mimicking the behaviour of a regular visitor (Goethem, Chen, Nikiforkais, Desmet, & Joosen, 2014), (Chen, Desmet, Huygens, & Joosen, 2016). The study was focused on two mechanisms: the presence of HTTPS and a mechanism that enables XSS mitigation. Their scan was time consuming since it took them five days to scan all the websites. Despite the slowness and plug-in ignorance, they reviewed only the core versions.

Nguyen and Hwang have carried out a similar analysis of 5,000 top-ranked Alexa websites looking for XSS and other vulnerabilities (Nguyen & Hwang, 2016). In a similar research by Son and Shmatikov, the top 10,000 Alexa websites were reviewed (Son & Shmatikov, 2013). They found that websites were exploitable to several attacks, including XSS and content injection due to the lack of proper checks in the cross-origin communication mechanisms (Son & Shmatikov, 2013). Another similar approach presented by Stock et al. involves a small scale of websites (24,000), where they mainly focused on XSS vulnerabilities in WordPress by comparing the file hashes (Stock, Pellegrino, Li, Backes, & Rossow, 2018).

In 2014, a tool called Wasapy was developed and used as a black box scanner (Akrout, Alata, Kaaniche, & Nicomette, 2014). The search for vulnerabilities with this tool was focused on detecting the code-injection vulnerability. From their finding it follows that although various vulnerability-testing systems exist, either on the market or as freely available software, including the academic attempts, a fully automated tool for a dynamic vulnerability measurement of a WCMS across the entire Internet is not yet available (Akrout, Alata, Kaaniche, & Nicomette, 2014).

Even though certain tools presented in the second group allow the detection of web vulnerabilities, they are time consuming and do not allow large-scale searches. This group of studies is forgetting the rest of the web, as they are focusing on an isolated specific vulnerability, such as XSS, SSL, SQL injection, phishing, Heartbleed and search-redirection attacks, instead of covering all of them at once. In addition, their methods are also time-consuming: they need more than 9 days to measure a dataset of 200,000 websites (Vasek, Wadleigh, & Moore, 2015). This would mean that even in the best-case scenario, the scanning of Alexa's Top 1 million websites would take over 45 days. Also running a vulnerability scan with aggressive assessment (black box approach), with the exception of (Vasek, Wadleigh, & Moore, 2015), against millions of websites would be unethical. Another disadvantage of these tools is that they do not provide a CPE/CVE vulnerability

assessment, nor do they provide a score system for quantifying the website vulnerability level.

A comparison of the functionalities of some publicly available scanning tools is presented below in Table 7. The properties of the tools were chosen to facilitate comparison and highlight the diversity of tools and to make it clear that a tool that would combine most of the features of existing tools was missing.

The table shows the choice between the passive and the aggressive assessment approach where an aggressive scan can result in the host or server becoming unresponsive. During an aggressive assessment, an attack on the network can be simulated, meaning it will detect the vulnerabilities that a potential attacker would notice. On the other hand, a passive assessment approach identifies operating systems or applications without simulating attacks.

Passive scanners can provide information about weaknesses, but cannot take action to solve security problems. These scanners can check the current software version, indicating which devices are using software that represents a potential gateway for attackers. The CPE / CVE feature is used to denote scanning tools capable of reporting about vulnerabilities. In addition to the CPE / CVE feature, the score system property is used to denote the scanning tools capable of quantifying the website vulnerability level. A recognized web application feature is used to denote if a scanning tool is capable of recognizing web applications and their vulnerabilities. The ethical feature is used to denote if the scanning tool performs scanning in an ethical manner or not. Namely, that scanning tools do not apply an aggressive approach to review website vulnerabilities. Finally, the scan speed feature is used to denote the speed of the scanning tool and the last large-scale scan feature is used to denote whether the tool is capable of large-scale scanning. Large-scale scanning covers both a global scan of IP addresses and domain names.

Table 7: A comparison of some selected scanning tools.

Tool	Aggressive assessment	Passive assessment	Automated CPE / CVE	Score system	Recognized web app.	Ethical	Scan Speed	Large-scale scan
Nmap	•	• •	• • •	• •	•	•	•	•
ZMap	•	• •	•	•	•	•	• • •	• •
Masscan	•	• •	•	•	•	•	• • •	• •
Shodan	•	• •	• •	•	•	•	• • •	• •
Censys	•	• •	•	•	•	•	• • •	• •
ZoomEye	•	• •	• •	•	•	•	• • •	• •
IVRE	•	• •	• •	•	•	•	• • •	• •
Nessus	• • •	•	• • •	• • •	• •	•	•	•
Skipfish	• • •	•	•	•	• • •	•	•	•
WPScan	• • •	• •	• •	•	• •	• •	•	•
OWASP ZAP	• • •	•	•	•	• • •	•	•	•
(Goethem, Chen, Nikiforkais, Desmet, & Joosen, 2014)	• • •	•	•	• •	• •	•	•	•
(Stock, Pellegrino, Li, Backes, & Rossow, 2018)	• •	•	•	•	•	• •	•	•
(Vasek, Wadleigh, & Moore, 2015)	•	• • •	•	•	• •	• • •	•	•

'• • •' is used to denote a strong support, '• •' to denote a moderate support, and '•' for a weak or no support (i.e., unavailable) of a specific feature. CPE, Common Platform Enumeration, CVE, Common Vulnerability and Exposure.

As presented in Table 7, many tools are mutually exclusive in terms of functionality. Even some functionalities are not available with most tools, such as vulnerability assessment, web application recognition, and ethical approach. In addition, most tools do not support collecting vulnerabilities in web applications on a large-scale, on the other hand, the tools that offer this functionality are time consuming and do not allow large-scale scanning. Namely, tools that enable large-scale scanning are tools that scan IP addresses and forget about domain names. Therefore, tools that allow large-scale scanning have a moderated support marked in Table 7.

Several studies have found that, currently, there are not many large analyses that evaluate the security features as well as the vulnerability in a broad range of websites. The reason for that is the enormous number of active websites on the Internet (Goethem, Chen, Nikiforkais, Desmet, & Joosen, 2014).

The review of tools in this chapter makes it clear that there is no optional / optimal solution for efficient large-scale website vulnerability scanning.

Running regular web vulnerability scanners against numerous websites is time consuming and, if exploit techniques are used, the scan is considered illegal if a scanning permission is not granted by the owners. Another way to legalize scanning is to detect the vulnerability without breaking the law.

Therefore, a better approach would be to detect the application and then identify its issuing version or its fingerprint and then look into a database with identified vulnerabilities of that particular version. The most well-known vulnerability database is the NVD that is hosted by the National Institute of Standards and Technology (NIST, 2019). There are some studies of researchers who have suggested similar solutions (Durumeric, Bailey, & Halderman, 2014), (Bou-Harb, Debbabi, & Assi, 2013), (Chiba, Tobe, Mori, & Goto, 2012). Ghaleb in his work "Website fingerprinting as a cybercrime investigation model: Role and challenges" stresses the need for development of hybrid approaches that combine fingerprinting practises with other anti-cybercrime methods for complementing each other (Ghaleb, 2015). Additionally, he recommends to make such techniques automatic in order to create a complete fingerprinting system that ISPs can use against cybercrimes (Ghaleb, 2015).

3.1.4 Web Vulnerability Notifications Studies and Findings

The Internet is a growing space with a variety of software deployed in many countries around the world. This heterogeneous structure, however, is reduced to a homogeneous way of addressing servers, i.e., their IP address. Thus, a more extensive vulnerability detection is becoming more feasible and the need to find effective large-scale notification mechanisms to inform affected parties increases too, and many researchers are focusing on an extensive detection of many types of errors. As mentioned in Chapter 2, researchers, CERTs, security companies, and other organizations with vulnerability data have a number of options for identifying, networking, and communicating with actors responsible for the affected system or service. On the other hand, the persons responsible for managing these services are as diverse as the Internet architecture itself: whether it is spoken languages or knowledge of the technical details of the services. It may be impossible to identify the appropriate recipient of the notice, the recipient may not trust it, may overlook or ignore it, or may misunderstand it. Notification of vulnerable services has long been considered a

side note in research. Lately, the community has been focusing more not only on vulnerability detection, but also on informing the affected parties.

Research conducted by Li et al. and Stock et al. investigates the feasibility of extensive online vulnerability notices (Li, et al., 2016), (Stock, Pellegrino, Li, Backes, & Rossow, 2018). In both studies, the effectiveness of various communication channels, including WHOIS email contacts and CERTs, was experimentally evaluated. In addition, they analysed the reachability and behaviour of their messages. Both results are largely consistent and provide a complementary study of notifications in a separate context. They also observed that although the notifications may have resulted in a statistically significant increase in corrections, the raw effect was small. Each of these studies showed that notifications can increase vulnerability repair and clean-up. In the study provided by Li et al., they tried to find an understanding of the factors influencing corrections and how to compile effective vulnerability notifications (Li, et al., 2016). They conducted an extensive study to inform thousands of network operators about security issues present in their networks, with the aim of identifying which aspects of notifications have the greatest impact on the performance. Their study explored vulnerabilities covering a range of protocols. The controlled multivariate experiments examined the impact of the choice of the client to be contacted, the detail of the messages, the hosting of the website to which the message is linked, and the translation of the message into the local language of the notified party. Vulnerable systems were monitored for several weeks to determine their recovery rate in response to changes. They also assessed the outcome of the e-mail sending process itself, feelings and perspectives expressed in user responses. Responses were mostly positive, with 96% of email responses expressing favourable or neutral feelings (Li, et al., 2016).

Their findings therefore indicate that notifications can have a significant positive effect on the correction. When users got notified, an additional 11% of contacts resolved the security issue. Repeated notifications did not further improve the remediation. In the light of these positive but unsatisfactory results, they encourage the security community to examine the notifications in depth and to establish standards and best practices that promote their effectiveness.

Stock et al. used various methods mentioned in Chapter 2 and found that while their notifications had a statistically significant impact on vulnerability remediation, the overall correction rate was unsatisfactory, making 74.5% of websites useful after one-month trial (Stock, Pellegrino, Rossow, Johns, & Backes, 2016). This raises the question of possible reasons for a large part of unfixed websites. The main cause could be the unresolved challenge of contacting people who can make the repair, such as developers or administrators. As they reported of all the contacts, only 5.8% viewed the vulnerability report. 40% out of these identified the vulnerability within one week. This shows that the main problem is actually the dissemination of vulnerability information. They also emphasize that the identification of contact points remains a major challenge for the Internet society, including network providers, and CERTs. They believe that this problem could be addressed with centralized contact databases, more effective strategies for disseminating information between hosts / CERTs, or even with a new notification channel or a trusted website responsible for such notifications. Further they claim that until a solution to the reachability problem is found, the effects of large-scale notifications are likely to remain low in the future.

Another approach presented by Soussi et al. in their work “Feasibility of Large-Scale Vulnerability Notifications after GDPR (General Data Protection Regulation)” consider the problem of effective notifications of domain owners, administrators, or webmasters about domain abuse or vulnerabilities (Soussi, Korczynski, Maroofi, & Duda, 2020). They developed a scanner to test whether selected email aliases specified in RFC 2142 are

correctly configured and whether notifications can be successfully delivered. They also tested the reachability of email addresses collected from the DNS Start of Authority (SOA) records. Based on a measurement campaign of a large number of domains compared to the previous studies (4 602 907 domains), they proved that domains are more reachable through SOA contacts. Also, they found that the country-code TLD names are better reachable compared to the new generic top-level domain (gTLD) names. In their study, they also observed that the most used generic email alias is abuse (67.95%). Their results confirm that direct notification channels are currently not scalable, so they propose a scheme that preserves user privacy in compliance with GDPR and supports large-scale vulnerability notifications (Soussi, Korczynski, Maroofi, & Duda, 2020).

Chapter 4

Design and Construction of the Method

This chapter proposes a novel method for large-scale detection of vulnerable websites. The method is presented in detail, including a signature database construction, and scanning algorithms. A scoring mechanism section presents the key ideas behind the developed approach for vulnerability identification and scoring. Further, the implementation of the tool is discussed and the tool evaluated.

4.1 Applied Basic Process for Vulnerability Detection

Many tools (Nessus, Skipfish, WPScan, OWASP ZAP) and research approaches presented in the previous chapter use black box scanning methods, which are designed to assess the vulnerability of an individual application (Schagen, Koning, Bos, & Giuffrida, 2018), (Son & Shmatikov, 2013), (Nguyen & Hwang, 2016), (Akrout, Alata, Kaaniche, & Nicomette, 2014). These methods which are mainly designed to scan a single application are unsuitable for large-scale scanning of the Internet. The use of these tools requires additional time for development of an application to perform the scanning and processing a list of target websites. The scan time for such an experiment is difficult to predict, as adding multiple simultaneous scanners would result in a fairly rapid reduction of the scan speed, due to network bandwidth limitations and computing power. Regarding this type of scanning, there are also ethical issues, not to mention legal ones, as running a black-box scanner against an unknown website can reduce its web performance or affect it in some other way as they are usually based on performing attacks. On the other hand, there are tools (Zmap, Masscan, IVRE) and research approaches presented in the previous chapter that enable large-scale scanning but do not support detailed scans of web applications. Therefore, based on these findings, a need appeared to improve the existing methods and add new algorithms for a better and faster vulnerability assessment based on large-scale scanning while still following ethical principles.

The methodology for designing and implementing a tool is a process consisting of the development of algorithms for solving one or more sets of problems. The overall process for developing algorithms is presented as a chart in Figure 10. The scanning algorithm takes care of discovering new websites and collecting data about the accessed WP website. In the second step, the obtained data about the core and the plug-ins on the website are compared with the data in the vulnerability databases including risk score assessment. This comparison provides information whether there is a vulnerability in the server or not. In the event where a simple comparison is not good enough to match the detected software

version regarding vulnerability, a more sophisticated algorithm needs to be implemented to ensure a better match between the discovered and existing data.

In addition to known vulnerabilities in the web server, an attack can be successful for other reasons due to inappropriate security policies or password practices. However, the WPCMS platform and aligned plug-ins are mainly at risk due to the presence of vulnerabilities. This is justified with the evidence presented in Chapter 2 and 3 about the large number of exploits targeting the WPCMS that contribute to the high compromise rates. Secondly, most of the data or information within the web content page cannot be directly observed, such as default usernames and passwords, the server or website configuration, and many other security flaws. To get this information, one needs to attack or carry out other actions that exploit the presence of vulnerabilities within websites. Therefore, in the context of this thesis, the secure and insecure parameters represent only the vulnerabilities discovered by the developed WPCMS-specific method for collecting data about vulnerabilities. A comprehensive analysis of the website and the server running the website are not included.

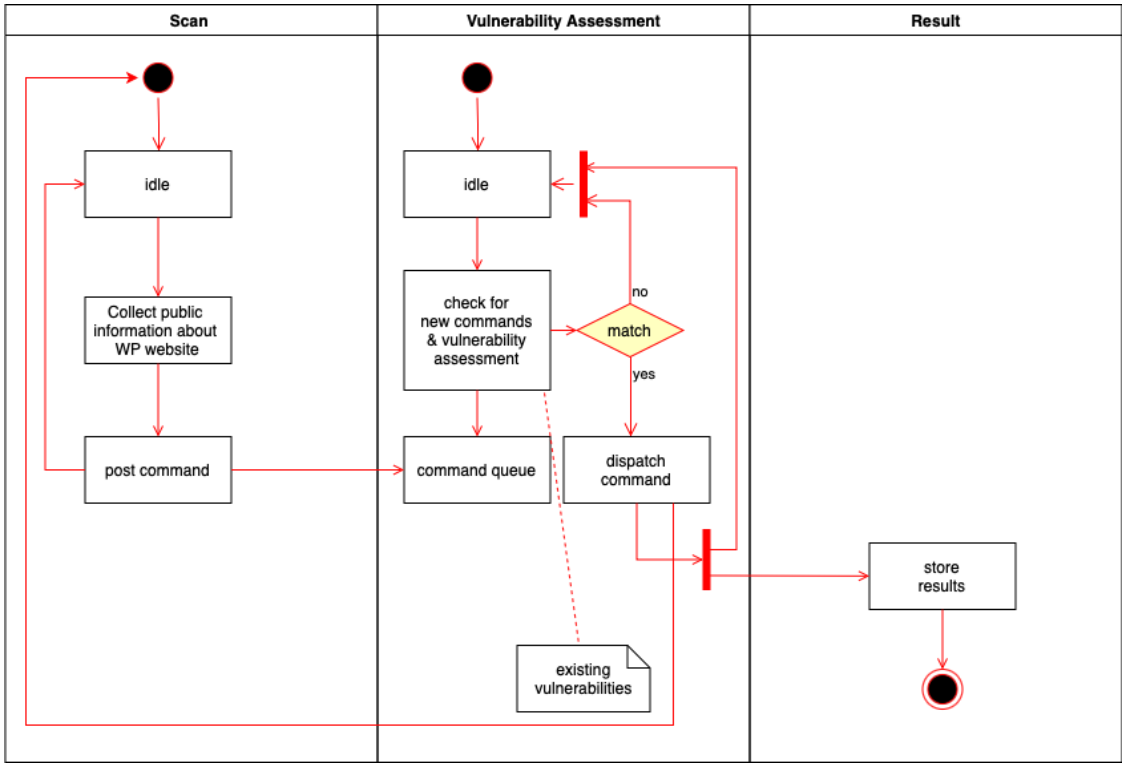

Figure 10: The general process of the algorithms developed.

4.1.1 Data Collection Method

Data is collected by applying a crawler mechanism. The crawler mechanism needs to be fast and should reduce the likelihood of false access. This is done by providing feedback information from the crawler about accessed websites and matching these data with the data within the temporary database that holds information about the websites visited previously. Since there is no list of WP websites on the Internet, the first step in the search process is to scan the entire web space and find the websites built with WPCMS. In the beginning of the scan process, websites from the list of root domains created from the available Domain Name Server (DNS) zone files are accessed. In the context of the use of the list of root zone domains as seed URLs, the search method can be accelerated. To speed

up the process, all services that allow the use of shortened URLs, and large websites such as Facebook, YouTube, and Instagram, are removed from the search list as they are not running on the WP platform.

After identifying the presence of a WPCMS application, the query part of the tool looks for the presence of a file named "robots.txt". This file informs the visitor whether the website allows browsing or not. Reviewing applications with the tool is legal if browsing is not forbidden. The next step is sending queries to the website for collecting information about the content. When scanning the Internet's web servers, the tool identifies also whether the accessed server is hosting several WCMS websites (e.g., known as a virtual web server, hosting URLs) and analyses and assesses the vulnerabilities in all present websites. This functionality is enabled with the in-built Algorithm 1 presented below.

Algorithm 1: Finding websites and their neighbours

```
procedure CrawlerProcess(queue)
    while queue > 0 do
        website <- queue.nextURL()
        url <- website.URL()
        if website.permitsCrawl() then
            content <- retrieveURL(url)

            // store Domain info
            // ipv4, asinfo, city, countrycode, ISP, latitude, longitude,
            // organization, region, timezone, zip number, server type

            storeWebsite(content, url)
            for each url in parse(content) do
                // find domain neighbours on the same server address
                neighbours <- url.getNeighbours()
                if neighbours > 0 then
                    for each n in neighbours do
                        // add new domains into the queue
                        if n not in queue and n not like socialLinks()
                        then

                            queue.addURL(n)
                        end
                        if
                    end
                    for

                end if
                // add new domains into the queue
                if url not in queue and url not like socialLinks() then
                    queue.addURL(url)
                end
                if
            end for

            If website.wordpress() then
                // check wordpress core and plug-ins vulnerabilities
                website.identifyVulnerabilities(content, url)
```

```
                else
                        website.addToIgnoreList()
                end if
            end if
            queue.releaseWebsite(website)
        end while
end procedure
```

The vulnerabilities found on the accessed website are inspected for further information. The Algorithm 2 presented below is analysing the vulnerability details. The core version is inspected first. Information about the WP core version is matched with the data within the local pre-pre-pared signature database with stored known core version vulnerabilities. The retrieved signature record contains a c_secure element that returns a true or false, depending on the WP core version. If the core is vulnerable, the tool returns true, otherwise false. This is followed by the c_score element, which stores the score level of vulnerability found in the local signature database, and the signatureID, which is a unique identification number for each WP core version. In case of detected plug-ins, the plug-in version is matched with data from the local database where all known vulnerabilities are obtained. The signature record contains the p_secure element, which returns true or false depending if the plug-in is vulnerable or not. The p_score from the database represents the score level of vulnerability and signatureID as the unique identifier of this plug-in. The domain name of the WP website and identified plug-in vulnerability are then stored in the database. If there is more than one plug-in present in the website, the highest score from the present plug-ins is allocated.

Algorithm 2: Vulnerabilities identification

```
procedure identifyVulnerabilities(content, url)

    // check wordpress core version
    domainID <- url.getDomainID()
    wordpress_version <- parseCoreVersion(content, domainID)

    if wordpress_version then

        // retrieve signatureID record with identified vulnerabilities
        c_secure, c_score, signatureID <- checkVuln(core, wordpress_version)

        // store in database WordPress core version and score
        storeCore(wordpress_version, signatureID, c_secure, c_score, domainID)
    end if

    if hasPlug-ins(content) then
        max_score <- 0

        // parse all the plug-ins available on the website
        for each plug-in in parsePlug-ins(content) do
            p_version <- plug-in.Version()

            p_secure, p_score, signatureID <- checkVuln (plug-in, p_version)

            // store in database Plug-in version and score
            storePlugin(plug-in, signatureID, p_secure, p_score, domainID)
```

```
                    if max_score < p_score then
                        max_score <- p_score
                    end if
                end for
            end if
        end procedure
```

To match the core version and the plug-in in the signature database, the Algorithm 3 is applied, where the detected slug of the plug-in and the version are provided. The slug (plug-in identifier) can be used as a unique name to identify the path to the plug-in on the website or as a unique identifier on the official WP website where plug-ins are listed. Depending on the given slug and version, a search is performed by reviewing the entire signature database for a particular slug and version. If there is a match, the probability of the match is calculated, and the results returned back to the scanning method. A successful match of the detected core or plug-in version within the plug-in and core version in the signature database allows an instant identification of all known vulnerabilities for an individual core version or plug-in.

Algorithm 3: Find the best match in a signature database

```
procedure checkVuln(slug, slugVersion)

        // slug <- 'wysija-newsletters'
        // slugSearch <- 'wysija + newsletters'
        // find match between parsed plug-in and plug-in in local database

        procedure calculateMatch(slug, signature)
            return 1 / (abs(length(slug) – length(signature.slug)) + 1)
        end procedure
        for (signature, version) in database do
            if version.version equals slugVersion then
                match <- 0
                if signature.slug equals slugSearch then
                    match <- calculateMatch(slug, signature)
                end if

                results.add([version, signature, match])
            end if
        end for
    end procedure
```

4.1.2 The Scoring Mechanism

In reviewing the existing tools and used methods of different researchers in Chapter 3, it was found that some of them (Nmap, 2020), (Shodan, 2019), (ZoomEye, 2020), (Lalet, 2020), (Nessus, 2019), (WPScan, 2019), (Goethem, Chen, Nikiforkais, Desmet, & Joosen, 2014) can assess vulnerabilities but do not include a scoring system for quantifying the website vulnerability level. Given the dispersion of information about vulnerable web applications and exploits, it is understandable that a scoring mechanism providing an estimated score of exploitation of all WPCMS core and plug-in versions is not available. To better present and make users aware of the severity of websites' vulnerabilities, just the vulnerability information is not sufficient, since most users are unaware of the harm that

exploitation of vulnerabilities can cause. If users were aware of the impact of threats, they would become more aware of the seriousness of the vulnerabilities and would take a more serious approach to resolve and eliminate vulnerabilities on their websites. In addition, the score or impact of a potential vulnerability exploitation is additional information for a better comparison and analysis and enables a comparison of the particular country's web space vulnerability level, not only in terms of the number of vulnerabilities, but also in terms of the impact of vulnerabilities on insecurity.

A scoring method is required to index each plug-in and each version of WordPress with a specific risk score parameter for the vulnerability. For easier vulnerability assessment during crawling, the scoring mechanism was developed based on the Common Vulnerability and Exposure (CVE) database, which contains more than 300 identified vulnerabilities for the WordPress core version and more than 2,168 identified vulnerabilities for plug-ins. Similar approach to CVE and CVSS was applied in the developed scoring method that numerically defines the level of risk on a scale from 0 to 10, where 0 represents the lowest risk and 10 represents the highest risk in the case of exploitation of a particular vulnerability.

Unfortunately, the estimated score of exploitation of a particular vulnerability is not always available. Therefore, it was necessary to manually review vulnerabilities and exploits and determine the level of the impact of vulnerability exploitation based on past assessments and the type of the individual vulnerability. The vulnerabilities described in Chapter 2 were considered. To maintain and match the existing vulnerability scores of cores and plug-ins, the CVSS equation was used in the baseline (FIRST, A Complete Guide to the Common Vulnerability Scoring System, 2019).

Based on the CVSS guide (FIRST, 2020), the vulnerability score depends on its exploitability and impact. The exploitability score is calculated according to the following equation:

*Exploitability = 20 x AccessVector (AV) * AccessComplexity (AC) * Authentication (AU)*

The meaning of AV, AC, and AU is described on the next page.

Further, the impact score is calculated according to the following equation:

*Impact = 10.41 x (1 – (1 – ConfidentialityImpact (C)) * (1 – IntegrityImpact (I)) * (1 – Availability Impact (A)))*

The meaning of C, I, and A is also described on the next page.

Finally, the total base score (vulnerability score) is calculated using the following equation:

*BaseScore = roundTo1Decimal((0.6 * Impact) + (0.4 * Exploitability) – 1.5) * f(Impact)); f(Impact) = 0 if Impact = 0, 1.176*

Since the WordPress core and plug-in vulnerabilities can be exploited by remote exploitation over the Internet, the access vector (AV) such as adjacent, local, or physical, was not used to determine the attack vector score. Therefore, the default score value for AV was set to 1. The individual score values for all the components listed below are predetermined by the CVSS.

As described above, the calculation of the base score is based on the exploitability and impact metrics. The first sub-group, exploitability metrics, includes two components - access complexity (AC) and authentication (AU). The first component, access complexity, is measured by determining how easy or difficult it is to exploit an open vulnerability. The access complexity measurement was rated as low, medium, or high:

- LOW (L) – no special conditions are required to exploit the vulnerability (score 0.71).
- MEDIUM (M) – additional conditions are required for a successful attack, i.e., the vulnerable website operates with a non-default configuration (score 0.61).
- HIGH (H) – for a successful attack, the attacker must take some preparatory steps to gain access. These may include social engineering methods (score 0.35).

The second component, authentication, measures the level of privilege or access that an attacker must have before a successful exploitation. The authentication was classified as:

- NONE (N) – the attacker does not require authentication and no privileges or special access are required to perform the attack (score 0.704).
- SINGLE (S) – the attacker needs basic privileges at the "user" level to perform the exploitation. The attacker must be authenticated once (score 0.56).
- MULTIPLE (M) – the attacker needs basic privileges at the "user" level to perform the exploitation. The attacker must be authenticated once or multiple times, even if the same credentials are used each time (score 0.45).

The second sub-group, impact metrics, includes three components: confidentiality (C), integrity (I) and availability (A). The metric is used to determine the severity of vulnerability of individual components (core or plug-in) in a successful attack. In other words, what is the final negative outcome that occurs as a result of the exploitation. The first component, confidentiality, measures the impact on the confidentiality of the data processed by the system. Confidentiality refers to the disclosure of sensitive information that the exploitation of a vulnerability offers to the attacker. Confidentiality was measured using three metric values:

- NONE (N) – there is no impact on the confidentiality of the system, as the data is not accessible to unauthorized users due to exploitation of security vulnerabilities (score 0).
- PARTIAL (P) – there is a possibility of data disclosure to a limited extent. The attacker has no control over what data they can access. Therefore, not all data is available (score 0.275).
- COMPLETE (C) – there is full disclosure of information where the attacker has full access to all data, including sensitive information (score 0.660).

The second component describes the impact on the integrity of the compromised system. It measures the degree of data changes the exploitation enables. If the attacker cannot change the data, integrity is maintained. Integrity was measured at the following three levels:

- NONE (N) – integrity is maintained and the system is not compromised (score 0).
- PARTIAL (P) – there is a possibility to change some data within a limited extent. The tampered data does not have a serious impact on the whole system (score 0.275).
- COMPLETE (C) – there is a complete loss of integrity as the attacker can manipulate all data on the target system (0.660).

The third component describes the impact on system availability. System availability can be negatively impacted by attacks that consume bandwidth, processing power, or other resources. The availability was measured using the following three metrics:

- NONE (N) – there is no impact on system availability (score 0).
- PARTIAL (P) – there is a loss of some features or system performance may be affected by the attack (score 0.275).
- COMPLETE (C) – there is a complete loss of system or information availability (score 0.660).

Vulnerabilities with a higher score represent a greater threat that can have serious consequences for the website owner and enable the attacker to obtain data from the database, to access files and directories stored outside the web root directory, or even to allow full access to the server.

The accuracy of the site vulnerability assessment was improved by the insecurity score calculation that enabled the classification of each website found. The scoring mechanism starts with the identification of vulnerabilities of the WPCMS core from the data of the signature database within the local signature database built from different resources available online (CVE databases, Exploit databases etc.). If several vulnerabilities were identified in the core, the highest value of the identified vulnerabilities is selected as the website core's risk score value. The same approach is applied to the plug-ins; the plug-in with the highest vulnerability is used to calculate the overall vulnerability of the WCMS. If the version of the plug-in is not available in the CVE database, then the plug-in is stored in a separate database table for further analysis and a search for exploits.

The scoring assessment for the vulnerability of the website is based on two sets of parameters: the first set is used to assess the risk of the website's core $C_{(s)}$, and the second set is used to assess the risk of the attached plug-ins $P_{(s)}$. The version of the website's core is first verified and then the tool looks for potential vulnerabilities of the core in the signature database. If several vulnerabilities were identified, the score with the highest value is selected for the website core's risk parameter. The same approach is applied to the plug-ins: the first match is found for each of the plug-ins and then the vulnerability with the highest risk parameter is selected for the plug-in's risk value.

The calculated vulnerability risk score for the web WCMS $A_{(s)}$ is calculated as an aggregated score from the score obtained for the web-server analysis of the website's core $C_{(s)}$and the score for the attached plug-ins $max(P_{(s)})$:

$$A_{(s)} = max(C_{(s)}, max(P_{(s)}))$$

Equation 1: Vulnerability risk score calculation.

where $A_{(s)}$ is the final risk score, $max(P_{(s)})$ is the largest score among the scores for the detected plug-ins' vulnerabilities, and $C_{(s)}$ is the highest score among the vulnerabilities detected for the web server as well.

4.2 Design of the Scanning Tool

The developed tool with the implemented algorithms was named VulNet. The tool is composed of the components represented in Figure 11. They are: the crawler, which is responsible for the scanning and implements a process consisting of six steps: a) searching and collecting the IP addresses of web targets, b) accessing the web server and asking for responses for the queries sent, c) getting responses, d) sending queries for application patterns, e.g., plug-ins, and collecting information from the signature database about the found plug-ins, and f) saving and validating results with the in-built scoring mechanism. The last three steps of the tool operation differ from the currently known web scanners by the very high ability for quickly access the websites, fast processing of the collected data, high speed in inspecting the web servers and analysing and scoring the plug-in vulnerability with the presented scoring mechanism. The second component is a locally built signature database (Signature DB) from different scrapped publicly available vulnerability data or databases.

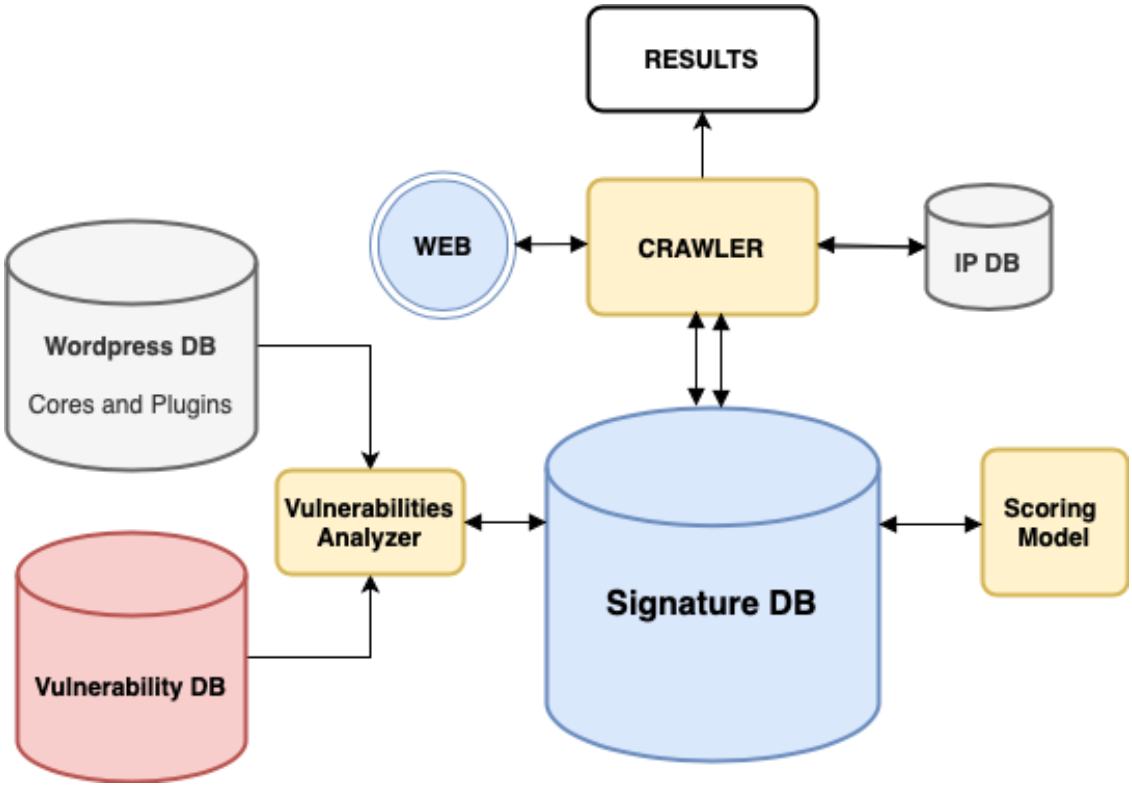

Figure 11: Developed platform components.

The implemented scanning mechanism is fast and reduces the likelihood of false access due to the feedback information. The specific scoring mechanism enables also a rapid and reliable vulnerability analysis and assessment.

For an effective detection of hosted domains (URLs) on the same server, a common crawl database is used. The Common Crawl database contains petabytes of data collected over 8 years of web crawling. The Common Crawl data is stored on Amazon Web Services' Public Datasets and on multiple academic cloud platforms across the world (Crawl, 2019). Before the start of the scan, a database of active and live domains is built within the Common Crawl database. This is done by a DNS lookup for each domain. In case the domain responded, the domain name and IP are stored in the local Common Crawl database. When a new URL is found, DNS resolves the domain and checks whether this IP exists in the local Common Crawl database. This enables automated retrieval and increases the list of new potential URLs to be examined.

In the beginning of its testing, the tool applied multithreading but it has proven ineffective. As there were many synchronization methods that use locks during the visit of a website, it was found that the efficiency of the program and especially the speed was affected. More details about the characteristics and the background of multithreading applied in the tool are given in Chapter 2.

Due to the ineffectiveness of multithreading, multiprocessing was used. The multiprocessing library gives each process its own Python interpreter and GIL. Also, it should be noticed that the scanning speed when using a multi-threaded solution or a multi-processor solution does not represent a significant difference in the processing speed if multiple cores are used. Therefore, common problems associated with threading in the developed tool, such as data corruption and deadlocks, were discarded and were no longer an issue.

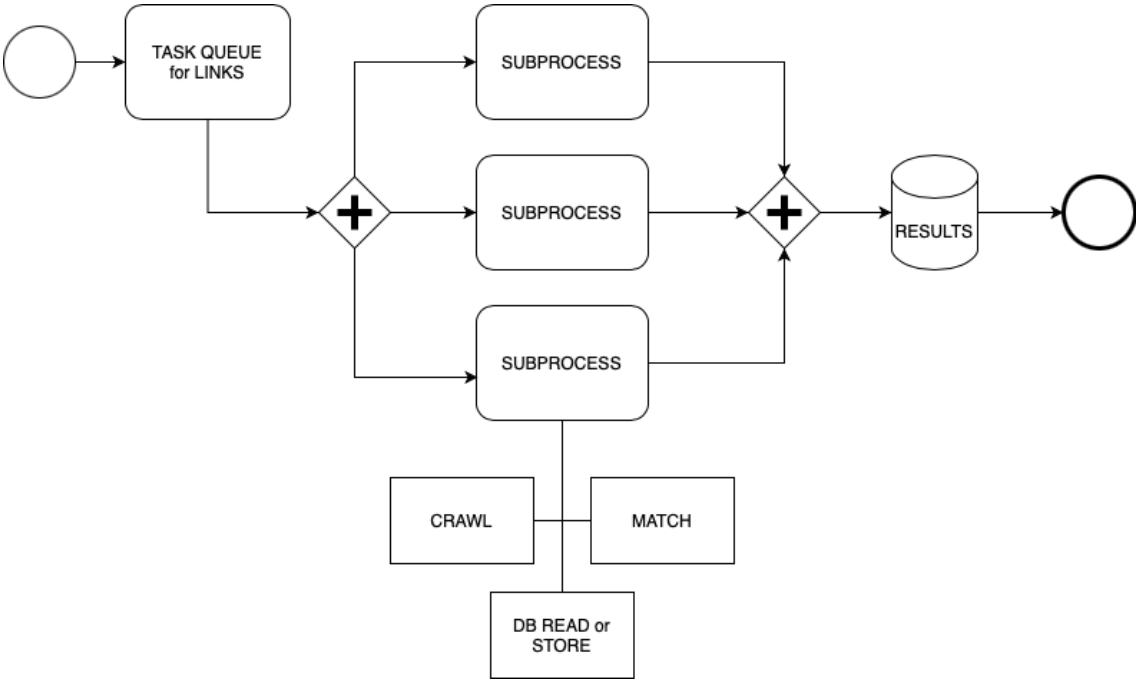

Figure 12: Multiprocessing scanning flow.

The processing speed of data collection and multiprocessing manipulation was accelerated in this way, presumably due to the GIL. I/O-bound work in both cases provides the same speed. Therefore, multiprocessing in Python was the only real way to achieve true parallelism for the developed tool. Multithreading cannot achieve this as the GIL prevents threads from running in parallel. Multiprocessing is always a useful approach when more than one thing should be done at any given time. For example, take a tool that needs to connect to 6 databases and perform a complex matrix transformation on each dataset. Putting each job in a separate thread might help a little as when one connection is idle another one could get some CPU time. However, the processing would not be done in parallel as the GIL assumes that resources are available from one CPU. By putting each job in a multiprocessing process as presented in Figure 12, the multiprocessing accelerates the scanning speed and the synchronization of methods. Therefore, the development of the tool to achieve the best performance applied the iterative scanning method and multiprocessing.

For the Crawler part a local IP WHOIS database (IP DB) was needed, from which useful information for an individual IP address or hosting provider could be obtained. Some hosting providers may already have an infrastructure in place to receive and respond to security complaints. Contacts for abuse can be found in the Regional Internet Registry (RIR) contact databases or queried via the IP WHOIS protocol. Both systems have limited rates. To avoid this limitation, a local IP database was built and was included in the tool. The WHOIS database is used mainly for the geographic location of the domains for statistical data processing. A parsing technique and some of our own python scripts were used to process IP WHOIS records from different RIRs:

- The American Registry for Internet Numbers (ARIN) is the regional Internet registry for Canada, the United States, and many Caribbean and North Atlantic islands.

- Réseaux IP Européens (RIPE, French for "European IP Networks") which uses IPs and domains in Europe, Central Asia and the Middle East
- APNIC (the Asia Pacific Network Information Centre) is the regional Internet address registry (RIR) for the Asia-Pacific region
- LACNIC (Latin America and Caribbean Network Information Centre), which serves IPs and domains in Latin America and parts of the Caribbean
- AFRINIC (African Network Information Centre) is the regional Internet registry for Africa, which serves IPs and domains on the African continent

Beside RIRs, the WHOIS Referral (RWHOIS) protocol was used too. It is a less known source and generally overlooked by researchers. RWHOIS is a directory services protocol which extends and enhances the WHOIS concept in a hierarchical and scalable way. RWHOIS was first designated by Network Solutions in 1994 within IETF (RFC 1714) and then replaced in 1997 (RFC 2167).

The collected data, such as the geolocation, Autonomous System Number (ASN), and company data were imported into the tool's local databases and are refreshed weekly. To facilitate the work of other researchers, after building the IP WHOIS local database, access to this data is free and available to all researchers using an API call that returns results in the JSON format. Custom plans can provide additional fields, more frequent updates, and other print formats. An example of the response of this API is presented below, where one can see that the request returns basic information about the IP address.

```
#~ curl https://vulnet.e5.ijs.si/api/193.138.1.1

{
   "query": "193.138.1.1",
   "status": "success",
   "hostname": "birt.e5.ijs.si",
   "city": "Ljubljana",
   "region": "Ljubljana",
   "countryCode": "SI",
   "country": "Slovenia",
   "latitude": 46.051079,
   "longitude": 14.50513,
   "currency": "EUR",
   "timezone": "Europe/Ljubljana",
   "isp": "Institut Jozef Stefan",
   "org": "ARNES",
   "as": "AS2107 Arnes",
   "route": "193.138.1.0/24"
}
```

Based on the retrieved AS (Autonomous System) number, additional information about that AS number can be obtained. As presented in the Appendix A2, upon a successful query the API returns the name of the AS owner (e.g., the acronym ARNES-NET) and the full name (Academic and Research Network of Slovenia). It also returns the website address (http://www.arnes.si/), the abuse contact addresses (abuse@arnes.si), physical address (Tehnoloski park 18, SI-1000 Ljubljana, Slovenia), traffic estimation (10-20 Gbps),

RIR allocation name (RIPE), allocation date (1992-11-06 00:00:00) and WHOIS server. In addition to all the above information, it also provides information about IPv4 or IPv6 prefixes, IPv4 or IPv6 peers, and IPv4 or IPv6 upstreams and downstreams. The example in Appendix A2 is limited, namely the entire printout is several pages long.

During the scan of each individual page, general information about the IP address for further communication with the responsible persons is stored.

4.2.1 Components of the Signature Database

The signature database is the main component of the tool, as without it, scanning in an ethical and footprint-based way would not be possible. Signature databases consist of two different databases. The first database is a WordPress DB which stores all known plug-ins associated with the WordPress (WPCMS) core versions and plug-ins available on the internet. It contains more than 430 available core versions and more than 56,000 plug-ins with all their previous version entries. WordPress core versions and plug-ins are regularly updated from the WordPress API URL. The WordPress security team often works with other security groups to resolve issues in shared dependencies, such as resolving vulnerabilities in the WP core. WordPress offers the JSON API where all plug-ins, core versions, and basic vulnerability data are on disposal. There are two API endpoints for the core versions. As presented in Table 8, the first API contains information about specific core version compatibility with PHP and MySQL.

Table 8: API response for basic core information.

API call: curl https://api.wordpress.org/core/version-check/1.7/ | json_pp

Response:


```
{
    "response" : "autoupdate",
    "current" : "5.5.2",
    "version" : "5.5.2",
    "packages" : {
      "full" : "https://downloads.wordpress.org/release/wordpress-5.5.2.zip",
      "new_bundled" : "https://downloads.wordpress.org/release/wordpress-5.5.2-new-bundled.zip",
      "no_content" : "https://downloads.wordpress.org/release/wordpress-5.5.2-no-content.zip",
    },
    "php_version" : "5.6.20",
    "download" : "https://downloads.wordpress.org/release/wordpress-5.5.2.zip",
    "mysql_version" : "5.0",
    "partial_version" : false,
    "new_bundled" : "5.3"
  }
```


The second API presented in Table 9 provides information about the core security (insecure or secure core version), update, or latest version status. Unfortunately, the API does not specify the vulnerability and score of the vulnerability for a specific core, therefore a local database of all possible vulnerabilities had to be built.

Table 9: API response for insecure and secure core versions.

API call: curl https://api.wordpress.org/core/version-check/1.7/ | json_pp

Response:


```
{
  "5.0.7" : "insecure",
  "5.0.10" : "insecure",
  "4.5.17" : "insecure",
```

```
    "4.2.26" : "insecure",
  etc.
  }
```

On the separate WP API for plug-ins, there is a list of all plug-ins, where the information about the latest version of the plug-in, its compatibility with other plug-ins, number of downloads, popularity, rating, keywords, and a list of all published versions can be found (look at the Appendix A1 for details). Similar to WP core API, WordPress does not provide vulnerability information for individual plug-ins. Figure 13 presents the data flow process from WordPress APIs into the first component of the tool, the signature database.

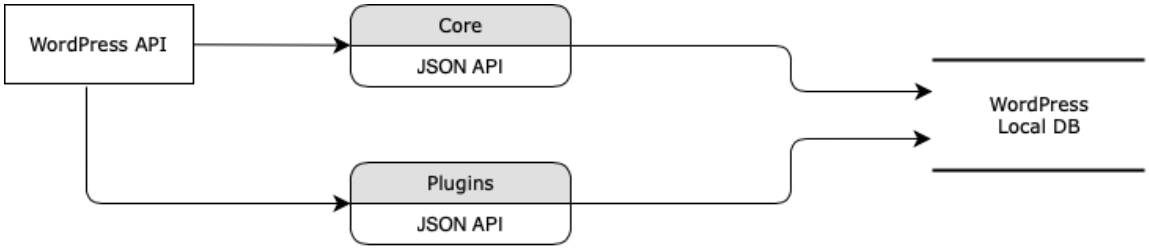

Figure 13: Data flow for the first database of the tool signature database.

The WP API does not provide vulnerability data of cores and plug-ins, yet these were necessary for the automation of the of vulnerability evaluation process and to build a vulnerability database, which is a key part of the signature database. This database was built from data found in different on-line resources such as the CVE database, the exploit database and other similar sources that contain about 2,500 unique entries for core and plug-in vulnerabilities.

These and other vulnerability databases help organizations to develop and apply patches to their systems in order to mitigate critical vulnerabilities. The WPCMS vulnerability database was built manually by examining and scraping freely available vulnerability databases on the Internet: i.e., the NVD or security websites where information about recently discovered software vulnerabilities are listed. The NVD is the U.S. government repository of standards-based vulnerability management data with the Security Content Automation Protocol (SCAP). This data enables the automation of vulnerability management, security measurement, and compliance. The NVD includes databases of security checklist references, security-related software flaws, misconfigurations, product names, and impact metrics (NIST, 2019). The NVD includes databases with safety checklists to help find vulnerabilities. It uses CVE to name the vulnerabilities and a standardized CVE-style description for each vulnerability. These vulnerabilities are assessed by means of the Common Vulnerability Assessment (CVSS) system, and used as a source for assessing the impact of software vulnerabilities.

CVE is the “de facto” standard used to uniquely identify vulnerabilities and is widely accepted. Each vulnerability in the CVE database has a unique identifier in the format CVE-YYYY-nnnn, where YYYY is the code for the year in which the vulnerability was first published in the CVE database, and nnnn is the serial number. A single CVE record can combine multiple vulnerabilities, or the same vulnerability may appear in multiple records. This can lead to a less accurate number of vulnerabilities.

There are other websites with information about vulnerabilities that use the same data as provided by NVD and combine it with other databases. One such is CVE Details, which uses an NVD resource and combines them with other additional useful resources such as

the Exploit Database (Database, 2019) and Metasploit. There is also a vulnerability database dedicated only to WordPress (WPScan database) vulnerabilities and its plug-ins, which lists all possible vulnerabilities related to a specific WP version. Unfortunately, access to this database is limited and is not free, so it is important in detecting WP vulnerabilities to build a new custom database with all possible vulnerability entries to facilitate the detection of vulnerabilities.

The vulnerability databases determine which application versions contain vulnerabilities. Therefore, the identified WP version is in most cases sufficient to detect a known vulnerability in an installed application. Discovering applications and their versions speed up the search for vulnerabilities. The tool vulnerability database consists of various data formats and databases combined into a single database. As shown in Figure 14, the main data source for compiling the second component is data from the NVD and DB exploit websites.

The second component signature database (Signature DB) from Figure 11 is the data collection from both previous components (the WordPress DB and the vulnerability DB), after a vulnerability analysis and scoring mechanism was performed.

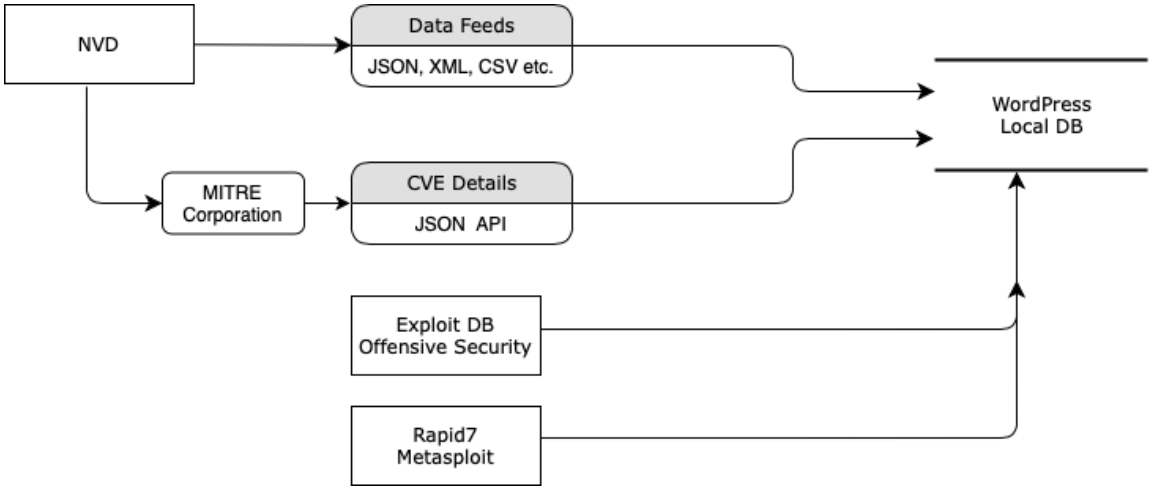

Figure 14: The vulnerability database compiled from different vulnerability sources.

4.2.2 Detection of WordPress Application Presence

The most basic elements in the semantic web are metadata. Metadata is structured information that describes, explains, or makes it easier for users to obtain or manage sources of information. Generally, websites contain metadata that describes content on the website. Even though this information is not displayed in the browser on the client side as part of the website, this data is very useful for crawlers and search optimization. Metadata is found inside the <head> section in the header of the website, where the <meta> tag is used. Among the meta tag attributes, there are the title, description, content type, author, application name, generator, etc.

The successful detection of whether a particular website is running on a specific web application varies between different tools. In the case of WordPress, there are different methods for performing checks whether a website is running on the WP platform. One of possible ways is to check the website metadata generator and a content attribute that contains WordPress meta value. Figure 15 presents the common WP website header meta data attributes from TechCrunch.com, which is ranked among Alexa's top 1000 websites in the world.

```
<!DOCTYPE html>
<html lang="en-US" class="no-js">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=Edge"/>
    <meta name="viewport" content="width=device-width">
    <meta name="viewport" content="initial-scale=1.0,width=device-width,user-scalable=no,minimum-scale=1.0,maximum-scale=1.0">
    <title>TechCrunch &#8211; Startup and Technology News</title>
    <link rel='stylesheet' id='all-css-0' href='https://techcrunch.com/_static/??/wp-includes/css/dist/block-
library/style.min.css,/wp-content/plugins/wp-parsely/wp-parsely.css,/wp-content/themes/techcrunch-2017/build/ec/css/main.css?
m=1604003458' type='text/css' media='all'/>
    <meta name="oath:guce:product-eu" content="false"/>
    <meta name="oath:guce:consent-host" content="guce.techcrunch.com"/>
    <meta name="oath:guce:inline-consent" content="true"/>
    <link rel="EditURI" type="application/rsd+xml" title="RSD" href="https://techcrunch.com/xmlrpc.php?rsd"/>
    <link rel="wlwmanifest" type="application/wlwmanifest+xml" href="https://techcrunch.com/wp-includes/wlwmanifest.xml"/>
    <meta name="generator" content="WordPress 5.5.2"/>
</head>
```

Figure 15: Common WordPress metadata in the header of the website.

There are several ways used for identification that a website is powered by WordPress. For example, if a website has the "/wp-content/" directory or the "/wp-login.php" file available, it means that WordPress is running on that web application. However, a web application can be configured in a way that renames these paths or files, so it's not always possible to detect whether the website is running on WordPress. One way is by reading the meta tag, namely the tag "generator" and the "content" attribute with WordPress text. In case of hidden paths or tags, there are other methods to recognize the WP version, but they are time consuming and can slow down the scanning process. One such method is to visit the feed route and detect the presence of WP. Another way is to look at the README.html file where WordPress lists its version.

There is also the option to calculate the hashed value for attached style sheets, scripts, and other files for each software version and compare them with the files hosted by the website. However, this can also be unreliable and time consuming. Choosing an HTML Tag and versions of WordPress seems like a good approach, because it is displayed with the default WordPress configuration. By doing so, a completely reliable way of examining the installation version is possible. The obtained data core version and plug-ins are then compared with the entries in the vulnerability database to identify possible vulnerabilities. A very popular tool for reviewing an individual page with manual management is WPScan. Unfortunately, this tool does not allow to scan on a large scale in a decent time.

The scrapping mechanism for obtaining the core version of the WCMS website inspects the meta tag generator (<meta name=”generator” content=“WordPress 5.1.1”>). This information can also be obtained by analysing the CSS and JavaScript (JS) files, as there are cases where the WordPress core version is appended to the end of these files (/wp-includes/js/wp-emojirelease. min.js?ver=5.2). The developed tool applies both methods for retrieving the core version information. The information about plug-in versions is collected with the same method and their names (wp-content/plug-ins/wp-hide-post/public/js/wp-hidepost-public.js?ver=2.0.10) are collected as well. Unfortunately, not all plug-ins specify the issuing version, so these plug-ins are stored in a separate database.

4.3 Use of the Tool

The VulNet tool was developed to offer Internet WP website owners and users easy access to the data indicating the Internet web vulnerabilities. In that context, the usability of the tool is important. The VulNet user interface is built with web technology and is very simple

to use. The individual use starts with initiating the user-friendly client. Then the user enters the website domain or IP address, and this triggers the scan and the display of the results. This is a selective search type that makes it possible to detect vulnerabilities within a particular web server, identified by its IP or domain name. Additionally, the selection allows the user to request future automatic feeds regarding their website's vulnerabilities in period of time selected by the user. This VulNet feature is open to the user's needs, which seems to be especially important for web-hosting providers.

The scanning results provided by the VulNet tool are displayed using a web interface developed with PHP language. The VulNet interface shows the progress of the live scanning by showing the scan path on the map. A screenshot of the running portal with the crawler path and the reviewed WCMSs with their locations is shown in Figure 18. The backend scanning and parsing logic are written in the Python programming language.

In addition to the scanning method, the database for the data collected is also important in the development of the backend system. VulNet is using MongoDB, as it enables better processing of terabytes of data. Figure 16 presents the implementation of a python script for analysing 56,000 plug-ins in the signature database and determines the current version of the plug-in, checks the security of the latest version, and marks each version as vulnerable if one or more vulnerabilities are found. It also determines the individual type and the score of vulnerability.

Table 10 presents the VulNet functionality for analysing a specific domain or multiple domains with the command line. The commands vulnet.py -d domain.com, where the -d parameter is the ability to scan specific domain, or the -l parameter, where a list of domains intended to be specified, are listed. The tool also enables export of the results in json or txt format. The -v parameter is used for the output functionality which displays the current scan in real-time in more detail.

```
[+] Plugin slug: syntaxhighlighter
 |_ Plugin version: 3.5.1 - fixed: 3.5.1
 |_ Plugin rating: 88
 |_ Plugin downloads: 752323
 |_ Plugin last update: 2019-10-08 2:44pm GMT
 |_ Plugin added: 2007-09-10
 [+] Storing signature into DB (syntaxhighlighter)
 [+] Storing versions...
 [+] Storing version (1.1.0)
 [+] Storing version (1.1.2)
 [+] Storing version (2.2.2)
 [+] Storing version (2.3.2)
 [+] Storing version (3.0.0)
 [+] Storing version (3.1.0)
 [+] Storing version (3.1.1)
 [+] Storing version (3.1.11)
 [+] Storing version (3.1.13)
 [+] Storing version (3.1.2)
 [+] Storing version (3.1.4)
 [+] Storing version (3.1.5)
 [+] Storing version (3.1.6)
 [+] Storing version (3.1.7)
 [+] Storing version (3.1.8)
 [+] Storing version (3.1.9)
 [+] Storing version (3.2.0)
 [+] Storing version (3.2.1)
 [+] Storing version (3.3.2)
 [+] Storing version (3.5.1)
 [!] Skip wrong format version: trunk
 [*] Checking vulnerabilities...
 [!] Storing vulnerability (SyntaxHighlighter Evolved <= 3..)
 [!] Storing vulnerability (SyntaxHighlighter Evolved 3.1...)
 [+] Latest version - SAFE
 [*] Checking version vulnerability...
 [!] Version (1.1.0) VULNERABLE
 [!] Version (1.1.2) VULNERABLE
 [!] Version (2.2.2) VULNERABLE
 [!] Version (2.3.2) VULNERABLE
 [!] Version (3.0.0) VULNERABLE
 [!] Version (3.1.0) VULNERABLE
 [!] Version (3.1.1) VULNERABLE
 [!] Version (3.1.11) VULNERABLE
 [!] Version (3.1.13) VULNERABLE
 [!] Version (3.1.2) VULNERABLE
 [!] Version (3.1.4) VULNERABLE
 [!] Version (3.1.5) VULNERABLE
 [+] Version (3.1.6) SAFE
 [+] Version (3.1.7) SAFE
 [+] Version (3.1.8) SAFE
 [+] Version (3.1.9) SAFE
 [+] Version (3.2.0) SAFE
 [+] Version (3.2.1) SAFE
 [+] Version (3.3.2) SAFE
 [+] Version (3.5.1) SAFE
 [x] All vulnerabilities: {'6809-XSS': 4, '6810-XSS': 4}
 [*] Max score from all versions: 4
 [T] 0.1893913745880127
```

Figure 16: Python script performing the vulnerability check of plug-ins.

Table 10: VulNet Command Line Tool options.

Option	Functionality
-h, --help	Print out help message and instructions how to use the command line tool
-d DOMAIN, --domain DOMAIN	Domain name to crawl and asses
-v [VERBOSE], --verbose [VERBOSE]	Enable verbosity and display results in real-time
-l LIST, --list LIST	Initial domain list to crawl
-s START, --start START	Position to start crawling in case crawling was interrupted

-e JSON|TXT, --export JSON|TXT	Export results into JSON or TXT file

Figure 17 presents the operation of the VulNet tool, where the tool first connects to a remote server or a domain name and checks whether this domain already exists in the local database. In the next step, the tool checks whether the WP platform is running on the visited domain. In case WordPress is not present, the page is moved to the list of processed websites. If WordPress is present, a scan of the website with several different attempts for assessment is initialized, for the core version and the plug-ins detected. In the next step, each individual plug-in is verified, where the probability of matching the plug-in on the local database is calculated.

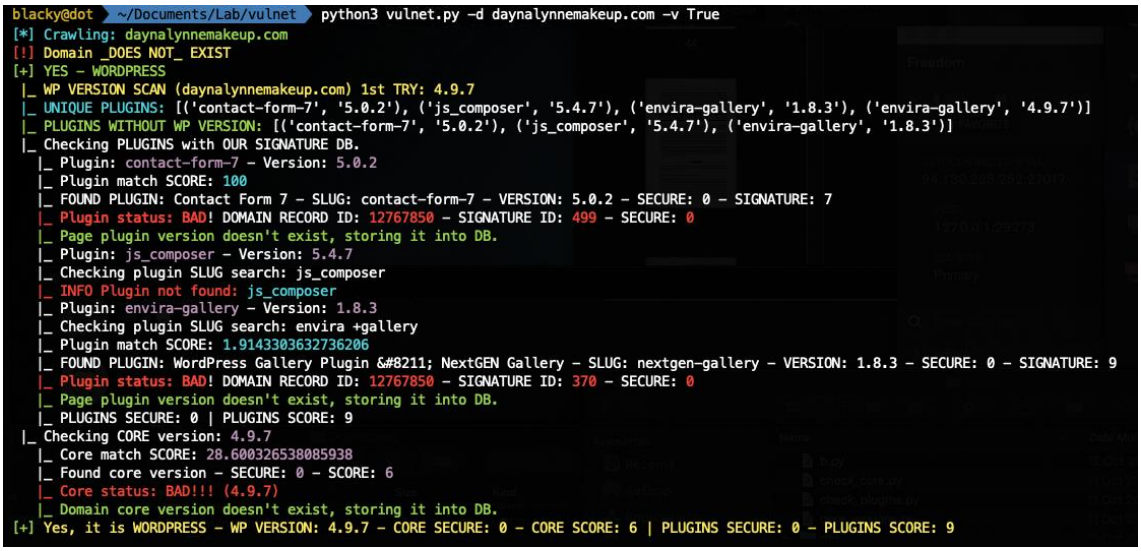

Figure 17: Python script that analyses domain vulnerabilities.

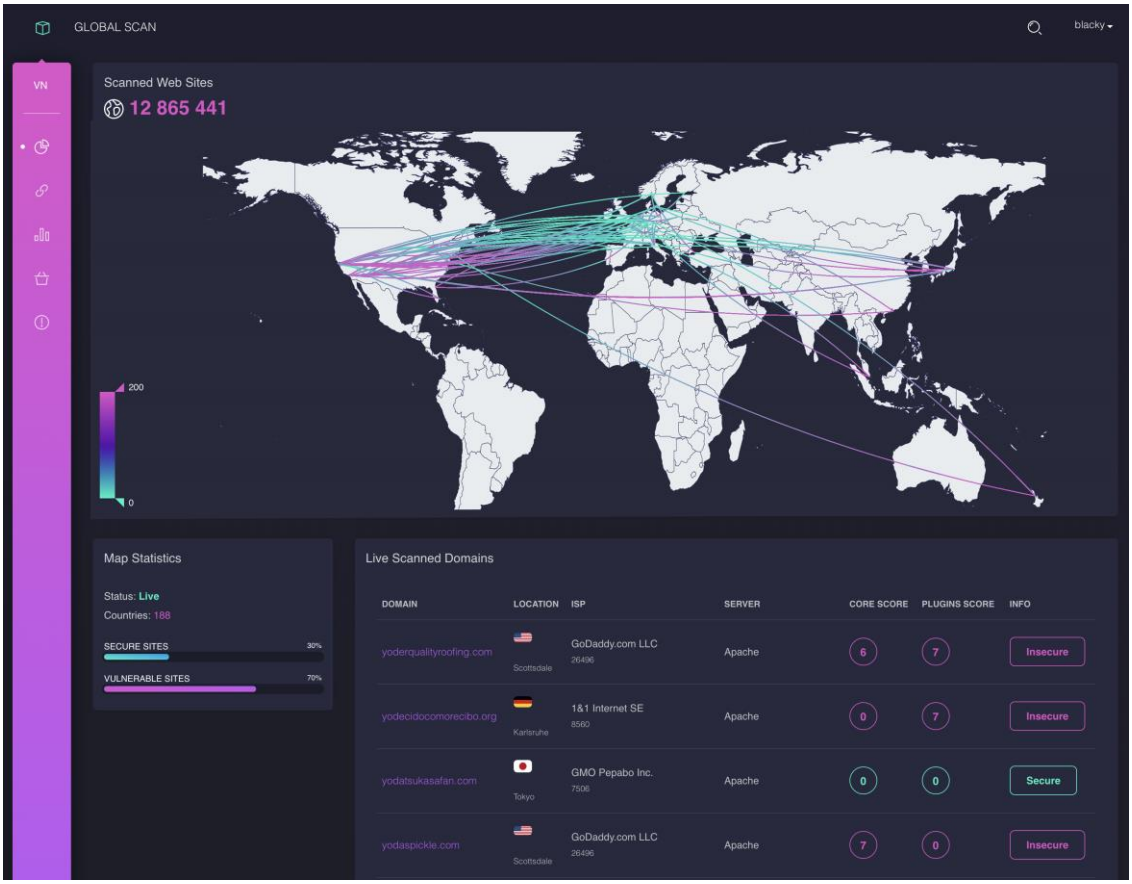

Figure 18: Real-time monitoring platform.

The same is done for the core version, where the probability of matching the version within the local database is calculated. If there is a match, the vulnerabilities are checked, and the plug-in or core is marked as secure or insecure. If several different plug-ins are vulnerable and insecure, a total vulnerability score for the plug-ins is calculated according to the Equation 1 after the core vulnerability is reviewed. A similar script is implemented in the backend where random live scanning of the web is performed. The only difference from the command line tool and from large scanning is that newly discovered links are added to the queue for further scanning.

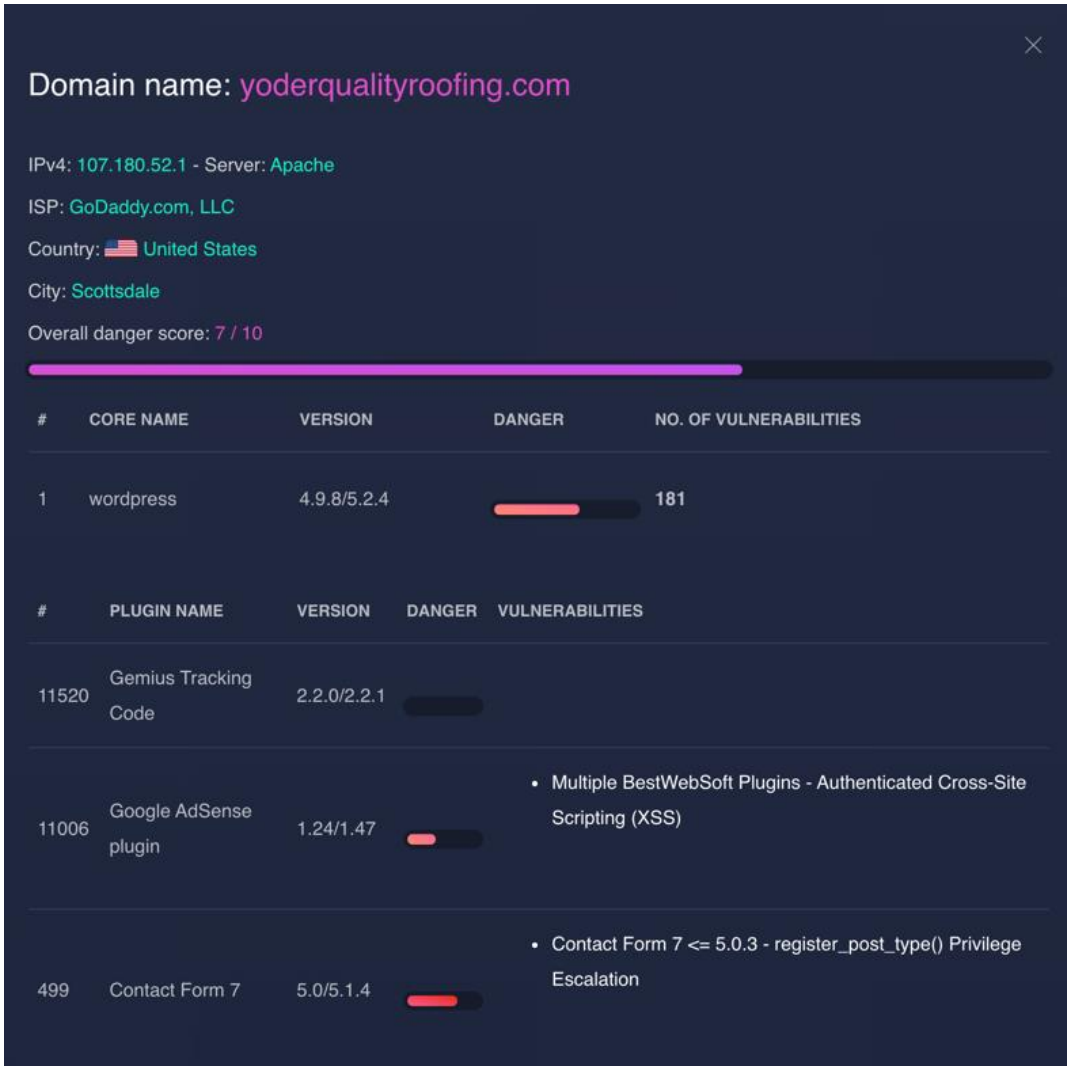

Figure 19: The measured vulnerability of the affected domain.

The measured vulnerability of the affected domain (e.g., the country's top-level domain) is shown in Figure 19. A report is generated for each individual domain based on the obtained data. The data on the screen show that the website is powered by an apache server. In addition to the server, there is also information about the geolocation of the server and a complete vulnerability score. Below the overall score is a list of identified components. The core version and vulnerability level are listed first. In addition, the number of all vulnerabilities that may affect the core version is shown. Below the core version, the detected plug-ins are listed with the plug-in version, vulnerability level and vulnerability type for each plug-in.

Figure 20 presents the previously described steps in the command line process of the VulNet tool when performing a scan of an individual domain. A similar process is carried out in the background within the VulNet tool, where a mass scan of the entire web is performed.

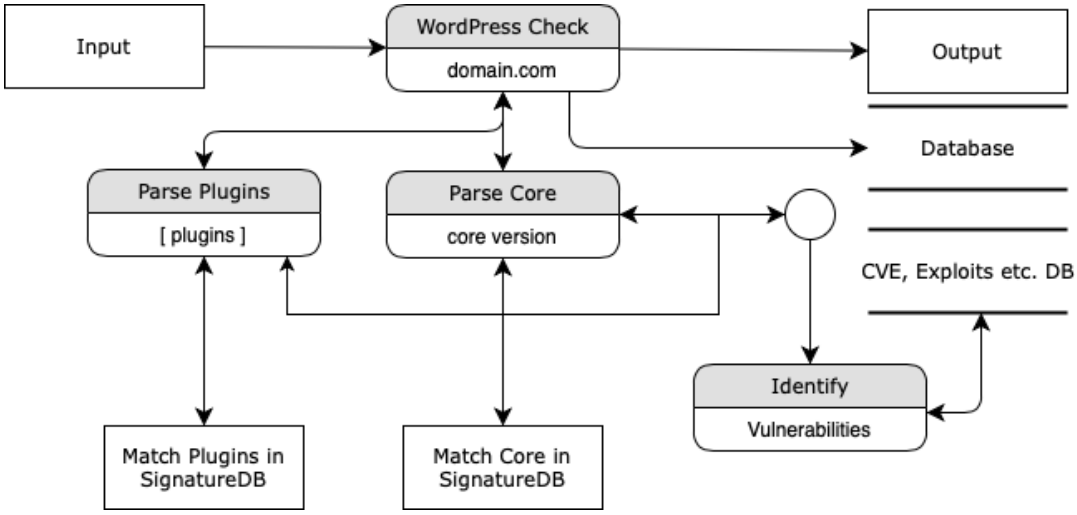

Figure 20: Domain scanning process and analyses.

4.3.1 Notification Service Design and Implementation

The public notification mechanism provides a vulnerability check through the VulNet portal, which enables regular notifications to users about new vulnerabilities in connection with the core version and plug-ins.

The use of the portal is restricted in the way that web domain owners must prove domain ownership. Verifying domain ownership is the process of proving that the domain owner is indeed its real owner. The ownership of the domain name must be verified and once it is confirmed, the owner has access to information related to the verified domain. The VulNet automatic mechanism regularly checks whether the site is still valid (i.e., verifies that the HTML validation tag is still present). If the verification is no longer confirmed, the permission for this website expires and the domain is suspended after a specified period of time. On the other hand, if a verified website owner leaves or loses ownership of the domain, another website owner is verified. The system allows verification via DNS records, meta tag and via a file upload to the server. Website owners decide for themselves how to prove their ownership of the domain name. The most common verification is through a file upload verification process, as it does not represent additional work or interference with the code or DNS records.

The site ownership verification can be carried out by uploading a special HTML file that is linked to a specific user. On the VulNet portal, the user follows the displayed instructions. If a user removes this verification file from their website, verification for that website is lost. The user agent performing the file upload check has the VulNet-Site-Verification user agent token. During the upload verification process the following errors may occur:

- The verification file could not be found. The file name or content may not match the specified HTML file.
- The verification file contains incorrect content.
- Redirection of the verification file to an unauthorized location. VulNet does not track redirects. Therefore, if the website redirects all traffic to another website, it is recommended that users use meta tag verification. Redirects within one website are allowed (i.e., from http://domain.com to http://www.domain.com).

If any error occurs, the ownership of the website could not be verified. Another way to verify the website ownership is to add a <meta> tag to the HTML of a particular page. The user selects the method of adding the HTML tag on the verification details page and follows the displayed instructions. If the meta tag is not found on the target website, an error information is returned. Each generated tag is tied to a specific user. The user agent performing the HTML tag verification has a VulNet-Site-Verification user agent token. The following errors can occur during the HTML tag verification process:

- The meta tag could not be found or is in the wrong place. The verification meta tag must be within the <head> section of the page. An example of the correct placement is shown below:

```
<html>
 <head>
   <title>Your page title</title>
   <meta name = "vulnet-site-verify" content="your verification string">
 </head>
```

- The meta tag was found, but is incorrect. In this case the VulNet agent found a meta tag to verify, but the content was incorrect.

The third way to verify the domain ownership is by adding a DNS record to the domain name provider portal. VulNet supports both the TXT format and the Canonical Name record (CNAME) format for DNS verification. To complete the domain ownership verification process, the user needs to log in into the domain name provider's DNS management system and add a DNS record as described below. Afterwards, VulNet checks to see if a record is present in the domain assignment. The CNAME record can be used to verify a domain or subdomain property that is itself defined by the CNAME record (i.e.: example.com or en.example.com). The user first obtains the DNS record, which is created on the VulNet web portal, and receives the string value that is used when entering the DNS record with the domain provider. Verifying DNS changes can take some time, therefore VulNet may not find the record immediately and the user may need to wait a few hours for the DNS record to be refreshed.

In verification with the TXT DNS record, the user first obtains the DNS record created by VulNet. The user receives a TXT record as the value of the string that the user adds to the domain provider. If the verification attempt is successful, the domain is added to the list of verified domains. As with the CNAME record, checking for DNS changes can take some time. The following verification errors may occur when checking the DNS record:

- The DNS record could not be found for verification. It may take a few minutes after the record is published for VulNet to see.

- The DNS verification record does not match.

- None of your DNS provider verification records match the value given to you by VulNet. Use the record provided on the verification page.

Once the page has been successfully added to the list of verified ownership websites, the user can start exploring website vulnerabilities. The daily manual domain verification through the VulNet portal can be time consuming and at the same time unnecessary in the absence of new vulnerabilities. Therefore, this platform presents a more automatic

system for informing users about new vulnerabilities. The user can set parameters when and how he wants to be informed about newly discovered vulnerabilities. The current version allows notifications via e-mail or Short Message Service (SMS), as this is the only way for the user to be immediately informed of the newly discovered vulnerabilities. The user can set the parameters defining the intensity of e-mail notifications. In addition to the vulnerability scan, the website keeps historical record of scans so that each individual user can view different scan states of their websites over time.

4.3.2 Integral Ethics in VulNet

Using web crawlers to search for vulnerabilities can be very beneficial, however it can also have negative consequences such as the consumption of network resources, as web crawlers require a large part of bandwidth, and this causes server congestions, especially if the frequency of accesses to a particular server is too high. The main values of VulNet are its scanning speed, ethical crawling and vulnerability assessment. Respecting the willingness of the website to allow browsing of WPCMS websites and the attached plug-ins is the main feature of the integral ethics. This property is not present in most of the known web scanners (Nessus, Skipfish, and WPScan). Also, for the scanning methods presented in the Chapter 3, researchers do not provide this property (Schagen, Koning, Bos, & Giuffrida, 2018), (Son & Shmatikov, 2013), (Nguyen & Hwang, 2016), (Akrout, Alata, Kaaniche, & Nicomette, 2014).

Another issue is that privacy concerns can arise if the obtained information is used in an unethical manner, i.e., if a crawler collects users' e-mail addresses or other sensitive information from the websites and adds them to a spam list. Crawlers are often used for illegal activities, i.e., copying copyrighted material, and therefore are becoming unwanted website visitors and regularly blocked by administrators. If the administrators do not want web crawlers to visit any of their websites, they can add a robots.txt file to the server or add a meta tag within the html header code, namely: <meta name="robots" content="noindex, nofollow"> that prevent access. Besides respecting robots.txt, VulNet gives users the ability to assess their website by themselves when verifying its ownership.

The creators of Masscan and ZMap recommend using their tools only in an ethical manner. Masscan and ZMap cannot honour the robots.txt protocol as they are port scanners and there is currently no similar standard for port scanning applications. Durumeric et al. suggest that researchers should set up a website about vulnerability crawlers to inform the administrator of the scanned website about the purpose of the scan and include contact information (Durumeric, Wustrow, & Halderman, 2013). In their case, the website was hosted at the same address where the scans originated, therefore it was possible to find the information about the owner of the active scan. The creators of ZMap also present seven guidelines for good scanning practices (Durumeric, Wustrow, & Halderman, 2013):

- Coordinate closely with local network admins to reduce risks and handle inquiries
- Verify that scans will not overwhelm the local network or upstream provider
- Signal the benign nature of the scans in websites and DNS entries of the source addresses
- Clearly explain the purpose and scope of the scans in all communications

- Provide a simple means of opting out, and honour requests promptly
- Conduct scans no larger or more frequent than is necessary for research objectives
- Spread scan traffic over time or source addresses when feasible

Based on guidelines and recommendations, the following notice is published on the VulNet crawler website: "*VulNET is a unique search engine which crawls the web with the one and only purpose to index the current running software version (i.e., Wordpress) on a user's website. Based on the security issues and the indexed version of the software it is possible to assess the potential damage. Owners of websites that are running vulnerable software can be warned to update their software. In every record, the domain owner information is required. The tool was developed at the Laboratory for Open Systems and Networks (J. Stefan Institute) for research purposes.*" In addition to this polite notice, some of the most common questions and answers were added too.

4.3.3 Evaluation and Comparison of the Tool with Existing Scanning Tools

The methods and algorithms presented in this chapter offer a more accurate vulnerability measurement by implementing innovative solutions in the scan part of the tool and in the vulnerability identification and analysis. This thesis aimed to construct a method for large-scale collecting of vulnerable WPCMS. The proposed methods for data collection and vulnerability assessment were presented in the beginning of this chapter as a continuation of the presentation of the tool implementation.

The server running the VulNet code uses the Ubuntu 20.04 operating system, 30 GB of memory, 64 cores and a gigabyte link. The tool was able to scan 126,086,633 unique links to websites. Among these websites, there were 16,274,980 with WPCMS installations that contained 17,126,445 installed plug-ins. The first scan with the tool started at the end of May 2019. The data collection covering the entire world ended within two months, at the end of July 2019. The second repeated scan was carried out in November 2019 to compare web space vulnerabilities in two separate periods. Excluding interruptions during the two months in the first scan, the algorithm was able to collect the data in 39 days, averaging about 2,245 reviewed websites per minute. When the conditions were ideal while having no latency during the crawling period, which can happen due to the slow responses from the visited server, the tool was able to process more than 6000 websites in one minute. Therefore, the developed tool and method outperform the methods of existing tools and researchers' approaches for vulnerability detection related to WCMS applications presented in chapters 2 and 3. Vulnerability testing scanners (Nessus, Skipfish, WPScan and OWASP ZAP) are designed for the small-scale data collection where one application or server is scanned in depth at a time. On the average, tools take from 10 seconds (best-case scenario) to several minutes to browse an individual web server. Vasek et al. inspected a random sample of around 200,000 web servers from the Alexa database in 9 days (Vasek, Wadleigh, & Moore, 2015). This would mean that even in the best-case scenario, scanning the Alexa Top 1 million websites would take over 45 days. Another approach of the web security scan was carried out by Goethem et al. and Chen et al., where in total they analysed 3 million websites (Goethem, Chen, Nikiforkais, Desmet, & Joosen, 2014), (Chen, Desmet, Huygens, & Joosen, 2016). They have obtained 22,000 websites which were then visited by mimicking the behaviour of a regular visitor. They run the experiment in a distributed

fashion using 60 networked machines, and as a result of the crawling experiment it took them five days to scan 22,000 websites. This would mean that even in the best-case scenario, the scanning of approximately 183 websites would take 1 hour.

Table 11 compares the properties of VulNet with known Internet scanners. The proprietary tools, Shodan Censys, ZoomEye and IVRE, are scanning open ports, as their main task is to detect vulnerabilities in web-connected devices such as cameras, sensors, etc. in the IoT environment. The other listed scanners detect the vulnerabilities of WCMS and show some common properties with the VulNet tool; however, none of them contains all the listed tool characteristics presented in Table 11.

Table 11: Properties of Known Web-Scanning Tools.

Tool	Aggressive assessment	Passive assessment	Automated CPE / CVE	Score System	Recognized web app.	Ethical	Scan Speed	Large-scale scan
Nmap	•	• •	• • •	• •	•	•	•	•
ZMap	•	• •	•	•	•	•	• • •	• •
Masscan	•	• •	•	•	•	•	• • •	• •
Shodan	•	• •	• •	•	•	•	• • •	• •
Censys	•	• •	•	•	•	•	• • •	• •
ZoomEye	•	• •	• •	•	•	•	• • •	• •
IVRE	•	• •	• •	•	•	•	• • •	• •
Nessus	• • •	•	• • •	• • •	• •	•	•	•
Skipfish	• • •	•	•	•	• • •	•	•	•
WPScan	• • •	• •	• •	•	• •	• •	•	•
OWASP ZAP	• • •	•	•	•	• • •	•	•	•
(Goethem, Chen, Nikiforkais, Desmet, & Joosen, 2014)	• • •	•	•	• •	• •	•	•	•
(Stock, Pellegrino, Li, Backes, & Rossow, 2018)	• •	•	•	•	•	• •	•	•
(Vasek, Wadleigh, & Moore, 2015)	•	• • •	•	•	• •	• • •	•	•
VulNet	•	• • •	• • •	• • •	• • •	• • •	• • •	• • •

'• • •' is used to denote a strong support, '• •' to denote a moderate support, and '•' for a weak support (i.e., unavailable) of a specific feature. CPE, Common Platform Enumeration, CVE, Common Vulnerability and Exposure.

Further, the tools WPScan, Nessus, Skipfish and OWASP ZAP are compared to VulNet. These tools do not enable a large-scale vulnerability assessment, nor do the solutions of other researchers enable the detection of vulnerable websites on a large-scale (Goethem, Chen, Nikiforkais, Desmet, & Joosen, 2014), (Stock, Pellegrino, Li, Backes, & Rossow, 2018), (Vasek, Wadleigh, & Moore, 2015), (Nguyen & Hwang, 2016), (Son & Shmatikov, 2013), (Nappa, Rafique, Caballero, & Gu, 2014). Their scanning results represent a much smaller sample of the reviewed websites compared to what VulNet can do. As mentioned in previous chapters, there are researcher approaches that enable fairly fast crawling using tools such as Masscan and ZMap, which offer the scanning of the Internet IPv4 protocol, but they are forgetting about the rest of the web; in addition, they do not support the assessment of plug-in vulnerabilities (Schagen, Koning, Bos, &

Giuffrida, 2018), (Nappa, Rafique, Caballero, & Gu, 2014), (Kim, Kim, & Jang, 2018). A good example of such a study is the one by Laitinen, where a sample from the Censys dataset was used with a total of 63,297,814 unique IP addresses (Laitinen, 2018). They were able to find 692,039 unique IP addresses which responded to HTTP / HTTPS requests with metadata that matches the WordPress installation. Given the number of websites searched and the number of detected WPCMS websites, it is clear that crawling by IP and using tools such as Zmap or Masscan is not as efficient as general web crawling.

As presented in Table 11, the property of ethics is not present in most of the known web scanners (Nessus, Skipfish, WPScan and OWASP ZAP). The other researcher approaches in their scanning methods do not provide this property either, but rather use the black box approach for vulnerability assessment (Chen, Desmet, Huygens, & Joosen, 2016), (Stock, Pellegrino, Li, Backes, & Rossow, 2018), (Schagen, Koning, Bos, & Giuffrida, 2018), (Son & Shmatikov, 2013), (Nguyen & Hwang, 2016), (Akrout, Alata, Kaaniche, & Nicomette, 2014). Unlike most of these tools, VulNet does not use the black box approach to assess vulnerabilities. VulNet does this by means of certain features of the white box approach and maintains its ethics and friendliness to website owners. In terms of assessing the impact of vulnerability (scoring), it is noticeable that none of the presented tools enable a scoring mechanism in the way provided by VulNet.

For a decade, the research community has expensively focused on discovering various issues, such as WCMS vulnerabilities (Akrout, Alata, Kaaniche, & Nicomette, 2014), (Goethem, Chen, Nikiforkais, Desmet, & Joosen, 2014), (Vasek & Moore, 2014), but relatively recently the security community has begun exploring the effectiveness of outreach efforts to inform affected parties. On the one hand, there are services that offer publicly available collected information about websites like Shodan, Censys and ZoomEye do. VulNet initially offers similar publicly available information about vulnerable websites. Unfortunately, due to the potential abuse by attackers, the decision was made to disable this feature. On the other hand, there are researchers who discover vulnerabilities and report them to website owners, and analyse the responses with different communication approaches (Stock, Pellegrino, Li, Backes, & Rossow, 2018), (Li, et al., 2016) and (Soussi, Korczynski, Maroofi, & Duda, 2020). Some of these approaches have been described in the previous chapters. It was also observed that communication via e-mails or other channels was no longer as effective, therefore researchers suggest other approaches. That is why VulNet has introduced a new approach, namely by verifying the ownership, so that users can obtain data about detected vulnerabilities on their site safely. The advantage of such an approach is that users choose to inform for themselves, but at the same time attackers cannot exploit this information.

Chapter 5

Evaluation of the Collected Data

The goal of this chapter is to present the results of the analyses performed with the aim to provide information about the current state of vulnerabilities in WP-based web servers in the wider Internet space. Another objective is to discuss the factors that have an impact on the higher presence of vulnerabilities in WP websites in some European countries. This chapter first introduces the collected data in sufficient detail and then provides information about the impact of plug-ins, shared hosting and running the latest core version on a higher insecurity of the examined web space. The impact of digital skills within the population of the country and the level of insecurity are analysed as well.

5.1 Collected Data

VulNet was developed with an aim to examine as many websites as possible in an acceptable time period. The exploratory study over the Internet started at the end of May 2019. Data were collected for two months and the collection ended by the end of July 2019. This scanning time was longer than expected due to the change that happened in the meantime because the versions of the WPCMS core and plug-ins changed. The VulNet code had to be adjusted to accommodate the numbers of the new versions. In the first scan, the tool successfully established 126,086,633 unique links to websites. Among these websites, there were 16,274,980 with WPCMS installations that contained 17,126,445 plug-in installations.

The second scan was carried out in November 2019 and lasted for 5 days. The second scan accessed 16,274,980 websites that were found in the first scan. Among them, 12,865,441 WPCMS installations were found with 18,127,403 plug-in installations. The number of servers decreased due to the unavailability of some servers identified in the first scan.

The scanning of the European web space provided a set of 23,131,336 websites in the first scan, among which 3,738,654 websites (16%) were found to run WPCMS applications. In the European sample, the hosted websites were selected as websites from to a particular European country according to their domain suffix and not according to the location of the web server. The sets of websites were from to the following European countries: Germany (DE), Netherlands (NL), France (FR), Great Britain (GB), Italy (IT), Denmark (DK), Poland (PL), Spain (ES), Sweden (SE), Switzerland (CH), Czech Republic (CZ), Ireland (IE), Finland (FI), Austria (AT), Romania (RO), Belgium (BE), Hungary (HU), Bulgaria (BG), Norway (NO), Slovakia (SK), Estonia (EE), Slovenia (SI), Portugal (PT), Croatia (HR), Lithuania (LV), Luxembourg (LU), Greece (GR), Island (IS), Latvia (LT), and Cyprus (CY). The selection of these countries was based on the availability of indices and data used by Eurostat (Eurostat, 2019) and the ITU International Telecommunication

Union (ITU, 2019) for measuring the level of digital development; Eurostat is the statistical office of the European Union located in Luxembourg. Eurostat is a partnership with the national statistical authorities of all EU Member States and EFTA countries that are obliged to provide reliable and high-quality statistical information applied in decision-making, research and discussion.

ITU is an agency of the United Nations that covers the field of telecommunications and is defined as a public-private partnership consisting of 193 United Nations (UN) member states and around 800 private sector entities and academic institutions. The ITU membership represents the world's largest manufacturers and telecommunications operators and also small, innovative players working with new and emerging technologies, along with leading research and development institutions and academia. One of the main tasks of the ITU is to set international standards for Information and Communication Technology (ICT). In addition, the ITU is also active in broadband internet, wireless technology, aeronautical and maritime navigation, radio astronomy, satellite meteorology, television broadcasting and next generation networks. One of the main activities of the ITU is the collection, verification, and coordination of ICT statistics for different countries around the world. There are two key sets of ICT data collected by the ITU directly from countries:

- Collected ICT data from national ministries of ICT and regulators, including data about fixed telephone networks, mobile cellular services, internet/broadband, transport, revenue and investment; and prices of ICT services.

- ICT data for households collected from national statistical offices (NSOs): including data on household access to ICT and individual use of ICT.

Statistics can answer many questions, therefore the ITU and Eurostat data were used as a source in our study with the aim of discovering the factors which may have an impact on the appearance of a higher percentage of vulnerable WP websites in the studied sample.

5.2 Summary of the Statistical Analysis

The countries' web spaces used in the study are shown in Figure 21. The minimum scanned total websites per country was 3,554 (Cyprus) and the maximum total number of scanned websites was found in Germany, a total of 6,556,776. This points to a high variability in the number of websites per country; the mean value of the whole sample is 1,246,212 and the median 53,261. The smallest number of WP websites detected per country was from Cyprus (864) and the highest in Germany (805,279) as displayed in Table 12, which is in line with the population size in these countries. Out of the total European sample of 3,738,654 WP websites, 1,339,325 were found to be vulnerable, while the number of websites without vulnerabilities was 1,187,085. On the average, the vulnerability status counts for 34% of all found WP websites could not be identified due to the hidden WordPress versions of either the core or plug-ins.

Table 12: Summary statistics of all WPs found in the European sample, percentage of vulnerable websites, and DS index.

	total WP	**unknown** [%]	**secure** [%]	**insecure** [%]	**critical** [%]
Mean	124621.8	34.02	27.6	38.38	30.98
Std	183737.33	3.56	4.66	4.52	4.57
Min	864	27.38	21.22	30.31	22.1
25%	14924.75	31.38	24.19	35.32	27.29
50%	53261	34.21	26.05	39.44	31.97
75%	140195	36.98	30.98	41.57	34.76
Max	805279	40.15	37.85	47.37	41.31

Each plug-in or WP core version was assessed according to the developed scoring system described in Section 4.1.2 and was classified into one of three categories: secure, insecure and unknown. A website was considered secure if the core version and all (if any) plug-in versions are also secure, meaning that no vulnerability was detected. WP websites with hidden versions of either core and plug-ins which made it impossible to assess were labelled "unknown". Even in cases when the site was flagged as unknown, VulNet was capable to identify the website as a WP application. It is well known that web owners do not provide the version number of the core because they believe that if the core version is hidden, the probability that the website will be attacked with malicious code is much lower as that the attackers do not have information about existing vulnerabilities.

However, this strategy was found as not useful because the study by Vasek et al. did not confirm reliably that hiding the server information promotes or inhibits the compromising of the web server (Vasek, Wadleigh, & Moore, 2015). However, the practice shows that this belief is still in place as WordPress and Google have the practice to not remind WPCMS owners who hide their version (Vasek, Wadleigh, & Moore, 2015) for security reasons. However, some other studies have shown that hiding the core version does not ensure a higher website protection (Chen, Desmet, Huygens, & Joosen, 2016).

The share of secure WP websites within the 30 countries was between 21.22% and 37.85%, with an average of 27.6%. The percentage of insecure WP websites in individual European countries ranged between 30.31% and 47.37%, the average of the whole set being 38.38%.

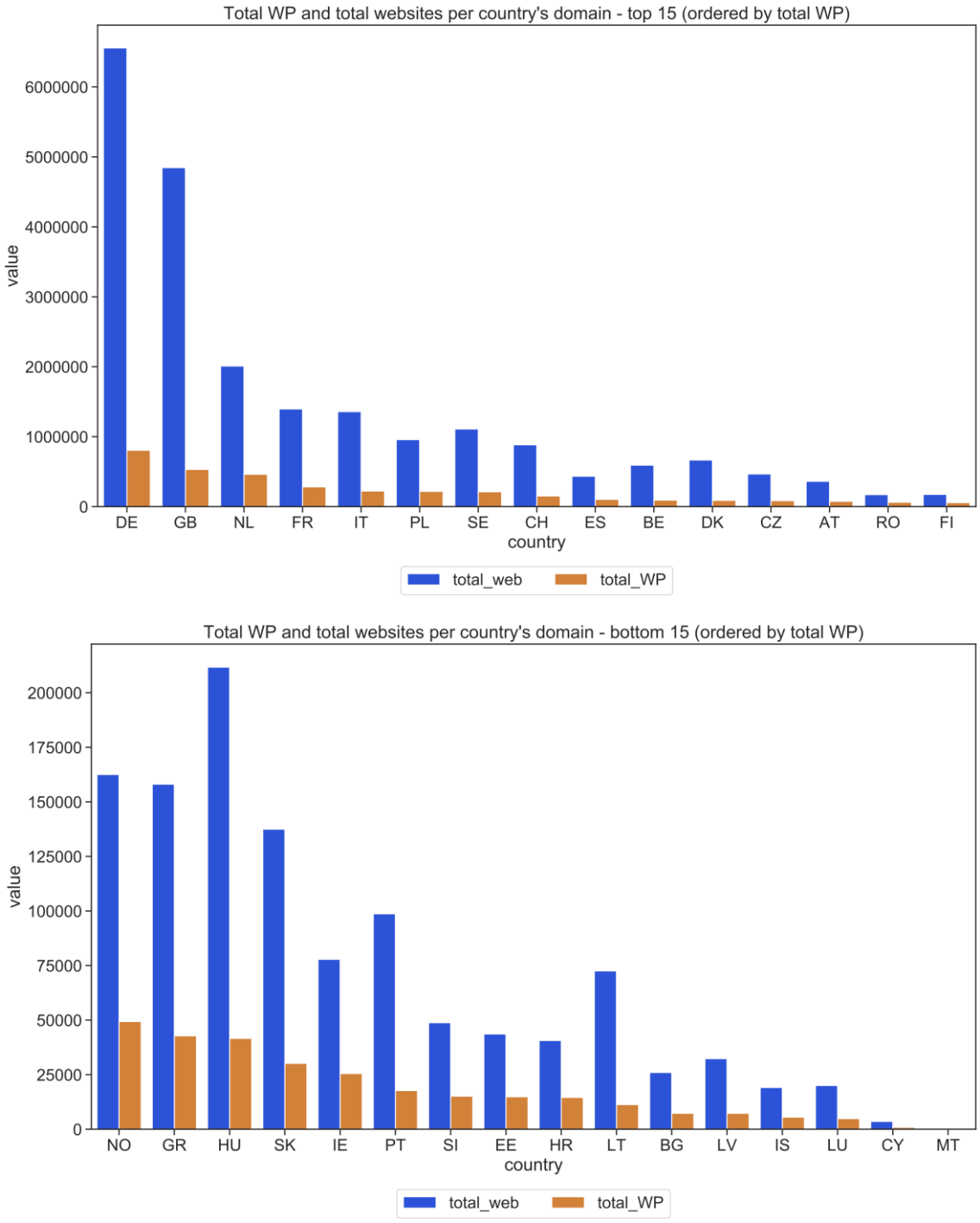

Figure 21. The total number of websites and the number of detected WordPress websites.

The macro-analysis of the collected data provided a good insight into how plug-ins and the state of the core influence the overall situation of a WP website's vulnerability. Figure 22 presents the percentages of the combined insecurity score values (core and present plug-ins), the vulnerability score value of the website core and the plug-in versions.

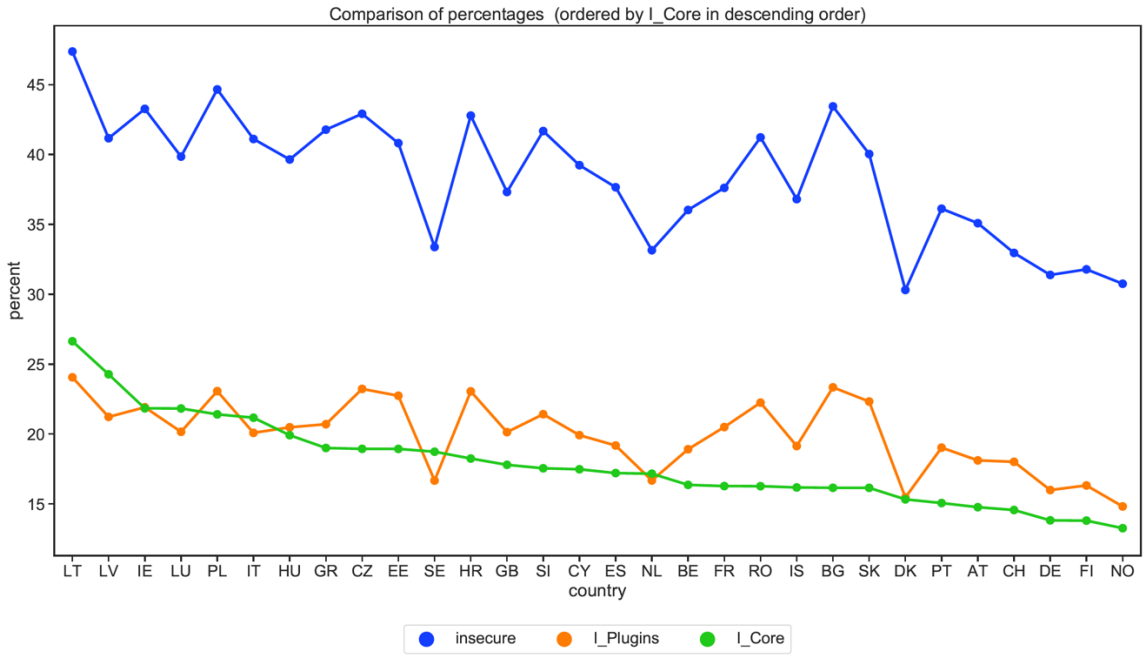

Figure 22: Percentage of all insecure WP websites, all insecure plug-ins and insecure core versions.

Based on the similarities in the shape and size of the insecure line (blue) and plug-in line (orange) with the high correlation (r = 0.91) we can conclude that the percentage of insecure websites primarily depends on the number of insecure plug-ins. In the cases of Lithuania, Latvia, Luxemburg, Italy, and Sweden, the overall percentage of insecure core versions is higher than the percentage of insecure plug-ins, and that affects the overall insecurity in these countries, which can be explained by the delays in replacing the WPCMS core with the newest version where most of the bugs in the old version are removed. All insecure core websites that were critically unsafe were those that have an insecurity score value at least 5 out of the maximum score value of 10.

5.3 Plug-ins as a Risk Factor for a Higher Insecurity

One of the most popular features of WordPress applications is the use of plug-ins. Plug-in-related vulnerabilities tend to be recurrent, exploitable, hard to detect, and may lead to severe consequences for the website in question. With many plug-ins and missing software updates on one platform, a website is quickly obsolete and can be attacked, as several studies confirmed.

Vasek and Moore found dozens of WordPress plug-ins and Joomla extensions that were positive risk factors for compromise (Vasek & Moore, 2014). In their first publication addressing the issue, they found that a key driving factor for most targeted CMSs is the popularity of the platform. Another research by Koskinen et al. reported that the quality and number of vulnerabilities detected in plug-ins varies (Koskinen, Ihantola, & Karavirta, 2012). Their study of both good user ratings and a high download count plug-ins discovered a severe SQL injection vulnerability. They reviewed the WordPress vulnerability catalogued by the NVD database along with patches maintained by the WordPress plug-in repository and identified the main types of vulnerabilities within plug-ins studied along with their impact on the size of the patch that fixes the vulnerability. They found that plug-in-related vulnerabilities, if exploited, have severe consequences as they frequently

remain unnoticed for years before being fixed (Koskinen, Ihantola, & Karavirta, 2012). Walden et al. presented the research results about plug-in vulnerabilities in twelve open-source web applications assessing 13,778 plug-ins for those applications (Walden, Doyle, Lenhof, Murray, & Plunkett, 2010). They used an automated static analysis tool for counting vulnerabilities. Plug-ins made up 93% of the aggregate code base and contained 92% of the 125,110 vulnerabilities found. The same authors analysed the density of individual vulnerability categories, finding plug-ins to have many more cross-site vulnerabilities and fewer injection vulnerabilities than the website core code (Walden, Doyle, Lenhof, Murray, & Plunkett, 2010).

With previous research of plug-in vulnerabilities in mind, we examined the situation of plug-in vulnerabilities within the larger-scale scan data collected among the European countries. Table 13 presents the data from the top five countries with the most vulnerable and out-of-date plug-ins, including the number of vulnerable installations detected. There is a noticeable difference between the number of vulnerable plug-ins detected and the number of outdated plug-ins, meaning that even though many plug-ins are not up to date, this does not mean they are vulnerable, however these numbers are absolute. Namely, certain exploits and detected vulnerabilities are tied to specific versions of plug-ins.

Table 13: Top five ranked countries with vulnerable and outdated plug-ins.

	Vulnerable Plug-ins			**Outdated Plug-ins**	
	Country	**Total Installations**	**Plug-in Score > 5**	**Country**	**Total Installations**
1	United States	1,889,748	1,025,013	United States	6,026,011
2	Germany	563,823	398,109	Germany	2,844,033
3	France	227,126	174,263	France	1,045,303
4	Netherlands	223,258	126,173	Netherlands	1,026,131
5	Japan	189,847	125,930	United Kingdom	794,444

Table 14 presents the top ten vulnerable WP plug-ins found in the collected data from the first large world-wide scan and the top ten installed WP plug-ins among the vulnerable websites, including the rate, the score, and downloads. The total number of the top ten most vulnerable plug-ins has nearly two and a half million numbers of detected plug-in installations. However, plug-ins are very popular and are appreciated by the end user community as their user satisfaction rating is close to 80%. Contact Form, WooCommerce, Photo Gallery, NextGen Gallery, and MailPoet are the top five plug-ins that have a vulnerability score higher or equal to level 7. The max score of plug-ins is determined by the max vulnerable score value of all versions for each plug-in. The top four installed plug-ins on the list of websites with the most vulnerable plug-ins are Contact Form 7, WooCommerce, JetPack, and Instagram feed. They introduce a greater threat to the users of the WP platform. End users use popular plug-ins as they trust them because of their satisfaction rating score, which in these four cases is quite high.

Table 14: Top ten vulnerable plug-in installations and top ten installed plug-ins in the 1st scan.

	Top vulnerable plug-ins – 1st scan					**Top plug-in installations – 1st scan**			
	Plug-in	**Downloads**	**Rating**	**Total Installations**	**Max Vulnerability Score**	**Plug-in**	**Downloads**	**Rating**	**Total Installations**
1	Contact Form 7	108,553,280	82	1,692,082	8	Contact Form 7	108,553,280	82	2,809,317
2	WooCommerce	68,594,243	92	425,952	7	WooCommerce	68,594,243	92	663,696
3	Instagram Feed	6,973,505	98	236,387	4	JetPack	123,609,861	78	554,627
4	Jetpack	123,609,861	78	216,783	5	Cookie Notice	4,718,788	96	547,433
5	GDPR Cookie	1,468,293	94	137,626	3	Elementor	26,513,464	96	387,513
6	Nextgen Gallery	26,080,698	84	137,105	7	Instagram Feed	6,973,505	98	279,063

7	Add to Any	10,609,790	94	109,193	4	Google Analytics	47,302,691	86	269,596
8	Gallery Bank	3,146,983	90	95,414	6	Site origin Panels	23,771,290	96	254,733
9	Photo Gallery	9,621,845	92	80,482	7	TablePress	4,485,166	100	244,012
1 0	bbPress	5,419,583	80	72,683	6	Mailchimp	19,121,457	96	222,095

The time interval between the first and second scan carried out in this study was 4 months apart and in this short period of time, the number of downloads of the most popular plug-ins, e.g., Contact Form, JetPack, WoCommerce and Instagram Feed, increased by 44 million downloads. This indicates how popular these plug-ins are and, consequently, when the number of installations increases, so does the number of vulnerable websites.

Table 15 presents the results collected within the second large scan. The number and the type of the identified top 10 vulnerable plug-in installations did not change with the time. However, some other changes were noticed. The bbPress plug-in slipped out from the top ten vulnerable plug-in scale. In the meantime, the NextGen gallery and Photo gallery plug-ins have increased their overall vulnerability score, namely Photo gallery has increased the threat level from 7 to 10 and NextGen gallery from 7 to 9 based on information found in updated vulnerability database.

Table 15: Top ten vulnerable plug-ins and top ten installed plug-ins in the 2nd scan.

	Top vulnerable plug-ins – 2nd scan					**Top plug-in installations – 2nd scan**			
	Plug-in	**Downloads**	**Rating**	**Total Installations**	**Max Vulnerability Score**	**Plug-in**	**Downloads**	**Rating**	**Total Installations**
1	Contact Form 7	120,919,557	82	1,227,669	8	Contact Form 7	120,919,557	82	3 472,531
2	WooCommerce	77,248,054	92	360,012	7	WooCommerce	77,248,054	92	659,439
3	Jetpack	145,368,543	78	143,218	5	JetPack	145,368,543	78	561,616
4	Instagram Feed	8,421,079	98	143,156	4	Cookie Notice	5,034,909	96	483,196
5	Photo Gallery	10,488,672	92	130,577	10	Elementor	36,588,270	98	413,775
6	GDPR Cookie	1,559,490	94	120,088	3	Google Analytics	55,528,987	88	347,211
7	Nextgen Gallery	27,627,089	84	107,673	9	Instagram Feed	8,421,079	98	240,229
8	Add to Any	11,219,052	94	104,533	4	Site origin Panels	25,781,369	96	229,949
9	Gallery Bank	3,184,763	90	87,806	6	Cookie Law Info	4,422,347	92	204,577
1 0	MailPoet	9,207,478	96	55,398	7	TablePress	4,750,634	100	198,339

The analysis of the scanned dataset from the European environment showed that on the average there is one plug-in per WP website in total numbers. As shown in Table 16, T_cores and T_Plug-ins represent the total number of cores and plug-ins scanned. In Figure 23, the F_cores and F_Plug-ins represent the number of identified versions of cores and plug-ins, respectively.

Table 16: Summary of total numbers (Plug-ins, Core versions).

	T_Plug-ins (Total Plug-ins)	**T_cores (Total Cores)**
No. of countries	30	30
mean	216743.60	124621.80
std	315978.85	183737.33
Min	1682.00	864.00
50%	94754.50	53261.00
Max	1423886.00	805279.00

The ratios of the percentage of insecure WP websites, all combinations of plug-ins and core versions are shown in Figure 24. The label ag_## presents the aggregated percentage of the aggregated website combination of plug-in and core version situation: 0-insecure, 1-secure, 2-unknown.

As shown in Table 17, the combination ag_01 represents the percentage of websites with at least one insecure plug-in and a secure core version. The ag_00 combination

represents the percentage of websites where at least one plug-in and the core are both insecure. In the combinations ag_20 and ag_02, the state of the plug-ins or the core is unknown. While the ag_20 state represents the percentage of an unknown plug-in and insecure core installation, the combination ag_02 represents the percentage of a vulnerable plug-in and an unknown core version. The combination ag_10 represents the percentage of secure plug-in installations with an insecure core.

Table 17: Combinations of plug-ins and core versions.

Combination ag_##	**Meaning**
ag_00	at least one plug-in and the core are both insecure
ag_01	at least one plug-in insecure and the core version secure
ag_10	all plug-ins secure and the core insecure
ag_02	at least one plug-in insecure and the core version unknown
ag_20	all plug-ins unknown and the core insecure

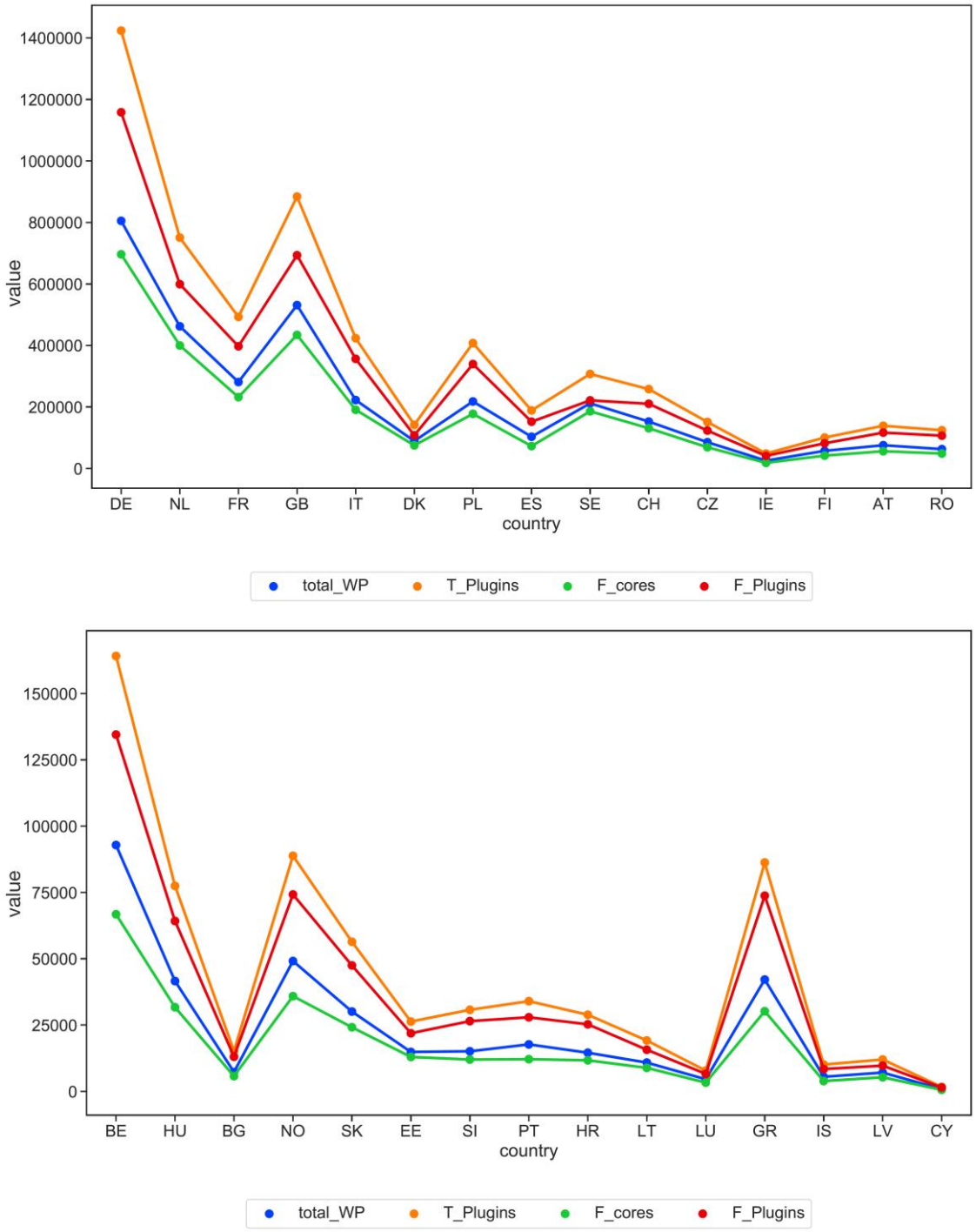

Figure 23: The difference in statistics between the total WP and total core versions and WP plug-ins in Europe.

The lines in Figure 24 show that most insecure WP websites are mainly due to the ag_01 combination (green line). If compared against the shape of the insecure (blue) line, the next biggest contributor would be the ag_00 combination (orange line). In both, at least one plug-in has an insecure version through which the whole website could be exploited. The previous studies confirmed that plug-ins - which are very popular features in WP applications - represent the greatest risk for the security of the website. Our findings confirm that plug-ins are a major factor for a higher presence of vulnerabilities in the

WPCMS websites. Therefore, the hypothesis (H2) that vulnerable plug-ins are a positive risk factor for a higher presence of vulnerabilities has been confirmed on this data sample.

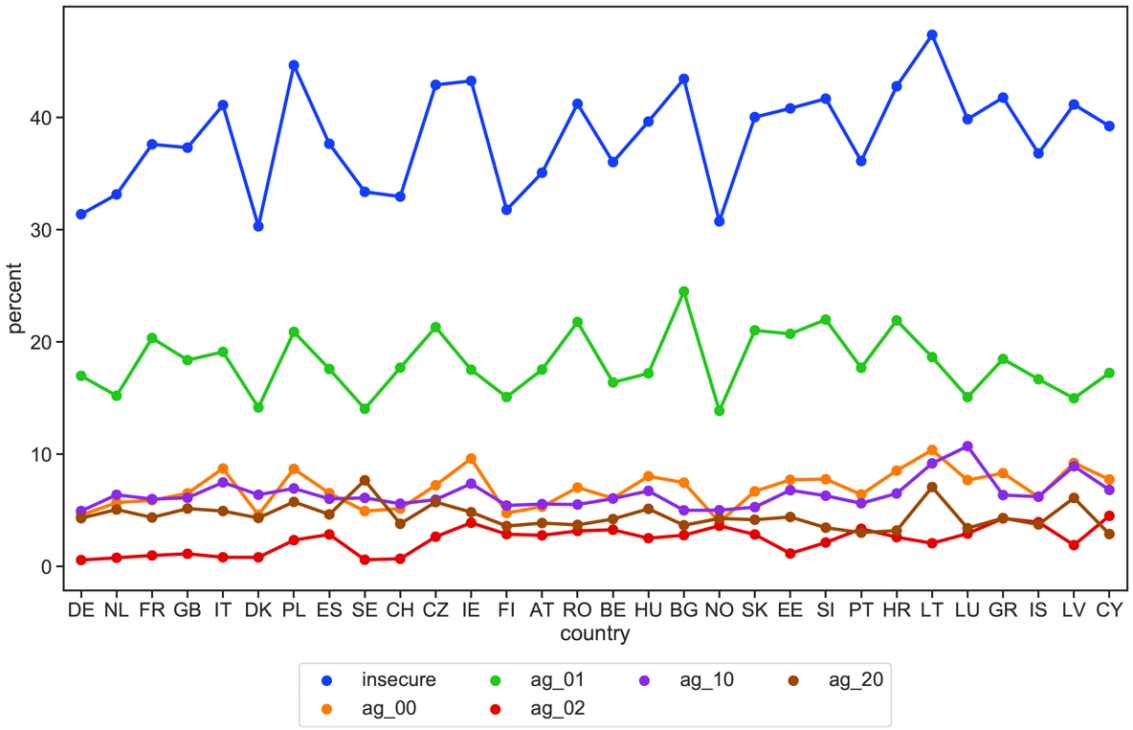

Figure 24: The combinations of plug-in and core versions vs. the percentage of overall insecurity by country.

5.4 Insecurity and Website Application Content

The WP application is used for offering services in different types of sectors. Among them, there are websites and services with contents from the business sector, e-commerce, data directories, social networking, news or magazines, education, e-health and many more.

The collection of data was used for exploring the security situation among web services offered in the fields of healthcare, different types of societies/associations, finance, news, education, and research institutions. For each of these six sectors, keywords (up to 60) were selected to describe their content (Appendix B1). These sectors were selected according to a survey carried out by Concordia cybersecurity competence centres (Blažič, 2021) which focused on specific industry sectors strongly relying on web services for whose services data protection is of utmost importance: finance and e-health. Other important sectors also strongly relying on web services were added to complement the sector selection. All selected keywords were translated into the languages used in the European country in question. The appropriateness of the selected keywords for a particular sector was compared with the words in the domain name and the website title. The number of keywords used to classify websites for each country is given in Table 18 under the column K_T. The table also presents the number of all keywords found (K_F) for each country. The following columns (rank1, rank2, etc.) rank the percentages of classified websites per sector of each country. For example, rank1 indicates the highest number of classified websites using the WP application for services in the selected sector. The next rankings contain the lowering numbers of classified websites.

Certain languages have specific letters or use a non-Latin alphabet (e.g., Cyrillic or Greek), and can have several synonyms for the selected words that can be used as keywords. For some countries where several official languages are used, for example in Luxembourg, Belgium, Switzerland and Cyprus, more sets of keywords were used in the exercised search.

Table 18: Total number of classified websites for a corresponding number of keywords and percentage in each of the five fields (sorted in descending order).

	classified	K_F	K_T	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
IS	412	28	35	EDUCATION [47.33%]	INSTITUT [17.23%]	HEALTH [11.89%]	FINANCE [9.95%]	NEWS [7.28%]
HU	4614	31	33	EDUCATION [36.24%]	FINANCE [17.17%]	NEWS [17.01%]	HEALTH [16.56%]	INSTITUT [12.03%]
IE	2373	19	30	EDUCATION [31.06%]	FINANCE [24.02%]	HEALTH [23.56%]	INSTITUT [10.54%]	NEWS [6.74%]
LV	595	27	33	EDUCATION [30.25%]	INSTITUT [28.40%]	FINANCE [14.79%]	HEALTH [12.10%]	NEWS [10.08%]
CZ	5937	25	26	EDUCATION [29.22%]	HEALTH [21.81%]	FINANCE [21.44%]	INSTITUT [12.97%]	SOCIETY [9.04%]
DE	60723	26	27	EDUCATION [23.21%]	FINANCE [22.28%]	HEALTH [21.72%]	INSTITUT [16.58%]	NEWS [9.04%]
CY	189	17	59	FINANCE [72.49%]	HEALTH [9.52%]	INSTITUT [6.35%]	NEWS [5.82%]	EDUCATION [4.76%]
PL	22314	26	26	FINANCE [44.61%]	HEALTH [24.71%]	EDUCATION [15.16%]	INSTITUT [10.08%]	NEWS [5.23%]
EE	1923	28	30	FINANCE [44.57%]	EDUCATION [30.16%]	HEALTH [16.38%]	INSTITUT [5.51%]	NEWS [2.60%]
GB	46353	17	17	FINANCE [44.25%]	HEALTH [21.71%]	EDUCATION [13.06%]	INSTITUT [11.37%]	NEWS [7.51%]
NL	33868	25	25	FINANCE [43.95%]	HEALTH [19.11%]	EDUCATION [16.21%]	INSTITUT [11.99%]	NEWS [8.03%]
SE	15132	27	27	FINANCE [34.17%]	HEALTH [31.46%]	EDUCATION [14.68%]	INSTITUT [11.39%]	NEWS [7.32%]
LU	361	34	55	FINANCE [33.24%]	EDUCATION [20.22%]	INSTITUT [18.01%]	HEALTH [14.40%]	SOCIETY [8.86%]
SK	2088	35	36	FINANCE [33.14%]	EDUCATION [24.90%]	HEALTH [19.64%]	SOCIETY [10.92%]	INSTITUT [8.00%]
NO	3078	23	25	FINANCE [31.90%]	HEALTH [31.35%]	EDUCATION [21.96%]	INSTITUT [9.58%]	SOCIETY [3.80%]
BE	6119	52	60	FINANCE [30.51%]	EDUCATION [20.66%]	HEALTH [19.68%]	INSTITUT [19.61%]	NEWS [6.34%]
FI	4220	27	30	FINANCE [28.96%]	HEALTH [28.91%]	EDUCATION [27.82%]	INSTITUT [10.92%]	NEWS [2.27%]
FR	24335	29	30	FINANCE [28.47%]	INSTITUT [25.80%]	HEALTH [17.99%]	EDUCATION [17.76%]	SOCIETY [6.12%]
BG	653	32	40	FINANCE [27.41%]	HEALTH [26.34%]	NEWS [24.04%]	INSTITUT [10.87%]	EDUCATION [8.73%]
CH	11581	55	60	FINANCE [26.20%]	HEALTH [23.65%]	EDUCATION [20.42%]	INSTITUT [16.60%]	SOCIETY [7.69%]
PT	1871	31	32	HEALTH [35.60%]	FINANCE [26.56%]	EDUCATION [13.58%]	INSTITUT [12.40%]	NEWS [6.57%]
RO	5338	29	31	HEALTH [32.75%]	NEWS [21.81%]	FINANCE [14.84%]	EDUCATION [13.38%]	INSTITUT [13.04%]
LT	866	23	25	HEALTH [29.79%]	EDUCATION [21.13%]	FINANCE [16.97%]	NEWS [15.70%]	INSTITUT [14.20%]
DK	5897	22	24	HEALTH [28.83%]	EDUCATION [23.45%]	FINANCE [21.11%]	NEWS [12.48%]	INSTITUT [12.40%]
AT	5039	24	27	HEALTH [27.96%]	EDUCATION [21.25%]	FINANCE [19.59%]	INSTITUT [18.81%]	NEWS [6.53%]
HR	1386	25	29	HEALTH [24.53%]	SOCIETY [20.35%]	EDUCATION [19.41%]	FINANCE [16.45%]	INSTITUT [13.42%]
ES	7417	24	24	INSTITUT [28.80%]	HEALTH [27.81%]	EDUCATION [16.19%]	NEWS [12.98%]	FINANCE [10.57%]
IT	16134	27	28	INSTITUT [26.86%]	NEWS [20.93%]	HEALTH [18.96%]	EDUCATION [16.53%]	FINANCE [11.41%]
GR	2908	29	40	NEWS [39.55%]	INSTITUT [17.26%]	EDUCATION [15.68%]	HEALTH [14.24%]	FINANCE [9.22%]
SI	2144	28	28	SOCIETY [24.86%]	HEALTH [23.55%]	FINANCE [18.28%]	EDUCATION [16.98%]	INSTITUT [10.91%]

The sorted insecure percentages by country and sector are given in Table 19. Percentages were calculated for each country for each sector. For example, in Slovenia there were 16.98 percent classified websites in the education sector and 55.49 percent of those were found to be insecure. The last two columns in Table 19 indicate the percentage of insecure websites from unclassified websites (others) and the overall percentage of insecure websites found per country.

Table 19: Sorted percentages of insecure WP websites for each of the five fields, as well as for the group of others and the overall sample.

	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	others_insecure	insecure
SI	EDUCATION [55.49%]	SOCIETY [44.09%]	INSTITUT [40.60%]	FINANCE [38.78%]	HEALTH [37.82%]	41.63	41.68
LV	EDUCATION [50.00%]	SOCIETY [50.00%]	HEALTH [41.67%]	INSTITUT [33.14%]	FINANCE [32.95%]	41.36	41.17
ES	EDUCATION [42.55%]	INSTITUT [42.23%]	HEALTH [41.40%]	SOCIETY [41.11%]	FINANCE [36.86%]	37.47	37.66
GB	EDUCATION [39.42%]	HEALTH [39.15%]	FINANCE [38.32%]	INSTITUT [36.79%]	SOCIETY [35.91%]	37.24	37.32
AT	EDUCATION [36.41%]	HEALTH [35.84%]	SOCIETY [33.90%]	INSTITUT [33.76%]	FINANCE [29.48%]	35.18	35.09
NL	EDUCATION [35.08%]	HEALTH [34.81%]	FINANCE [34.05%]	SOCIETY [34.03%]	INSTITUT [33.22%]	33.07	33.15
DE	EDUCATION [32.36%]	INSTITUT [31.50%]	FINANCE [31.05%]	HEALTH [30.25%]	SOCIETY [28.99%]	31.45	31.38
NO	EDUCATION [32.25%]	INSTITUT [31.53%]	FINANCE [31.16%]	SOCIETY [30.77%]	HEALTH [30.36%]	30.73	30.75
LT	FINANCE [59.86%]	INSTITUT [47.97%]	EDUCATION [44.81%]	HEALTH [42.25%]	SOCIETY [31.58%]	47.66	47.37
IS	FINANCE [51.22%]	NEWS [36.67%]	HEALTH [32.65%]	INSTITUT [29.58%]	EDUCATION [28.72%]	37.27	36.81
IE	FINANCE [46.14%]	EDUCATION [45.18%]	HEALTH [42.75%]	INSTITUT [42.00%]	SOCIETY [38.14%]	43.23	43.27
EE	FINANCE [41.31%]	HEALTH [41.27%]	INSTITUT [38.68%]	EDUCATION [37.07%]	NEWS [26.00%]	41.05	40.82
FR	FINANCE [41.09%]	INSTITUT [38.99%]	SOCIETY [38.66%]	EDUCATION [36.33%]	HEALTH [35.28%]	37.57	37.61
CH	FINANCE [35.83%]	HEALTH [34.65%]	EDUCATION [34.16%]	INSTITUT [31.77%]	SOCIETY [30.19%]	32.90	32.96
BG	HEALTH [49.42%]	INSTITUT [47.89%]	EDUCATION [38.60%]	FINANCE [37.43%]	SOCIETY [35.29%]	43.73	43.44
IT	HEALTH [43.87%]	EDUCATION [43.49%]	SOCIETY [43.41%]	FINANCE [41.39%]	INSTITUT [41.33%]	41.20	41.11
RO	HEALTH [41.02%]	FINANCE [40.53%]	SOCIETY [39.73%]	EDUCATION [39.64%]	INSTITUT [39.22%]	41.63	41.22
SE	HEALTH [33.24%]	INSTITUT [32.68%]	FINANCE [32.24%]	EDUCATION [32.24%]	SOCIETY [26.49%]	33.49	33.38
CY	INSTITUT [50.00%]	SOCIETY [50.00%]	FINANCE [37.23%]	EDUCATION [22.22%]	HEALTH [22.22%]	40.59	39.24
PL	INSTITUT [47.13%]	EDUCATION [46.22%]	HEALTH [44.02%]	FINANCE [43.74%]	SOCIETY [42.55%]	44.72	44.66
LU	INSTITUT [46.15%]	SOCIETY [40.62%]	HEALTH [40.38%]	FINANCE [40.00%]	EDUCATION [39.73%]	39.90	39.85
CZ	INSTITUT [45.84%]	EDUCATION [42.19%]	SOCIETY [39.85%]	HEALTH [39.31%]	FINANCE [38.81%]	43.09	42.91
SK	INSTITUT [41.32%]	SOCIETY [40.79%]	FINANCE [40.32%]	EDUCATION [37.12%]	HEALTH [36.10%]	40.14	40.04
BE	INSTITUT [38.50%]	EDUCATION [37.90%]	HEALTH [37.13%]	SOCIETY [35.71%]	NEWS [35.31%]	35.98	36.03
FI	NEWS [40.62%]	EDUCATION [32.88%]	SOCIETY [31.91%]	FINANCE [29.95%]	HEALTH [29.75%]	31.85	31.78
DK	NEWS [39.40%]	SOCIETY [31.37%]	INSTITUT [31.05%]	HEALTH [30.41%]	EDUCATION [29.21%]	30.26	30.31
HR	SOCIETY [50.71%]	INSTITUT [43.01%]	HEALTH [41.76%]	EDUCATION [39.41%]	FINANCE [34.65%]	42.90	42.79
GR	SOCIETY [44.07%]	EDUCATION [43.86%]	HEALTH [43.72%]	FINANCE [43.28%]	INSTITUT [42.63%]	42.07	41.78
HU	SOCIETY [43.48%]	EDUCATION [40.79%]	FINANCE [39.14%]	INSTITUT [38.02%]	HEALTH [35.73%]	39.85	39.64
PT	SOCIETY [42.42%]	EDUCATION [42.13%]	HEALTH [40.99%]	INSTITUT [40.52%]	FINANCE [36.62%]	35.83	36.12

In the financial sector, the highest percentage of insecure websites was found in Lithuania (59.86%), for healthcare, in Bulgaria (49.42%), and among institutional websites, in Cyprus (50%). The news sector had the next-highest percentage in Finland (40.62%), and the society sector with the highest value was found in Croatia (50.71%).

Within the whole sample and taking into the account only Rank 1 in Table 19, the less secure web services with WP applications are in education - they were found in Slovenia, Latvia, Spain, Great Britain, Austria, Netherlands, Germany, and Norway. Web services offered in finance have the highest share of insecure websites from the studied sample in Lithuania, Iceland, Ireland, Estonia, France, and Switzerland. The largest share of insecure websites from the health domain was found in Bulgaria, Italy, and Sweden. The institute sector had the largest percentage of insecure websites in Cyprus, Poland, Luxemburg, Czech, Belgium, and Slovakia. For the field of news, the largest percentage of insecure websites was found in Denmark and Finland, while societies had the largest number of insecure websites in Croatia, Greece, Hungary, and Portugal.

Figure 25 presents the results in the form of clusters. The presence of insecure WP websites among the examined sectors is coloured differently, depending on the percentage of insecure websites. Green squares indicate a lower percentage of insecure websites, red squares indicate a higher number of insecure websites. Clustering by sectors shows that the pair of Finance and Institute as well as Education and Health have a similar distribution per country. Clustering per countries shows two big groups with similar distributions across all sectors. The first group includes Estonia, Iceland, Denmark, Finland, Netherlands, Austria, Germany, Norway, Sweden, and Switzerland, with lower percentages of insecure websites in all sectors, while other countries with higher percentages of insecure websites are in the second group.

The small percentage of websites from the news sector can be explained with the fact that this sector uses its own commercial platforms, consequently the number of found websites with WPCMS applications in the news sector was small.

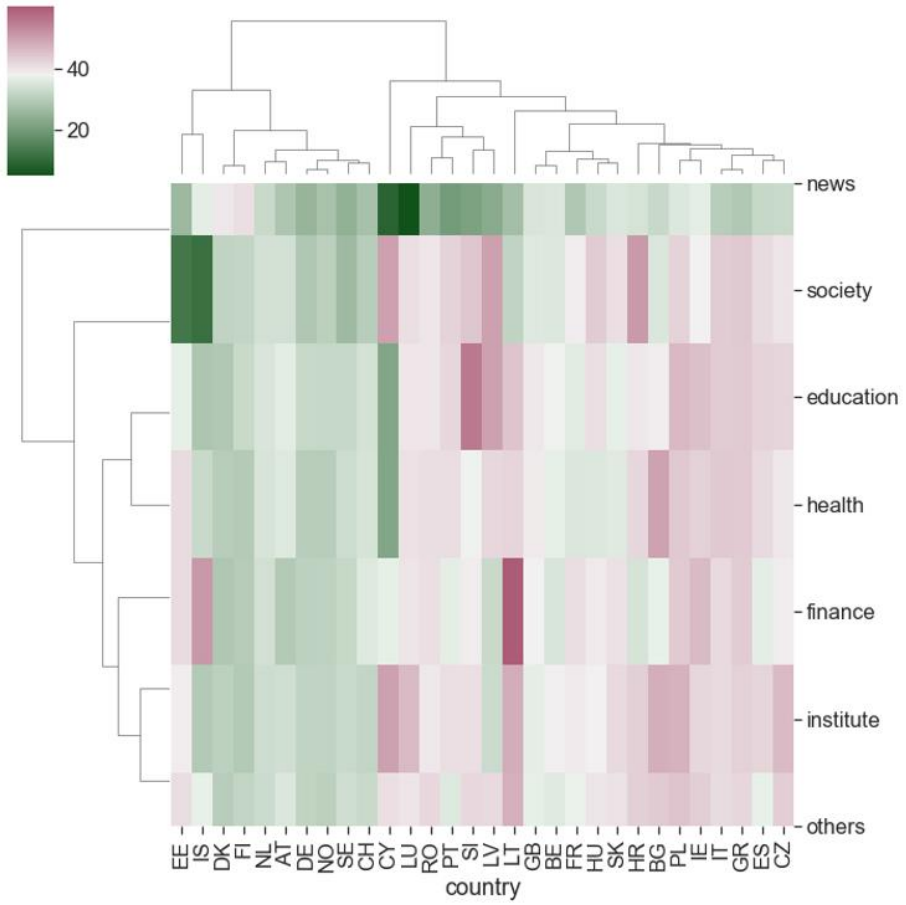

Figure 25: Content-based website insecurity in European countries

The Levene test for equality of variance ($statistic = 0.48$, $p - value = 0.491$) failed to reject the null hypothesis of equal variances, therefore the one-way analysis of variance to analyse the effect of sector type (news, education, health, institute, finance, society, others) on percentage of insecure websites was applied in the study.

A one-way variance analysis for analysing the effect of sector/field type (news, education, health, institute, finance, society, others) on the percentage of insecure websites gave a significant result, $F\,(6,\ 203) = 7.92$, $p - value = 0.000$, with an effect size $\omega = 0.17$. Tukey's HSD test was used to determine which field (or sector) had a significantly lower average percentage of insecure websites, and it was found this was the news sector ($p = 0.001$), the one with the lowest number of insecure websites. No statistically significant differences were found among the other groups, despite a slightly larger number of insecure websites in education and finance.

The rest of the websites in the sample that were not classified under any of these sectors is considered as the group "others". Descriptive statistics for each group are in the Table 20.

Table 20: Summary statistics of the percentage of insecure WP websites for each sector.

					95% CI	
SECTOR	**N**	**Mean**	**SD**	**SE**	**Lower**	**Upper**
EDUCATION_insecure	30	38.58	6.72	1.23	36.14	41.03
FINANCE_insecure	30	38.25	6.56	1.2	35.86	40.63
HEALTH_insecure	30	37.64	5.68	1.04	35.57	39.71
INSTITUT_insecure	30	38.88	6.04	1.1	36.69	41.08
NEWS_insecure	30	29.05	7.91	1.44	26.17	31.93
SOCIETY_insecure	30	36.69	9.12	1.66	33.37	40.01
OTHER_insecure	30	38.5	4.61	0.84	36.82	40.18

As can be seen in Table 20, there is a similar mean number of vulnerable websites of all sectors. No significant difference was found regarding the percentage of insecure WPCMS websites between almost all sectors, with one exception in the news sector, so Hypothesis (H3) cannot be confirmed.

5.5 Shared Hosting Platform and the Risk Factor for Compromising.

The Internet helped to facilitate access to applications developed by custom software applications for their customers. Throughout this process, there is an element that plays a very important role in creating efficient and effective web applications. This element behind the scenes is the web server which plays a significant role in deciding how to display the website content on user devices.

In the early days of the Internet, a web server could be defined as a server that can run a website by returning an HTML file over an HTTP connection. However, with complex applications becoming more common, the definition and the components of a web server have changed and a web server is today understood as any Internet server that responds to HTTP requests and offers content and services that can be handled by a web client. Based on W3Tech statistics, the web server market is dominated by three types of web servers, namely: Apache, NGINX, and Microsoft IIS (W3Tech, 2019).

The data collected with Vulnet made it possible to find out whether hosting multiple websites on the same server has an impact on the overall security of a site. The analysis of collected data did not confirm that hosting is an issue increasing insecurity. The Vulnet tool found the presence of the following known web servers: Apache, Microsoft, Lifespeed, Nginx, Ats, Openresty, Flywheel, Cloudflare. The unidentified servers were grouped as "other". Cloudflare websites (3%) were dropped from the set, as it was not possible to identify their real host provider.

Figure 26 presents the ratios between servers (number of websites per server) for individual countries in Europe. The collected data revealed that the most used web server types within the WP websites are Apache and Nginx. These two web servers are followed by Microsoft and Litespeed web servers. In North America and Asia, the distribution of Apache and Nginx is proportional to the number of the overall web installations, while Apache is more frequently used in countries in Africa, South America and Oceania.

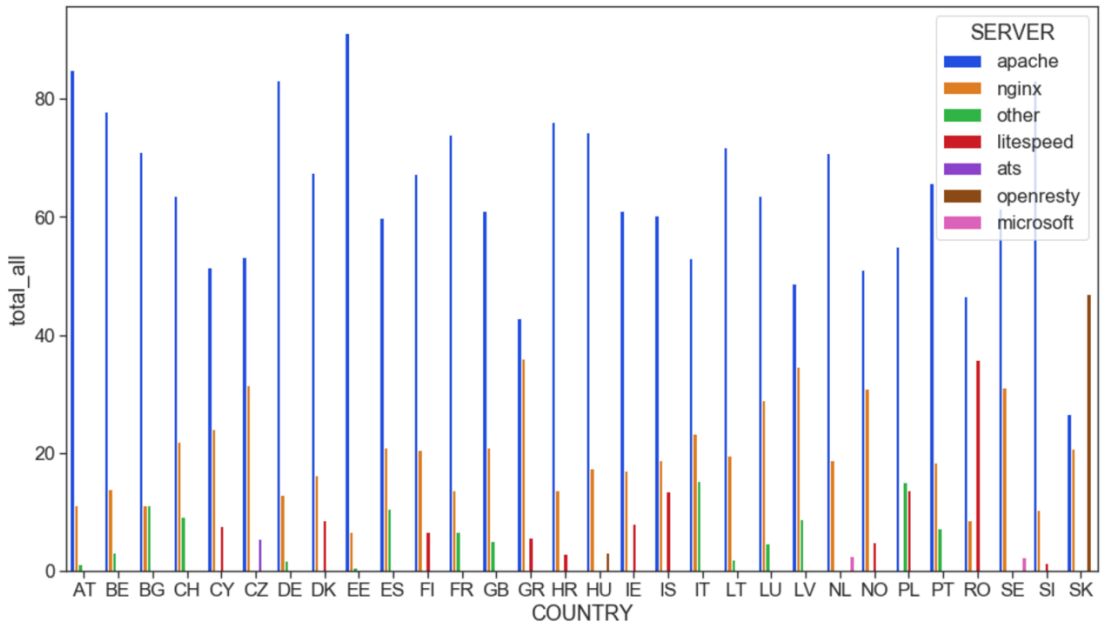

Figure 26: Server percentages per countries in Europe.

The presence of other web servers in these countries is rather low and can be neglected. The explanation is rather straightforward: Apache is an open-source web server. Apache has been developed and maintained by a team of developers under the auspices of the Apache Software Foundation. It is the result of joint efforts aimed to create a strong and secure web server of commercial quality that complies with all HTTP standards. Apache has been a market leader since entering the web server market in 1995 and remains the most selected web server choice because of its ability to run on multiple platforms.

The left pie chart in Figure 27a shows the summary of top three server choices per country in Europe for WP websites. The numbers in the pie chart represent the number of countries that had a specific web server as their first choice. In all countries except Slovakia, the most frequent type by far is Apache. Nginx is the second-ranked popular server. The third place was mostly occupied by server types that do not belong to any of the popular server types.

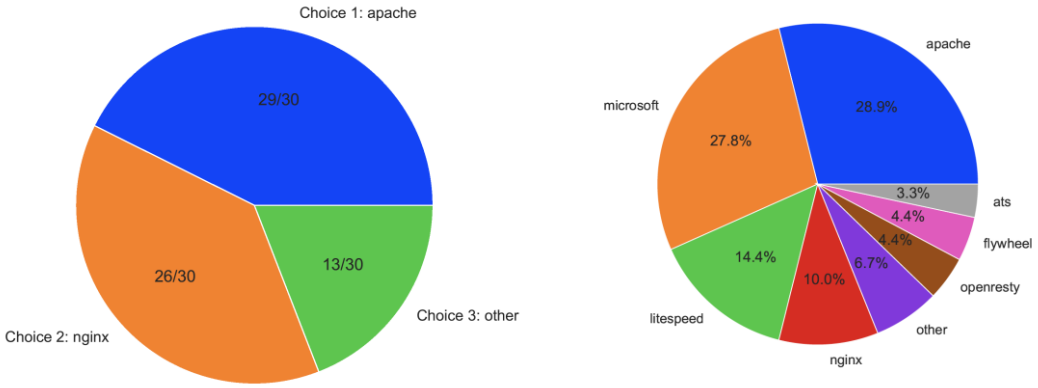

Figure 27: Pie charts for the top three servers per country (Figure 27a) and for server occurrences with the highest percentage of insecure WP websites (Figure 27b).

On the right in Figure 27b, the pie chart represents the percentage of server occurrences with the highest percentages of insecure websites (Appendix B2). In other words, the percentage of insecure websites was ranked by server type and country. After Apache (28.9%), which is the most popular and has consequently a larger number of insecure websites, the next most frequent type with high percentage of insecure websites across countries is Microsoft (27.8%), followed by Litespeed (14.4%).

In addition, the collected data from the WP web space of the European country set was analysed to find out how shared hosting of multiple websites on one server increases the risk of website insecurity. The collected data has shown that there is a large number of WP websites (3 368 334 of them) that share their host IP number but have different URL. The percentage of insecure WP websites for the set of websites on a shared host was found to be 35.8% and the number of different IP addresses present was 208,073. On the other hand, the number of WP websites with a unique IP was found to be 252,048, among them 99,425 were found to be insecure (39.5%). Bartlett's test (*statistic* = 0.286, *p*-value= 0.593) failed to reject the null hypothesis that the two samples' variances are equal. Therefore, a standard 1-tailed *t*-test ($t = 2.14$, *p*-value = 0.018) was used to compare the percentage of insecure WP websites for shared (*mean* = 39%) and unique (*mean* = 41%) hosting servers across all countries and rejected the null hypothesis of equal means in favour of the alternative that shared is lower than unique hosting servers.

A statistical comparison of the percentages of insecure WP websites between shared hosting servers and servers with unique IPs across countries shows that a shared hosting server does not remarkably increase the risk of a WP website vulnerability. A study by Tajalizadehkhoob showed that shared hosting mitigates the number of insecure websites on a shared IP, which was also confirmed in the study (Tajalizadehkhoob, et al., 2017). Web hosting providers try to eliminate the risk of attacks since multiple different valuable clients could be compromised. Therefore, the hypothesis (H4) cannot be confirmed, as running a WPCMS website on a shared hosting web server is not a positive risk factor for compromising the running applications.

5.6 Presence of Vulnerabilities and Running the Latest Core Version

The local database built within VulNet contains 430 core versions of WordPress, of which 361 were detected during the large-scale scan. The number of published core versions throughout the history of WordPress is presented in Figure 28. In the last six years, 90 critical vulnerabilities have been published, in the year 2014 there were only 19 new vulnerabilities published while in 2019 only 8 new vulnerabilities were published. This trend shows that although the number of new WordPress versions increases, the number of identified vulnerabilities decreases, which indicates that WordPress takes care to remove vulnerabilities by updating the versions with new ones that are less vulnerable.

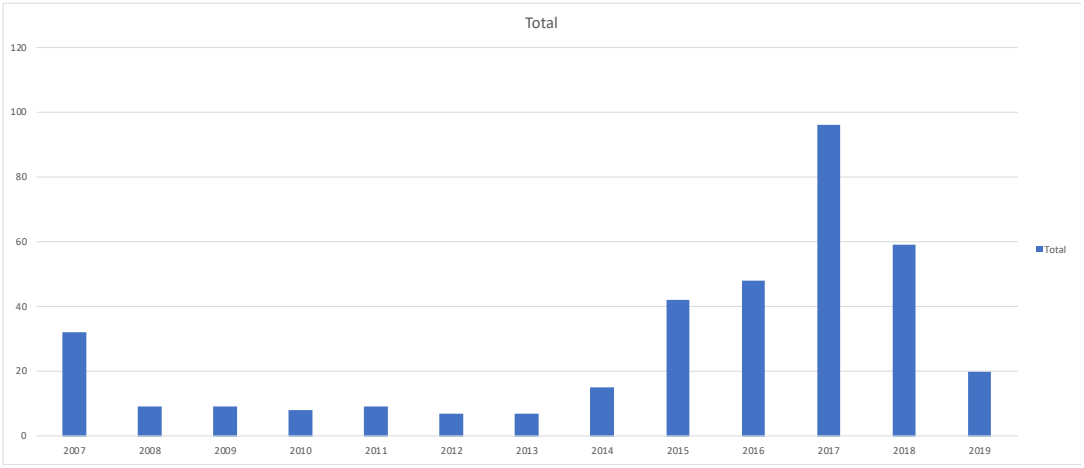

Figure 28: Number of published core versions throughout the history of WordPress.

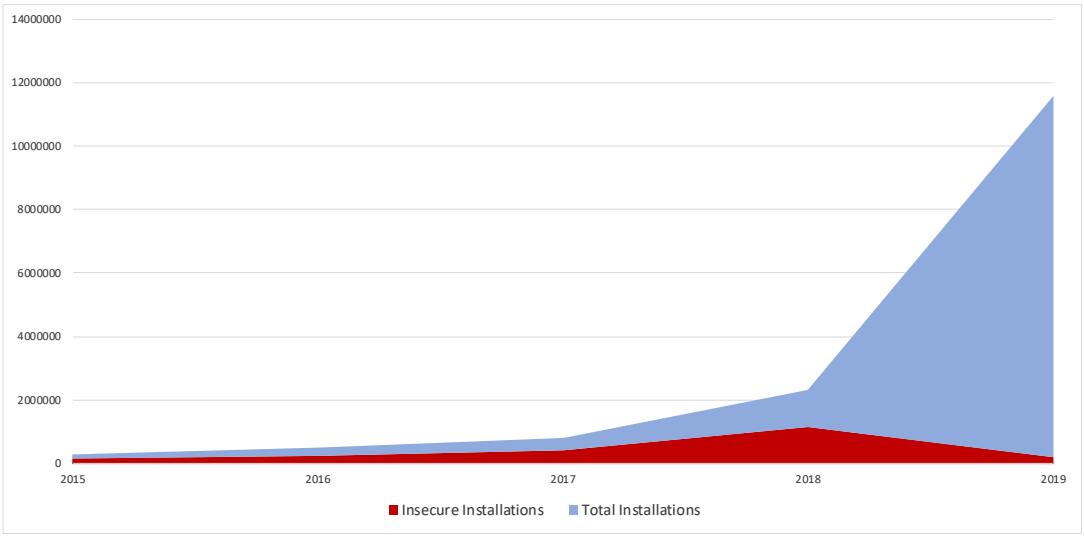

Figure 29: Comparison of total WP installations with total vulnerable WP installations sorted by years.

Figure 29 shows the number of installations that are vulnerable by WordPress core version in the whole world. It is evident that the number of vulnerable installations has dropped drastically compared to the number of total WordPress core installations on the Internet. All these installations are not older than 5 years which indicates that web owners are upgrading their platforms towards more security.

Table 21 presents the top ten versions (by November 2019) that show the highest number of installations, along with the number of vulnerable installations, the number of vulnerabilities, and the release date of the new WordPress core version. As WordPress releases security updates for older versions, several different core versions were released on the same day. For example, versions 4.9.10, 5.1.1, 5.0.4, 4.7.13, 4.8.9, and 4.6.14 are likely to include solutions for the same security issues. Version 5.2.2 is the most common installation with 3,194,399 identified WordPress installations. This core version was released on 18 June 2019 and had no vulnerabilities identified at the time of the scan performed with VulNet.

Table 21: WordPress top 10 versions with most installations.

Version	Total Installations	Total Insecure Installations	Disclosed Vulnerabilities	Release Date	Latest Major Version	Latest
5.2.2	3,194,399	0	0	18.06.2019	Yes	Yes
4.9.10	2,302,679	0	0	13.03.2019	Yes	No
5.1.1	1,572,594	0	0	13.03.2019	Yes	No
5.2.1	1,189,129	0	0	21.05.2019	No	No
5.0.4	843,781	0	0	13.03.2019	No	No
4.7.13	481,728	0	0	13.03.2019	Yes	No
4.8.9	472,699	0	0	13.03.2019	Yes	No
4.9.8	392,253	392,253	10	02.08.2018	No	No
4.9.3	208,518	208,518	15	05.02.2018	No	No
4.6.14	172,434	0	0	13.03.2019	No	No

Version 4.9.10 has the second highest number of installations with 2,302,679 detected unique WordPress installations and the release date 13 March 2019. The versions with most insecure installations are 4.9.8 and 4.9.3 with a total of more than 600,000 vulnerable installations. At the time of the VulNet large-scale scan, 25 vulnerabilities were detected in these two versions.

Figure 30 presents core versions released before the first VulNet scan that still have more than 2,581,613 active installations. Based on the results, versions 5.1, 5.0.3, 4.9.8, 4.9.3 are installed in a million websites. Version 2.8.4 is almost 11 years old with a release date of 12 August 2009 and still has 27 different vulnerabilities.

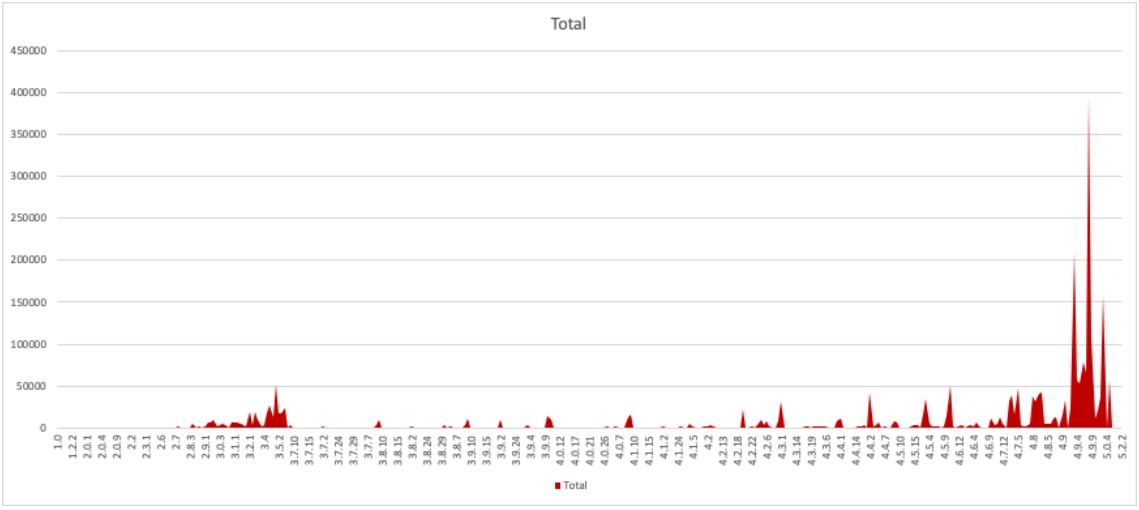

Figure 30: Total insecure WordPress installations in the world.

As presented in Table 22, version 4.9.8 has the most active insecure installations with 392,253 detected installations of the core versions released before 2019 with the release date of 2 August 2018. One of the top 10 most vulnerable versions is the 3.5.1 core version which is older than 6 years and has more than 30 disclosed vulnerabilities. Among the most vulnerable core versions is the 4th series version of the previous generation with most vulnerabilities and 961,909 installations.

Table 22: WordPress top 10 insecure versions with highest number of installations.

Version	Total Installations	Disclosed Vulnerabilities	Release Date	Latest Major Version	Latest
4.9.8	392,253	10	02.08.2018	No	No
4.9.3	208,518	15	05.02.2018	No	No
5.0.3	157,625	2	09.01.2019	No	No
4.9.9	106,276	1	13.12.2018	No	No
4.9.6	78,290	10	17.05.2018	No	No
4.9.7	66,189	10	05.07.2018	No	No
4.9.4	57,188	10	06.02.2018	No	No
5.1	56,640	1	21.02.2019	No	No
4.9.5	53,195	10	03.04.2018	No	No
3.5.1	52,093	30	24.01.2013	No	No

In their study, Vasek and Moore claim that some of the updated WPCMS versions are most vulnerable as they are not operational for long enough to allow discovering the new, yet not found vulnerabilities (Vasek & Moore, 2014).

The analysis of the collected data revealed that some websites have very vulnerable older core versions - especially the versions from series 4.0 appeared to be highly vulnerable (4.9.8, 4.9.3, 5.0.3, 4.9.6, 4.9.7, 4.9.4, 5.1, 4.9.5, 4.6.1), while the latest versions were much safer (5.2.2, 5.2.1, 5.2, 5.1.1) representing 6,065,968 secure WordPress core installations, but with one drawback, the plug-in vulnerabilities were not taken into account. These findings contribute to the view that recent versions of the software are safer, as the most vulnerable WCMS are those with outdated software. Even though current results reveal that newer versions are supposed to be more secure, we should keep in mind that new potential vulnerabilities have not yet been discovered in them. Four months after re-examining the WP core vulnerabilities, new vulnerabilities were discovered for the latest WP version (5.2.2).

5.7 Impact of the Country's Level of Digital Development on the Level of WP Website Insecurity

The comparison of the level of digital development with the level of web insecurity among the set of 30 European countries was intended to discover whether different parameters that measure the digital development within the complex social index known as Digital Skills (DS) have an impact on the WP web security situation in a particular country. As part of the "European 2020 Strategy", in 2016 the European Commission introduced a performance measurement system known as DS to track the evolution of EU member states in digital competitiveness (EU-Commision, 2016).

Digital Skills (DS) among the population of a particular country or region are considered as one of the measures that provide a view on both economic and social factors, such as human capital potential and e-services usage by the population that influence the digital development. DS was used as an indicator of the digital development for first time in 2015, and from 2016 on new data are provided each year by Eurostat (Eurostat, 2019). The DS index includes the skills that are needed to obtain and manipulate digital information, for communication and the use of Internet services including smartphones, problem solving (on-line work, shopping, etc.) and software skills (word processing, document creation, maintaining and processing, access to the e-services, etc.). Eurostat recognizes three levels of DS, low, middle (basic) and high (above basic). The following is

a list of the basic "activity indicators" used to calculate the DS index and the criteria used to determine the skill level (Eurostat, 2019):

- Information skills – includes skills such as copying or moving files or folders, saving information from the Internet, obtaining information from public authorities or service websites, finding information about goods or services and searching for health-related information

- Communication skills – includes sending or receiving emails, participating in social networks, making and receiving phone calls or video calls over the internet and uploading self-created content to any website

- Problem solving skills

    - Problem solving (A) – includes skills such as transferring files between computers or other devices, installing software or applications, or changing settings of any software, including the operating system or security programs
    - Familiarity with online services (B) – covers online buying or selling, the usage of online learning resources and internet banking

- Software skills

    - Basic – use of word processing software, spreadsheets and editing software for photos, videos, and audio files
    - Above basic – covers creating presentations or documents integrating text, pictures, tables or charts, the use of advanced functions in spreadsheets to analyse data (use of formulas, charts, etc.) and to write a code in a programming language

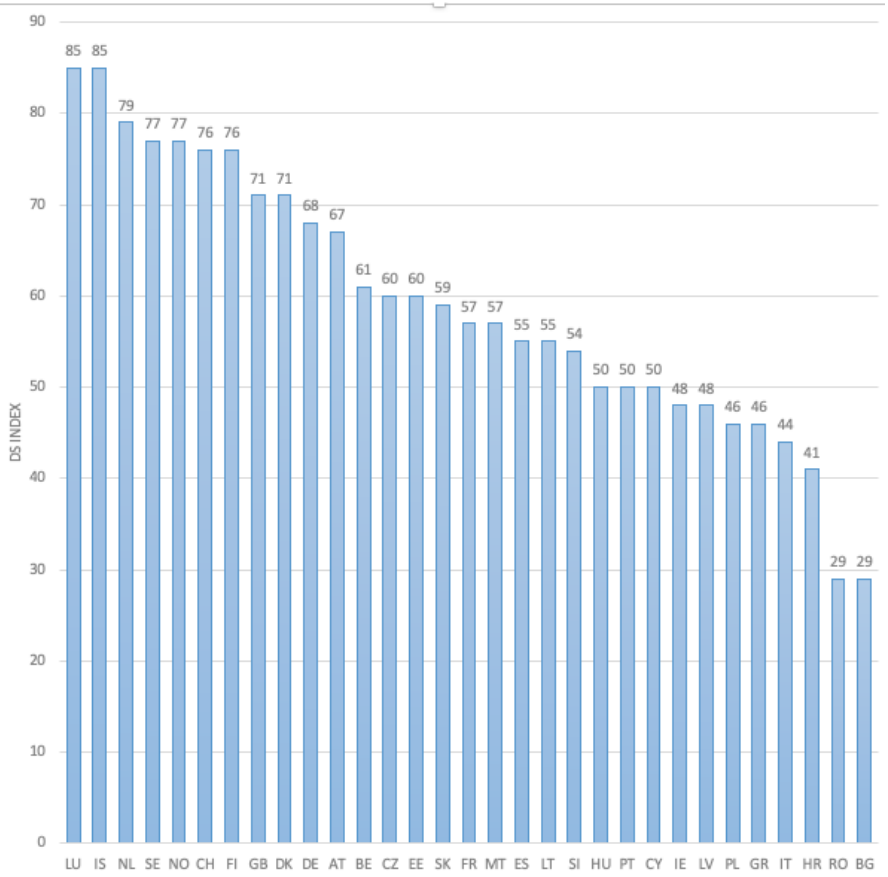

Figure 31: Individuals with basic or above basic overall digital skills in European countries (Eurostat, 2019).

The level of information skills and level of communication skills are rated as basic if one item is checked, and above basic if more than one is checked. The level of problem-solving skills is rated as basic if one or several items only from A or only from B are checked. Above basic is rated if at least one item from each A and B is checked. If none of the items from B is checked, the skill level for content creation is basic, and if one of the above-basic items is checked from B, then the skill level is rated as above basic. In the overall assessment of digital skills, there are individuals with an »above basic« level of skills who have an above-basic skill level in all 4 domains. Individuals with a »basic« level of skills have at least »basic« in all 4 domains. For individuals with a »low« level of skills, there are one or more skills that were not checked in one of the three domains, while individuals with no skills have no items checked in all four domains, despite declaring that they have used the internet at least once during the last few months.

Regarding the importance of website security for the digital development and the market in a country, it was decided in this study to explore different indexes to find out whether they have also an impact on the security and data protection within websites built with WP. Indexes were collected by ITU and Eurostat. In particular, the following three indexes: Frequency of Internet use by the population (FoIU), number of households connected to the Internet (HH), and the Cost of fixed access to Internet normalized with GNI data (Gross National Income) (Appendix B3) were used in the study regarding the infrastructure development. These indexes are parameters that define the level of digital development for a country and are expected to have an influence on the awareness about the security of the services offered by the websites.

A statistical analysis of each of these indexes are highly mutually correlated. Due to very high correlations of UI and FoIU with DS, where the lowest correlation between DS and HH was r = 0.87, HH and FoIU were dropped and the Digital Skill index and the fixed access cost normalized with GNI (that reflects the infrastructure of the country) were used as independent variables in the analysis of the data collected representing digital skills and the infrastructure.

Figure 32 shows the relationship between the percentage of secure and insecure WP websites in a particular country with indication of the DS levels. A DS index higher than 75 is coloured in green, orange is used for a DS index between 75 and 50, and the lowest DS country index below 50 is coloured blue.

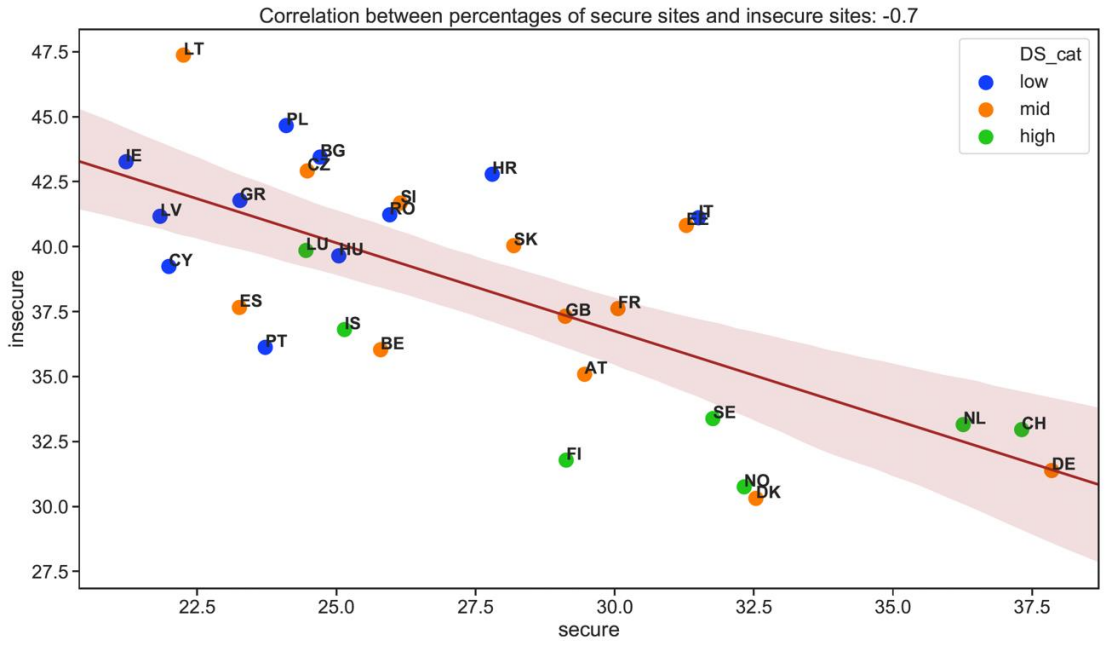

Figure 32: Percentages of secure vs. insecure WP websites per country with DS index and regression line.

The following countries with high digital skills – Finland, Sweden, Norway, the Netherlands, Switzerland – show a high percentage of secure websites and a low percentage of insecure websites; they are grouped in the bottom-right corner of Figure 32. Denmark and Germany are classified as countries with middle level of DS, but they are located at the top of the scale for group of countries with medium digital skills. In the group of countries with low digital skills and a high percentage of insecure websites, we can identify the following countries: Poland, Bulgaria, Greece, Latvia, Portugal, Ireland, Romania, Croatia, Hungary, Italy, and Cyprus. The group of countries with a medium level of digital skills and a moderate percentage of insecure websites includes Lithuania, Slovenia, Slovakia, the Czech Republic, Spain, Belgium, Austria, Great Britain, France, and Estonia.

A linear-regression analysis was carried out to model the percentage of insecure WP websites in a country based on the country's DS index. For a better understanding, we briefly present the regression method. A question in experimental science is how one set of variables affects other variables. Among the methods available to understand relationships, linear regression is the most commonly used method. The linear method summarizes the functionality and parametric relationship between variables. The linear method can distinguish two types of variables, namely independent and dependent variables. The variables where the desired value or a controlled value are set, are independent variables.

Therefore, we need to see how changes of the independent variables affect the values of the dependent variables. In linear regression, the values of one variable are predicted with respect to the values of the other variable. Linear regression is historically the first statistical method to model the relationship between the variable Yi and the independent variables Xi1, Xi2, … Xin as a linear equation (Hastie & Chambers, 1992):

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_n x_{in} + \varepsilon$$

where β_0 acts as a constant, $\beta_i, i = 1, \ldots, n$ act as coefficients of independent variables, and ε is the unexplained error. The coefficients are calculated from a predetermined system of linear equations by minimizing the total error, for which a normal distribution is assumed. Linear one-dimensional regression tries to find the most appropriate line through given points. The most fitting line is called the regression line.

The sample of the dataset from European countries was near-normally distributed and homoscedastic. A close correlation between digital skills and the percentage of insecure websites (r = -0.68) suggested a linear dependence. The first proposed regression model was:

$$insecure = 50.38 - 0.2 * \mathrm{DS}$$

Equation 2: The first proposed regression model.

with $R^2 = 0.463$, and RMSE = 3.25 was found to be significant, and $F\ (1,\ 29) = 24.11$ with ($p < 0.001$) as well as both coefficients being significant at $a = 0.001$. The analysis of the residuals, however, suggested improvements to be made for the model. Table 23 shows the results of the Jarque-Bera test for testing normality (Jarque & Bera, 1980), the Heteroscedasticity Test by Breusch-Pagan LM-statistic, (Breusch & Pagan, 1979) to test homoscedasticity, the Harvey Collier Test for Linearity (Harvey & Collier, 1977), and the Durbin-Watson test for first-order autocorrelation (Durbin & Watson, 1971) for the first and the second improved model. Improvements for a model were made based on the JB and DW statistics by removal of the residual outlier (LT), the three leverage points (IS, RO, BG), and one influential observation (LU). The largest residual outlier (LT), the influential observation LU and one leverage point IS, were removed from the dataset when deriving the improved model. The influential observation LU, (which is also a very small country with concentrated business entities from banking, finance, and commercial sectors where information security is very important) shows a very moderate skill level in the population, due to the fact that many employees within these sectors do not live in Luxembourg and are not captured in the population surveys. The leverage point, IS which is also a country with a very small population (357 000 inhabitants), but with the highest number of internet users, close to 100%, was also removed from the dataset to improve the model. These observations were found to be the same as those observations that were significant in the bivariate outlier analysis of the DS and the insecure websites using the Mahalanobis distance. The new model has the following regression equation:

$$insecure = 52.64 - 0.25 * \mathrm{DS}$$

Equation 3: The new regression model.

Table 23: Residual analysis.

model	Residual analysis								
	skew	**kurtosis**	**JB**	**Prob (JB)**	**LM-statistic**	***p*-value (LM)**	**HC-statistic**	***p*-value (HC)**	**DW**
1: insecure ~ DS	0.529	2.964	1.403	0.496	1.9532	0.1622	1.5631	0.1301	1.501
2: insecure ~ DS	0.134	2.518	0.342	0.843	0.0695	0.7921	0.5546	0.5846	1.925

R^2 has increased to 0.67 and RMSE dropped to 2.48, see Table 24. The residual analysis supports all four assumptions of linear regression, as shown in Table 23. The regression analysis confirmed that a higher level of digital skills has a negative impact on the presence of insecure WP websites. It is assumed that a population with a higher DS index is more aware of the potential security risks among users and web owners and has less vulnerable WP websites. These facts lead to the conclusion that the DS of a country's population is a negative factor for web insecurity.

Table 24: Regression results summary.

model	regressors	Regression results summary											
					95% CI				**Goodness-of-fit**				
		coef	**SE**	**beta**	**lower**	**upper**	**residuals**	***F*-statistic**	***p*-value**	**AIC**	**BIC**	R^2	**RMSE**
1: insecure ~ DS	const	50.38	2.52		45.22	55.55	28	24.11	< 0.001	160	162.8	0.463	3.25
	DS	-0.2	0.04	-0.68	-0.29	-0.12							
2: insecure ~ DS	const	52.65	2.11		48.30	57.00	25	50.68	< 0.001	129.7	132.3	0.67	2.48
	DS	-0.25	0.04	-0.82	-0.33	-0.18							

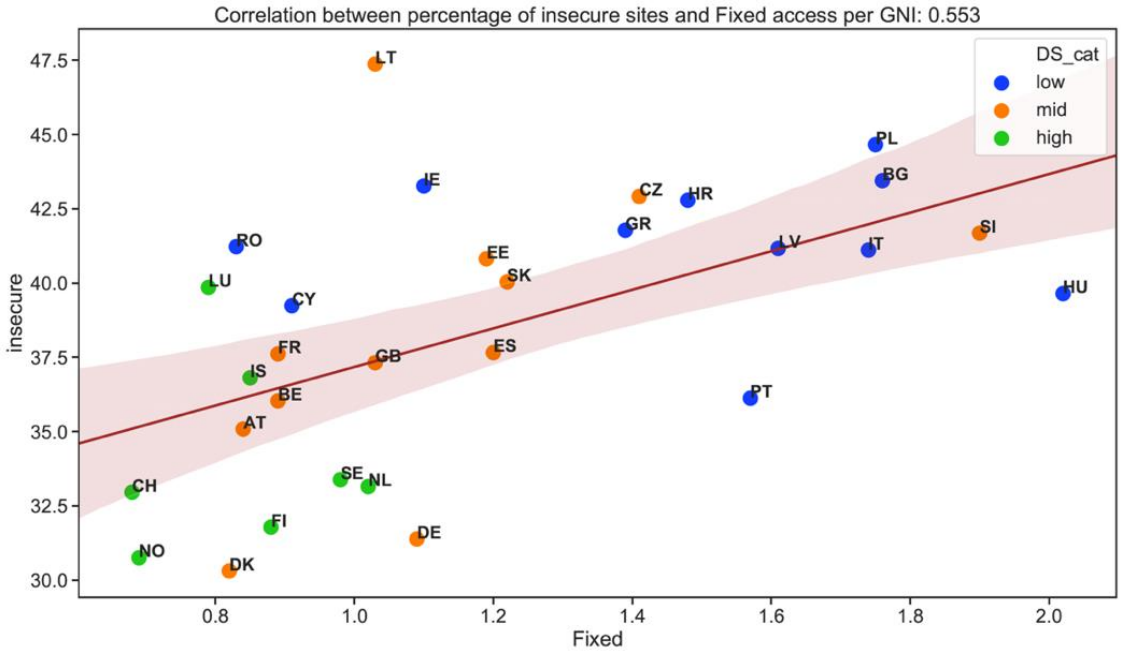

Figure 33: Correlation between percentage of insecure websites and fixed-cost access per GNI

The correlation of the fixed-cost access to the internet normalized with the country's GNI as shown in Figure 33 provides another insight into the correlation affecting the presence of insecurity. Where the fixed-access cost is lower, the percentage of insecure websites is also lowest, and the countries that belong to this group have the highest DS level or a medium DS level, similar to the group in Figure 32. This group includes the following countries: Denmark, Germany, Switzerland, Norway, Finland, Sweden, the Netherlands, and Austria. The other group of countries with a much higher cost of fixed access has a higher percentage of insecure websites. These are: Hungary, Bulgaria, Italy, Croatia, Slovenia, Poland, Latvia, Romania, the Czech Republic, Estonia, Italy, Iceland, Luxembourg, and Belgium. These findings are also in line with the linear dependency between the level of DS and the percentage of insecure websites (r = -068) and imply that a low fixed access price for the internet is a positive factor for higher security.

The analysis of the studied sample leads to the conclusion that European countries differ significantly regarding the level of security among all WPCMS websites. Leading countries with a lower percentage of insecure websites are Germany, Denmark, Austria, Sweden, Belgium, Finland, Netherland, Switzerland, and Norway.

The level of digital skills in the country's population has the dominating role in this distribution. The statistical analysis has shown also that there is a linear relationship between the percentage of insecure websites and the DS index. A higher DS index implies a lower number of vulnerable WP websites. Some noticeable bivariate statistical outliers were Lithuania, Luxemburg, Iceland, Romania, and Bulgaria. The latter two are the ones with the lowest DS index and some of the highest percentage of insecure websites, while Iceland and Luxemburg both have a DS index of 85, but also have a high percentage of insecure websites. Possible explanation may lay in their very small population.

Digital skills as an indicator of the knowledge present among a country's population and the affordability of internet access were found to be the dominating factors in shaping the level of insecurity in a particular country. Awareness and knowledge about security in setting up and using web services obviously impact website owners as well web service users to take care about their websites in prevention from cyber-attacks. Although both indexes used have clearly shown the impact of the insecurity present in the websites, other indexes related to the digital advancement may also confirm these findings.

5.8 The Second Run of the Tool and the Assessment of the WP Web Space Security

The first large-scale scan exercise was repeated 4 months later with the aim to find out whether the web space has changed in the meantime in terms of WPCSM website security. The purpose of the research was to verify whether web owners have changed their attitudes regarding the maintenance and updating of websites. VulNet was updated to identify whether there were any changes in each of the previously accessed and inspected websites. With a replacement of the core and the plug-ins or by introducing new secure plug-ins, a change of the overall status from insecure to secure and a change from an "unknown" website core to a known was expected. For that reason, VulNet was updated with a new feature that checks whether each website found was accessed as well in the first scan.

A Python script was written to compare the previous state with the new one by iterating over the first scan and verifying the state in the database generated in the second scan for each domain and compiling a merged json object with all parameters relevant to that domain.

The script verifies whether a domain name matches the domain name in the second scan database. During the match, all data of the accessed website domain are aggregated

and verified. Then a more complex approach is applied for a comparison of the core and plug-ins to identify any differences between the first and second scan. Table 26 below provides an example of a built JSON object between the first and the second scan, presenting a website that is located in Slovenia ("CC": "SI") and running on an apache server ("Ser.": "Apache"). Table 25 explains the meaning of individual parameters, where (2^{nd}) represents the second scan.

Table 25: Meaning of individual scan parameters.

Parameter	Meaning of Parameter
P&C Secure	Plug-in and core security: 0 = not secure, 1 = secure
P&C Score	Total danger score from both plug-in and core
C Secure	Core security: 0 =not secure, 1 = secure
C Score	Core danger score
C Version	Detected core version
C Latest	Core is latest version: 0 = not true, 1 = true
t(n) = latest	Number of versions till the latest
C(n) 1^{st} <-> 2^{nd}	Number of core versions between first and second scan
P MAX secure	All plug-ins are secure, 0 = not secure, 1 = secure
P MAX score	Highest danger score from all plug-ins' danger scores
P Version	Plugin version
plug-ins: [{ latest }]	Plug-in is latest version: 0 = not true, 1 = true
plug-ins: [{t(n) = latest }]	Number of plug-in versions till latest
P Secure	Plug-in security: 0 = not secure, 1 = secure
P Score	Plug-in danger score
TYPE	3 types: • NEW (plug-in detected only in the second scan), • EXISTING (plug-in detected also in the second scan), • DELETED (not detected in second scan).

Table 26: Example of a built JSON object between the first and the second scan.

JSON object

```
{
    "id": "#####.si",
    "CC": "SI",
    "Ser.": "apache",
    "P&C Secure": 0,
    "P&C Secure (2nd)": 0,
    "P&C Score": 7,
    "P&C Score (2nd)": 7,
    "C Secure": 0,
    "C Secure (2nd)": 0,
    "C Score": 7,
    "C Score (2nd)": 7,
    "C Version": "4.7.3",
    "C Version (2nd)": "4.7.3",
    "C Latest": 0,
```

```
        "C Latest (2nd)": 0,
        "t(n) = lastest": 36,
        "t(n) = lastest (2nd)": 47,
        "C(n) 1st <-> 2nd": 11,
        "P MAX Secure": 0,
        "P MAX Score": 4,
        "P MAX Secure (2nd)": 1,
        "P MAX Score (2nd)": 0,
        "plug-ins": [{
                "Slug": "tablepress",
                "Rating": 100,
                "Downloads": 4485166,
                "P Version": "1.8.1",
                "P Version (2nd)": "1.8.1",
                "Latest": 0,
                "Latest (2nd)": 0,
                "t(n) = latest": 3,
                "t(n) = latest (2nd)": 3,
                "P Secure": 1,
                "P Secure (2nd)": 1,
                "P Score": 0,
                "P Score (2nd)": 0,
                "TYPE": "EXISTING"
        }, {
                "Slug": "easy-fancybox",
                "Rating": 92,
                "Downloads": 3086643,
                "P Version": "1.5.7",
                "Latest": 0,
                "t(n) = latest": 18,
                "P Secure": 1,
                "P Score": 0,
                "TYPE": "DELETED"
        }, {
                "Slug": "fancybox-for-wordpress",
                "Rating": 86,
                "Downloads": 1314200,
                "P Version": "1.3",
                "Latest": 0,
                "t(n) = latest": 37,
                "P Secure": 0,
                "P Score": 4,
                "TYPE": "DELETED"
        }]
}
```


The overall vulnerability assessment and vulnerability score was obtained for both the first and the second review of the website (P&C Secure: 0, P&C Secure (2nd): 0). The scan shows that the overall vulnerability score remains 7 (P&C Score: 7, P&C Score (2nd): 7). This is followed by a core dissection, where the vulnerability state is given for the core

version in the first and the second scan (C Secure: 0, C Secure (2^{nd}): 0) and the vulnerability score for both scans (C Score: 7, C Score (2^{nd}): 7). The detected core version in the first scan (C Version: 4.7.3) and in the second scan remains the same (C Version (2^{nd}): 4.7.3). In addition to the core versions in the first (t(n) = last: 36) and second scan (t (n) = last (2nd): 47), the number of intermediate versions between the detected core version and the last core version is added, including the number of versions between the first and the second core inspection (C(n) 1^{st} <-> 2^{nd}: 11). The number of new versions increased by 11 between the first and second scan.

Plug-ins on the website were also inspected. Vulnerable plug-ins (P MAX Secure: 0) with the highest rated vulnerability level 4 (P MAX Score: 4) detected in the first scan have changed in the second scan as they were now secure (P MAX Secure (2^{nd}): 1). The website owner has removed the vulnerable plug-ins, which were labelled in the json object as "DELETED". The "EXISTING" state is kept for the case that plug-ins remain the same. If a new plug-in is detected, the plug-in label is “NEW”. This enables an effective monitoring of the websites.

The first and second scan retained the use of the TablePress plug-in, which is considered as one of the medium-popular plug-ins in the WordPress community. The popularity of plug-ins is illustrated by the number of downloads (Downloads: 4,485,166) and the community rating (Rating: 100). The user had the same version installed in the first (P Version: 1.8.1) and the second scan (P Version (2^{nd}: 1.8.1), but not the latest available version (Latest: 0, Latest (2^{nd}): 0). Similar to the core comparison, a record of intermediate versions and the number of versions up to the latest version for the plug-ins was recorded. From the currently detected version to the last updated version, there are three intermediate versions (t (n) = latest: 3, t (n) = latest (2nd): 3). If vulnerabilities are detected, the state of the plug-in and the vulnerability score are both determined in the same way as for the core. This plug-in was secure in both scans (P Secure: 1, P Secure (2nd): 1) with a risk score of 0 (P Score: 0, P Score (2^{nd}): 0).

12,865,441 WP websites were verified in the second scan and 10,330,577 among them retained the same vulnerability status. Approximately 80% of domains had the same status (secure, insecure, unknown) at both scans. Only 0.16% had implemented the latest core version at the second scan. Around 14% have an unknown version. The number of versions between the identified version and latest version ranges from 0 to 395. The maximum is defined by an outlier with the first version and release from 11 years ago. Websites were found on seven different types of servers (Apache, Nginx, Litespeed, Microsoft IIS, ats, Openresty, and Flywheel).

A detailed analysis of the websites status change was performed, and seven different switch states were defined. As presented in Table 27, 10,330,577 websites retained the same version of the second scan.

The I2S switch represents a change of state from insecure to secure, 287,263 websites were identified here. The I2U state represents the switch from insecure to unknown, with 120,718 of installations detected. In the transition of the state from S2I (from secure to insecure), 478,784 installations were detected, which is almost twice as many changes of the state as detected in the I2S switch. A minor number of changes was detected in S2U (from secure to unknown) switch, with 87,859 installations identified. Among 1,560,240 transitions from unknown state to secure or insecure, 420,128 installations were detected in the transition from unknown to insecure, and 1,140,112 in the transition from unknown to secure (U2S).

Table 27: All combinations of status changes.

		size
switch	**switch_type**	
False	Stay	**10,330,577 (80.3%)**
True	I2S	287,263 (2.23%)
	I2U	120,718 (0.94%)
	S2I	478,784 (3.72%)
	S2U	87,859 (0.68%)
	U2I	420,128 (3.27%)
	U2S	1,140,112 (8.86%)

A comparison of the score vulnerability between the first and the second scan is shown in Figure 34, where majority of the websites is at 0 (secure websites) in both scans. We can see that the higher scores were in the second scan, as the number increased with a score of 6 and 7, representing the largest share of insecure websites found.

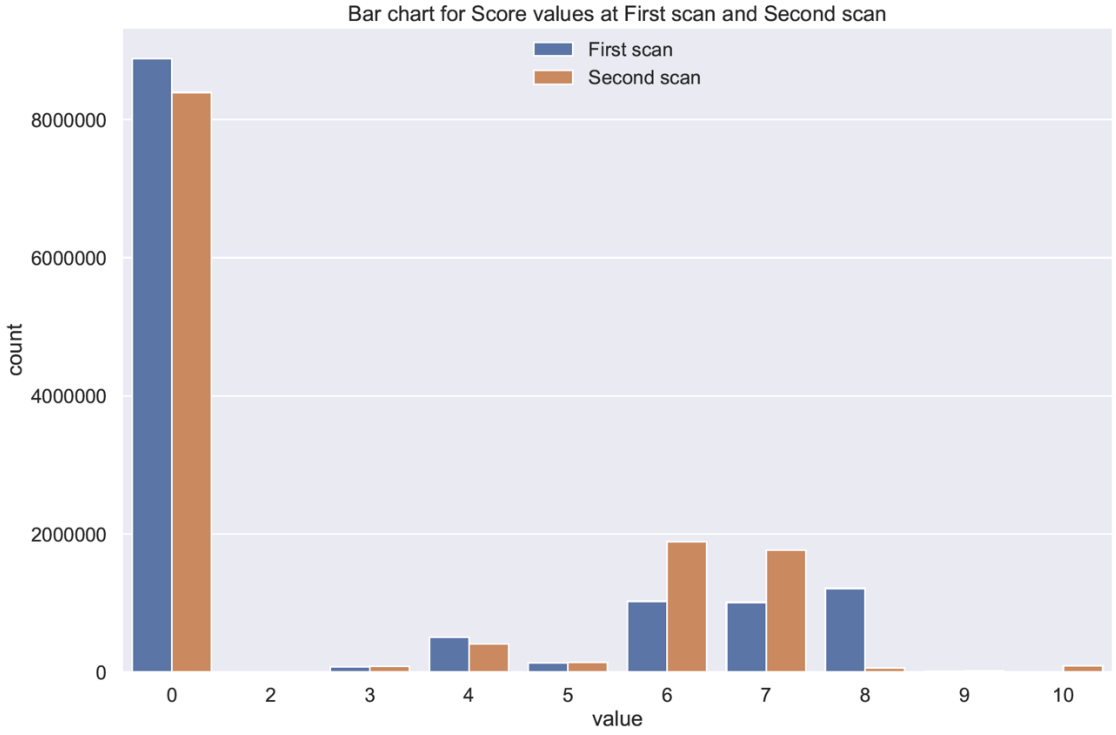

Figure 34: Comparison of overall score distributions between scans.

Aggregated values of each possible state (secure, insecure, unknown) for core, plug-ins, websites (overall) at both scans are presented in Table 28. The corresponding columns (StayS, StayI, StayU) present the percentages of each state in the second scan relative to the first scan (e.g., how much secure elements from the first scan remained secure in the second scan). It is visible that there are no significant deviations in the percentage states of both cores and plug-ins between the first and second scan. Nevertheless, some differences are noticeable, namely the percentage of secure cores has increased by 10 percentage points and the percentage of secure plug-ins has decreased by 5 percentage points.

Table 28: Aggregated values of all possible states from both scans.

	Total	Status1 S	Status1 I	Status1 U	Status2 S	Status2 I	Status2 U	StayS	StayI	Stay U
cores	395	49.89	15.30	34.82	59.95	19.62	20.43	44.86	13.75	19.65
plug-ins	8 813	79.98	19.95	0.08	74.69	25.25	0.06	72.26	22.68	0.05
sites	12 865 441	22.74	30.94	46.32	29.43	34.76	35.82	18.33	27.77	34.20

As shown in Figure 35a, the overall insecure status of the website is mainly determined by the insecure status of the plug-ins. The insecure status of the core version has less influence on the overall insecurity of the whole website. Figure 35b presents the overall security of websites. Around 20% of websites have secure cores and plug-ins in both scans.

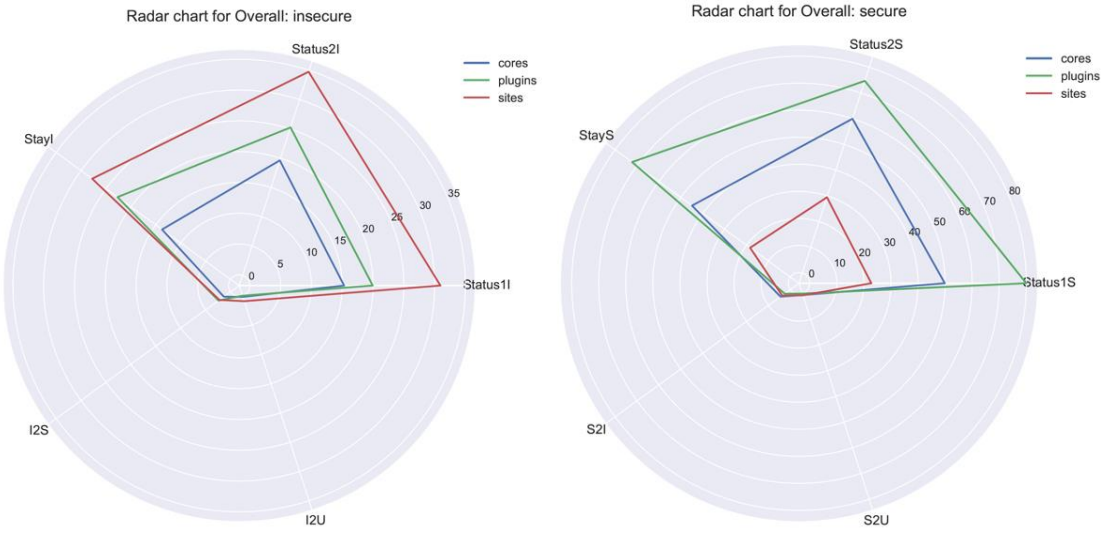

Figure 35: Overall insecure (Figure 35a) and secure (Figure 35b) cores, plug-ins and websites.

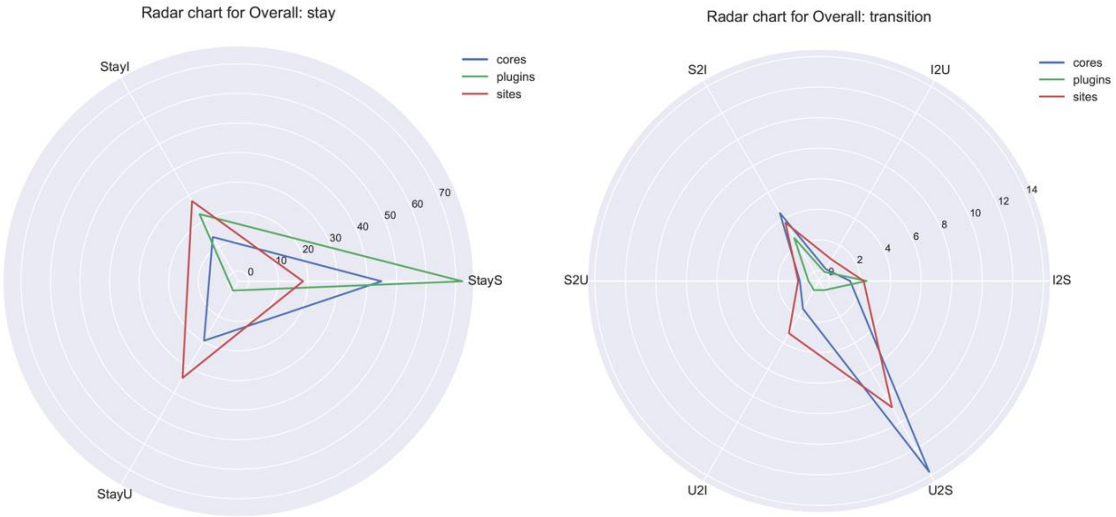

Figure 36: Overall stay (Figure 36a) and transition (Figure 36b) state.

Figure 36a (overall:stay) shows that the majority of plug-ins which were secure in the first scan stayed secure in the second scan too. This applies to around 40% of core versions. A combination of these two allows for the overall status of the site to be secure at both scans in only 20% of the cases. Figure 36b presents how the transitions (overall:transitions) for the core version happened mostly from unknown to secure. The shape for websites is defined mostly from the cores, as plug-ins transitions from secure to insecure occurred for only around 3%. For websites, there is approximately 4% of cases where the website became insecure at the second scan from either a secure or an unknown state.

When reviewing the core versions, we focused on the number of updates and the number of unchanged cores. The double pie shows the percentage of the status (secure, insecure, unknown) of the core version, according to the first (outer circle) and the second scan (inner circle). Each condition has its own colour in Figure 37: Percentage of all core data.

The majority of website cores were shown to be secure (50% and 58% on the first and the second scan, respectively). There is an increase of 4 percentage points in insecure versions on the second scan. Where the versions were unknown on the first scan, they either remained unknown on the second or became secure, with only 4% of them (1.4% of all cases) changing their status to insecure.

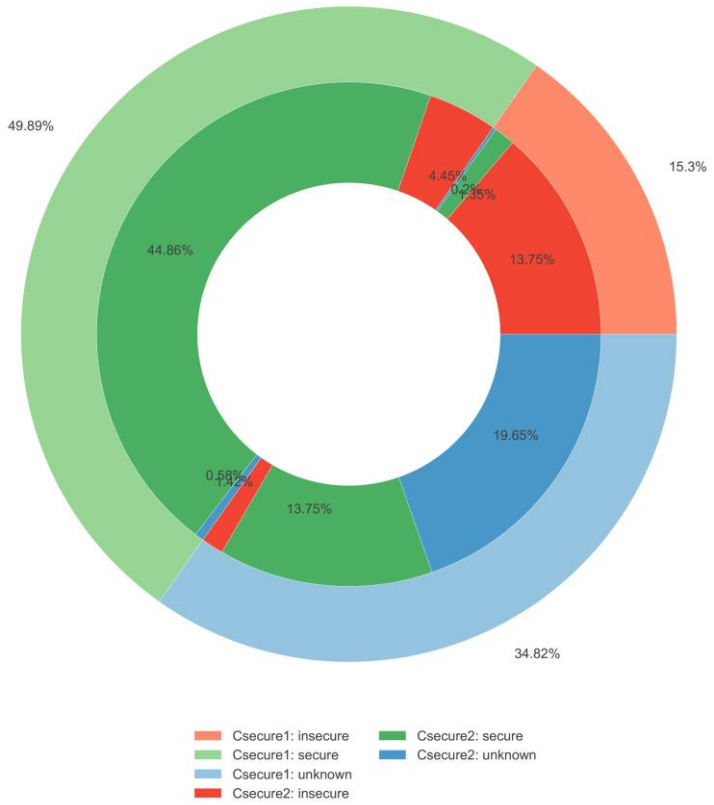

Figure 37: Percentage of all core data.

Some versions were changed, but not necessarily updated. Almost 56% remained unknown, while 40% have a secure version on the second scan. Most of the secure versions remained secure despite the change in version (those were mostly updated versions). There were 361 unique core versions on the first scan, while there were 395 unique core versions on the second scan and there were 6,004,652 detected updates of the core version in total.

Figure 38: Percentage of all cases in all existing data on plug-ins (Figure 38a) and percentage of all cases in updated existing data on plug-ins (Figure 38b).

The analysis has shown that the total number of secure cores in the second scan increased by 10 percentage points from the first scan. The total number of unknown states decreased by 14 percentage points due to an improved version detection algorithm.

As shown in Figure 38a, most unchanged plug-ins versions which were secure in the first scan, stayed secure in the second scan too. On the other hand, as shown in Figure 38b, the updates reduced the percentage of insecure existing plug-ins from 17% to 5%. Rarely, the version could not be identified (0.05% of cases).

A study by Ruohonen shows that deprecated plug-ins tend to increase the vulnerability counts (Ruohonen, 2019). The observation seems logical to them as the maintainers of the online portal may deprecate plug-ins with many unfixed vulnerabilities. Moreover, they found out that increasing lags in the update times tend to decrease the vulnerability counts as fixing multiple vulnerabilities requires more frequent updates. They also confirmed that their study supports earlier results. When comparing plug-ins with less than a hundred installations, the plug-ins with more than one hundred thousand online deployments had an about 0.455 higher probability of being affected by multiple vulnerabilities. Our study confirmed that finding, but on much higher number of inspected websites.

The number of vulnerable WP website installations fell sharply after May 2019, when the WordPress organization for the first time since the first release of the WordPress application in 2007, released an update of the core releases (core versions 1, 2, 3 and 4) that contained vulnerabilities. After rolling out the information about the size of the vulnerabilities in the Internet WPCMS space, the organization took care of all versions of the WordPress applications that were not updated previously with solutions that eliminated the identified vulnerabilities. WP's intention to keep their platform secure and the regularly provided releases based on automatic platform upgrades had a positive impact on the overall web security. This finding confirms the Hypothesis (H5) if the state of the cores is considered, as the data have shown that the trend of website insecurity with the WPCMS application decreases over time, which also indicates to a greater awareness about cyber security among the website owners. Similar care is needed for plug-ins as they play

a critical role in the overall security. As the number of vulnerable plug-ins was not changed, the insecurity among plug-ins did not change.

Chapter 6

Conclusions

Although large research efforts on the web application security have been ongoing for more than decade, the security of web applications continues to be a challenging problem. An important part of that problem derives from the vulnerability in the website core and the implemented plug-ins used to build the services. The WP web space over all continents was explored to detect the present vulnerabilities. The scan was carried out in two batches with the aim to find any potential changes and improvements in the provision of a more secure WP web space. Special attention in the exploratory study was paid to European countries in the search of parameters that impact the level of insecurity of the WP web space. The exploratory study was carried out with a newly developed tool with features that enable ethical, but reliable and very fast scanning.

With the study it become apparent that the current known services and tools for measuring the WCMS vulnerabilities lack to provide a deep insight into the security of websites operating with WordPress applications (Cernica, Popescu, & Tiganoaia, 2019). The effectiveness is low as they do not provide information about security holes within web applications supported by the installed plug-ins. WCMS owners need vulnerability information that involves multiple, attacker-controlled dimensions of possible exploits. The advantage of the presented results lies in the applied scanning approach that allows a fast and reliable overview of almost all accessible websites and by identifying the vulnerable ones.

Given the number of detected websites and the number of detected vulnerable cores and plug-ins, hypothesis (H1) was confirmed as it is possible to develop a tool for fast and reliable collection of vulnerable WPCMS websites at a large-scale on an ethical basis. Although the vulnerability could not be identified on 34% of WPCMS websites due to the use of core and plug-ins hiding, the tool could still identify all 34% of websites as a WPCMS platform.

The hypothesis (H2) was based on the assumption that vulnerabilities within the large Internet web space can provide information about risk factors for a higher presence of vulnerabilities and consequently insecurity in the web space. Factors that have a negative impact on WP security were identified, among them the presence of insecure plug-ins, which are less frequently updated manually, compared to core version updates, that can be updated automatically. In the subset of European countries, web spaces were inspected, and since it was possible to obtain parameters that measure a country's level of digital development, their impact on the higher presence of insecurities was analysed. Based on the findings that Digital Skills index and the fixed cost of Internet access normalised with GNI have impact on the appearance of different amounts of secure or insecure websites depend on their value for particular European country, hypothesis (H2) was confirmed. The Digital Skills index has a negative impact, meaning that the higher the index, the

lower the percentage of insecure WP websites; the cost of Internet access has a positive impact on the occurrence of a higher insecurity – the higher the cost, the higher the site insecurity percentage. However, the limitation of these results lay in the fact that the study was carried on European countries only, due to availability of reliable data, and was confirmed globally on data collected by means of a large Internet scan.

Hypothesis (H3) stated that that there were no significant differences in the presence of vulnerabilities within different types of web services deployed with WPCMS applications. The results obtained from the examined sample of seven sectors from the industry, health care and other socially important areas has confirmed that there are no significant differences in the percentage of insecure WP websites across different sectors for the whole sample, aside for the news applications.

Hypothesis (H4) assumed that shared hosting has an impact on the occurrence of insecurities. This hypothesis (H4) was not confirmed either, because running a WPCMS website on a shared hosting web server does not cause insecurity.

Website insecurity with the WPCMS application decreases over time and the integration of automatic updates has shown a decreasing trend in the overall percentage of insecure websites. Thus, Hypothesis (H5) stating that the insecurity of websites with the WPCMS application decreases over the time was confirmed. However, it should be noted that the most recent versions could be potentially vulnerable too, as they have not been operational for long time to allow discovering new, yet undiscovered vulnerabilities.

The results of this dissertation were supported by the application of statistical methods that have revealed correlations between the studied variables.

The study results concluded that the vulnerability of web application can be mitigated by trying to prevent the attacker before they reach a vulnerable application by notifying the owners about the existing insecurity and removal of the vulnerability. The VulNet tool can be one of the tools that can be used in this context to notify web site owners about a present vulnerability. While notifications about the website vulnerability appeared to be acceptable to a number of operators, however, the vast majority of the systems still remain unpatched because they ignore the notification received by e-mail. One reason why email notifications were unsuccessful lays in the anti-spam filters and in the lack of trust on the side of recipients. The explored alternative communication channels did not seem to offer any other promising solutions. Another way is to offer the tool as a public service to any individual web owner or web administrator to regularly monitor their websites while providing privacy to the owner and authentication of ownership. By entering the website URL and the owner's e-mail address, the platform developed with the Vulnet tool will send information about the inspection of this URL in terms of the presence of vulnerability. This platform functions enable vulnerabilities of the website to be revealed only to the owner, on the platform screen or with e-mail notifications. As a part of the future work, a public web portal with this service will be opened at end of 2021.

6.1 Summary of the Contributions

The scientific contributions of this dissertation that have governed the design of the applied tool can be summarized as follows:

- Advanced analysis and property evaluation of the current web vulnerability scanning tools for large Internet scanning and the quality and appropriateness of these properties for different types of scanning. Identification of missing features.

- A novel method and algorithms for identifying vulnerable WPCMS on the Internet at a large scale by following the ethical norms. The fast and ethical vulnerability detection features are based on the new methods beyond the state-of-the-art in the area of the world web space with WPCMS applications. The algorithms implemented in the tool outperform other competitors for large-scale scanning. The large-scale vulnerability identification provides information about the overall security of the web space with WPCMS all over the world.

- The next contribution is the score system for quantifying website vulnerability levels. The developed scoring mechanism calculates the website insecurity by identifying the WPCMS core vulnerabilities and the score for plug-ins. This is the first scoring vulnerability assessment method for WPCMS.

- The final contribution of this thesis are the scientific findings based on the analysis of the collected data. The impact factors for the presence of insecurities were revealed.

    - Risk factor analysis of plug-ins for WP websites.

    - Case study of the security state for different types of services provided with the WPCMS application. The result of the analysis shows that there is no significant difference in the percentage of insecure websites across different sectors of web services among European countries, with the exception of the news sector which differs from the others.

    - The influence on the security situation by shared hosting was evaluated in a large-scale scan. The analysis shows that there is no influence on the security situation when shared hosting is used to host the WPCMS website.

    - Trend analysis of the insecurity of WPCMS websites over time. Considering only the core versions, it was shown that the trend of the insecurity of WP websites decreases with the time, while the vulnerability of plug-ins remains the same.

    - The study results revealed that a correlation exists between the percentage and the level of the calculated insecurity score among the examined web space of European countries with the studied parameters. European websites using WPCMS with a higher DS rating among a country's population is reflected in a lower percentage of insecure websites from the analysed country's web space with WPCMS applications.

    - The lower cost of fixed Internet access normalized with GNI has a similar effect as the DS index. The percentage of websites that are insecure is lower within the country's web space with WPCMS applications where the cost of Internet access is lower.

    - The final contribution is the finding that ITU parameters, IU, FoIU, and HH of the European countries are highly correlated with the cost of Internet access normalized with GNI and consequently they have similar impact on the WPCMS web space security of the European countries.

6.2 Limitation of the Developed Tool

The developed tool is not intrusive, it can get the security of a website if the information of the core and plug-ins are not hidden. In the two scans, the share of WP websites with unknown core was relatively high, 34%. Also, the tool can find vulnerable websites with known vulnerabilities stored in public repositories, such as DVE or other accessible databases, so new unknown vulnerabilities cannot be assessed. The tool does not try to discover incorrect server configurations, default usernames or passwords and some other vulnerabilities, as well, as the scanning method respects ethics. If services such as CloudFlare are used, which offer website owners content delivery network services, DDoS mitigation, and Internet security, the tool is not capable to specify the country to which the server belongs.

6.3 Limitation of the Vulnerability Study at Large

This study provided some positive answers to our research questions and confirmed 3 of our hypotheses and rejected 2. The applied crawler method which explored the web space by interconnecting websites has some limitations as it leaves some domains underrepresented. Another limitation of the study is related with the percentage (34%) of unknown types of WP versions of the core and the plug-ins, due to hidden information. One of the limitations of the study is also that it addressed the web space of a country, as servers with domain belonging to that country were considered, and not the websites with other domain suffixes, which may be geographically present in the country's territory.

Another limitation is that the tool respects robots.txt files and consequently omits websites that site owners forbid to be scanned. Therefore, the actual number of secure or insecure WP websites may differ from the search results.

As with other studies, there were limitations to our research due to the selected data and the studied sample. The sample was composed of websites that have allowed the collection of additional data with the top-level domain from one of the 30 studied countries. The scanning of websites running WPCMS with our developed tool lasted two months, in order to collect all possible websites. The size of the sample obtained includes a sufficient number of websites to allow an exploratory analysis that provided credible results.

Although the number of downloaded and applied plug-ins is more than 56,000, the results of the study that was based on the scoring of almost 2,500 known vulnerabilities so the tool vulnerability database cannot be generalized. There are however several studies (da Fonseca & Vieira, 2014) that obtained similar results about the impact of plug-ins on website insecurity with WPCMS platform, but on much restricted number of analysed websites.

6.4 Further Work

The presented work is comprehensive in terms of the addressed scanning tasks and challenges in development of methods for large-scale WP vulnerability assessment. It answers the questions posed in the introduction and opens new directions for further research. The main directions for further research can be summarized as follows.

First, although the tool and methods were carefully designed, we plan to further investigate the effect of the use of other scanning and detection methods to improve the performance of the proposed methods. A different approach can be designed to detect

hidden versions of WordPress and reduce the number of unrecognized versions of the WP core.

Second, the proposed methods to analyse other popular open-source platforms, such as Joomla, Drupal, Magento and many others, will be extended. In this way, the usability of the tool would be more practical and beneficial for a wide range of users of open-source solutions and thus increase the reach of more websites in an even larger scale. The existing methods can be easily adapted to support other open-source platforms and enable functionalities that are already developed to this extent. Once the methods are extended, they can be evaluated and benchmarked on a variety of platforms.

Last but not least, the impact of user upgrades on website security and the trend of new discovered vulnerabilities will be investigated. A predictive model could be developed based on several consecutive performed scans and analyses.

Appendix A

Signature Data

A.1 API Response for Plug-in

API call: curl https://api.wordpress.org/plug-ins/info/1.1/?action=query_plug-ins | json_pp

Response:


```
"plug-ins": [
   {
    "name": "Contact Form 7",
    "slug": "contact-form-7",
    "version": "5.3",
    "author": "Takayuki Miyoshi",
    "author_profile": "https://profiles.wordpress.org/takayukister",
    "requires": "5.4",
    "tested": "5.5.2",
    "requires_php": false,
    "compatibility": [],
    "rating": 82,
    "ratings": {
     "1": 302,
     "2": 45,
     "3": 54,
     "4": 129,
     "5": 1285
    },
    "num_ratings": 1815,
    "support_threads": 515,
    "support_threads_resolved": 125,
    "downloaded": 160098441,
    "last_updated": "2020-10-21 10:25am GMT",
    "added": "2007-08-02",
    "homepage": "https://contactform7.com/",
    "sections": {
     "description": "Very long description...",
     "screenshots": "https://ps.w.org/contact-form-7/assets/screenshot-
1.png?rev=1176454",
    },
    "short_description": "Just another contact form plug-in. Simple but flexible.",
    "download_link": "https://downloads.wordpress.org/plug-in/contact-form-7.5.3.zip",
    "screenshots": {
```

```
        "1": {
          "src": "https://ps.w.org/contact-form-7/assets/screenshot-1.png?rev=1176454",
          "caption": "screenshot-1.png"
        }
      },
      "tags": {
        "contact": "contact",
        "contact-form": "contact form",
        "email": "email",
        "feedback": "feedback",
        "form": "form"
      },
      "versions": {
        "1.1": "https://downloads.wordpress.org/plug-in/contact-form-7.1.1.zip",
        "1.10": "https://downloads.wordpress.org/plug-in/contact-form-7.1.10.zip",
        "1.10.0.1": "https://downloads.wordpress.org/plug-in/contact-form-7.1.10.0.1.zip",
        "1.10.1": "https://downloads.wordpress.org/plug-in/contact-form-7.1.10.1.zip",
        "1.2": "https://downloads.wordpress.org/plug-in/contact-form-7.1.2.zip",
      ...
        "5.1.8": "https://downloads.wordpress.org/plug-in/contact-form-7.5.1.8.zip",
        "5.1.9": "https://downloads.wordpress.org/plug-in/contact-form-7.5.1.9.zip",
        "5.2": "https://downloads.wordpress.org/plug-in/contact-form-7.5.2.zip",
        "5.2.1": "https://downloads.wordpress.org/plug-in/contact-form-7.5.2.1.zip",
        "5.2.2": "https://downloads.wordpress.org/plug-in/contact-form-7.5.2.2.zip",
        "5.3": "https://downloads.wordpress.org/plug-in/contact-form-7.5.3.zip",
        "trunk": "https://downloads.wordpress.org/plug-in/contact-form-7.zip"
      },
      "donate_link": "https://contactform7.com/donate/"
    },
```


A.2 API Response for AS Number

API call: curl https://vulnet.e5.ijs.si/api/AS2107

Response:


```
{
  "query": "AS2107",
  "status": "success",
  "name": "ARNES-NET",
  "description": "Academic and Research Network of Slovenia",
  "countryCode": "SI",
  "website": "http://www.arnes.si/",
  "abuseContacts": [
    "abuse@arnes.si"
  ],
  "trafficEstimation": "10-20Gbps",
  "ownerAddress": [
    "Tehnoloski park 18",
    "SI-1000",
    "Ljubljana",
    "SLOVENIA"
```

```
    ],
    "rirAllocation": {
      "rir_name": "RIPE",
      "country_code": "SI",
      "date_allocated": "1992-11-06 00:00:00",
      "allocation_status": "allocated"
    },
    "ianaAssignment": {
      "assignment_status": "assigned",
      "description": "Assigned by RIPE NCC",
      "whois_server": "whois.ripe.net",
      "date_assigned": null
    },
    "prefixes": {
      "ipv4_prefixes": [
        {
          "prefix": "88.200.0.0/17",
          "ip": "88.200.0.0",
          "cidr": 17,
          "roa_status": "None",
          "name": "SI-ARNES-20051026",
          "description": "ARNES",
          "country_code": "SI",
          "parent": {
            "prefix": "88.200.0.0/17",
            "ip": "88.200.0.0",
            "cidr": 17,
            "rir_name": "RIPE",
            "allocation_status": "unknown"
          }
        },
  etc.
  }
```

Appendix B

Statistical Analysis Data

B.1 Classified Words per Country

	Sector	Keywords
SI	SOCIETY	društvo, drustvo, society
	HEALTH	zobo, zdrav, medic, bolniš, bolnica, health
	INSTITUT	razisk, inštitut, institut, lab
	FINANCE	podjet, poslov, posel, finance, bank, business
	NEWS	novice, news
	EDUCATION	šola, sola, fakulteta, gimnazija, srednja, school, facult
DE	SOCIETY	gesellschaft, society
	HEALTH	dentist, gesund, medizin, krankenhaus, medic, health
	INSTITUT	forschung, institut, lab
	FINANCE	firm, geschaft, geschäft, finanz, finance, bank, business
	NEWS	nachrichten, news
	EDUCATION	schule, fakultat, fakultät, gymnasium, oberstufe, school, facult
NL	SOCIETY	samenleving, society
	HEALTH	gezond, gezondheidscentrum, ziekenhuis, tandarts, medisch, medic, health
	INSTITUT	onderzoek, instituut, institut, lab
	FINANCE	bedrijf, financien, financiën, finance, bank, business
	NEWS	nieuws, news
	EDUCATION	faculteit, middelbare, school, facult
FR	SOCIETY	societe, société, society
	HEALTH	dentist, medec, médec, santé, sante, hôpital, hopital, medic, health
	INSTITUT	recherch, institut, lab
	FINANCE	entreprise, affaires, banque, finance, bank, business
	NEWS	nouvelles, news
	EDUCATION	école, ecole, facult, lycee, lycée, school, facult
GB	SOCIETY	society
	HEALTH	hospital, dentist, tablet, pill, medic, health
	INSTITUT	research, institut, lab
	FINANCE	company, finance, bank, business
	NEWS	news
	EDUCATION	school, faculty
IT	SOCIETY	società, societa, society
	HEALTH	dentist, sanitario, medic, ospedal, health
	INSTITUT	ricerca, institut, lab
	FINANCE	compagnia, affari, finanza, banca, bancario, finance, bank, business
	NEWS	notizi, cronac, news
	EDUCATION	scuola, facolta, facoltà, liceo, school, facult

DK	SOCIETY	samfund, society
	HEALTH	tandlaege, tandlæge, sund, medic, hospital, health
	INSTITUT	forskning, institut, lab
	FINANCE	virksomheden, virksomhed, posel, finans, bank, business
	NEWS	nyhed, news
	EDUCATION	skole, fakultetet, gymnasium, school, facult
PL	SOCIETY	społeczeństwo, spoleczenstwo, society
	HEALTH	dentyst, zdrow, medyc, szpital, health
	INSTITUT	badania, instytut, institut, lab
	FINANCE	firm, bizne, finans, finance, bank, business
	NEWS	aktual, news
	EDUCATION	szkola, szkoła, wydział, liceum, school, facult
ES	SOCIETY	sociedad, society
	HEALTH	dentist, salud, médic, medic, hospital, health
	INSTITUT	investiga, institut, lab
	FINANCE	compania, compañia, negocio, finanz, finance, banco, bank, business
	NEWS	noticia, news
	EDUCATION	escuela, school, facult
SE	SOCIETY	samhället, samhallet, society
	HEALTH	tänd, tand, häls, hals, medic, sjukhus, health
	INSTITUT	forskaren, studie, institut, lab
	FINANCE	företag, foretag, finans, finance, bank, business
	NEWS	nyheter, news
	EDUCATION	skola, fakult, gymnasiet, school, facult
CH	SOCIETY	gesellschaft, società, societa, societe, société, society
	HEALTH	dentist, medec, médec, santé, sante, hôpital, hopital, sanitario, ospedal, dentist, gesund, medizin, krankenhaus, medic, health
	INSTITUT	recherch, ricerca, forschung, institut, lab
	FINANCE	entreprise, banque, compagnia, affari, finanza, banca, bancario, firm, geschaft, geschäft, finanz, finance, bank, business
	NEWS	nouvelles, notizi, cronac, nachrichten, news
	EDUCATION	école, ecole, lycee, lycée, scuola, facolta, facoltà, liceo, schule, fakultat, fakultät, gymnasium, oberstufe, school, facult
CZ	SOCIETY	společnost, society
	HEALTH	zubar, zubař, zdrav, medic, nemocnice, health
	INSTITUT	vyzkum, výzkum, institut, lab
	FINANCE	spolecnost, společnost, podnikani, podnikání, finance, bank, business
	NEWS	novinky, news
	EDUCATION	škola, skola, fakult, school, facult
IE	SOCIETY	socha, society
	HEALTH	fiacloir, fiaclóir, slaint, sláint, leigheas, medic, ospideal, ospidéal, health
	INSTITUT	taighde, institi, institut, lab
	FINANCE	cuideachta, gno, gnó, airgeada, finance, banc, bank, business
	NEWS	nuacht, news
	EDUCATION	scoil, daimhe, dáimhe, school, facult
FI	SOCIETY	yhteiskunta, society
	HEALTH	hammasl, terve, laakintamies, lääkintämies, medic, sairaala, health
	INSTITUT	tutkimu, instituutti, institut, lab
	FINANCE	yhtio, yhtiö, yritys, liiketoimint, rahoittaa, taloudelliset, pankki, finance, bank, business
	NEWS	uutiset, news
	EDUCATION	koulu, tiedekun, lukio, school, facult
AT	SOCIETY	gesellschaft, society
	HEALTH	dentist, gesund, medizin, krankenhaus, medic, health
	INSTITUT	forschung, institut, lab

	FINANCE	firm, geschaft, geschäft, finanz, finance, bank, business
	NEWS	nachrichten, news
	EDUCATION	schule, fakultat, fakultät, gymnasium, oberstufe, school, facult
RO	SOCIETY	societate, society
	HEALTH	dentist, doctor, sanatate, sănătate, medic, spital, health
	INSTITUT	cerceta, cercetă, institut, lab
	FINANCE	companie, afaceri, finante, finanțe, banc, bank, business
	NEWS	stiri, știri, news
	EDUCATION	scoli, școli, școal, școal, facult, liceu, school, facult
BE	SOCIETY	samenleving, societe, société, gesellschaft, society
	HEALTH	gezond, gezondheidscentrum, ziekenhuis, tandarts, medisch, dentist, medec, médec, santé, sante, hôpital, hopital, dentist, gesund, medizin, krankenhaus, medic, health
	INSTITUT	onderzoek, instituut, recherch, forschung, institut, lab
	FINANCE	bedrijf, financien, financiën, entreprise, affaires, banque, firm, geschaft, geschäft, finanz, finance, bank, business
	NEWS	nieuws, nouvelles, nachrichten, news
	EDUCATION	faculteit, middelbare, école, ecole, facult, lycee, lycée, schule, fakultat, fakultät, gymnasium, oberstufe, school, facult
HU	SOCIETY	társadal, society
	HEALTH	fogorvos, egeszseg, egészség, orvosi, medic, korhaz, kórház, health
	INSTITUT	kutat, intézet, intézet, institut, lab
	FINANCE	tarsasag, társaság, uzlet, üzlet, penz, pénz, finance, bank, business
	NEWS	hír, hir, news
	EDUCATION	iskola, kar, kozepiskol, középiskol, school, facult
BG	SOCIETY	общество, obshtestvo, society
	HEALTH	зъбо, zubo, zŭbo, lekar, лекар, здрав, zdrav, medic, болница, bolnitsa, health
	INSTITUT	изследване, izsledvane, институт, institut, lab
	FINANCE	компан, kompan, бизнес, biznes, финанси, finansi, банка, finance, bank, business
	NEWS	новини, novini, news
	EDUCATION	училище, uchilishte, факултет, fakultet, гимназ, gimnazi, school, facult
NO	SOCIETY	samfunn, society
	HEALTH	tannlege, helse, medis, medic, sykehus, health
	INSTITUT	forskni, institut, lab
	FINANCE	selskap, virksomhet, bedrift, finans, finance, bank, business
	NEWS	nybeg, novise, news
	EDUCATION	skole, fakultet, school, facult
SK	SOCIETY	spoločno, spolocno, society
	HEALTH	zubar, zubár, zdravie, lekar, lekár, liecite, liečite, medic, nemocnice, health
	INSTITUT	vyskum, výskum, ustav, ústav, institut, lab
	FINANCE	spolocnos, spoločnos, obchod, financ, bankov, bank, business
	NEWS	spravy, správy, news
	EDUCATION	škola, skola, skolsk, školsk, fakulta, school, facult
EE	SOCIETY	uhiskond, ühiskond, society
	HEALTH	ham, tervis, medit, ravim, medic, haigla, health
	INSTITUT	uurija, uurimis, instituut, institut, lab
	FINANCE	ettevot, ettevõt, ari, äri, rahandus, finance, pangand, bank, business
	NEWS	uudised, news
	EDUCATION	kool, teadusk, school, facult
PT	SOCIETY	sociedade, society

	HEALTH	dent, saude, saúde, remédio, remedio, medico, médico, medic, hospital, health
	INSTITUT	pesquisa, institu, lab
	FINANCE	empre, negocio, negócio, financiar, finance, banco, bank, business
	NEWS	noticias, notícias, news
	EDUCATION	escola, faculdade, colegio, colégio, school, facult
HR	SOCIETY	društvo, drustvo, society
	HEALTH	stomatolog, zubi, zdrav, medic, bolnic, health
	INSTITUT	studij, istraz, istraž, institut, lab
	FINANCE	podjat, poslov, posao, finans, finance, bank, business
	NEWS	vijest, news
	EDUCATION	skola, škola, fakultet, gimnaz, school, facult
LT	SOCIETY	visuom, society
	HEALTH	danti, donto, sveika, medic, vaista, ligonin, health
	INSTITUT	tyrimai, tyrin, institut, lab
	FINANCE	kompan, verslas, finans, finance, bank, business
	NEWS	naujienos, news
	EDUCATION	mokykla, fakultet, school, facult
LU	SOCIETY	societe, société, gesellschaft, society
	HEALTH	spidol, zann, zänn, dokter, dentist, medec, médec, santé, sante, hôpital, hopital, gesond, gesund, medizin, krankenhaus, medic, health
	INSTITUT	fuersch, recherch, forschung, institut, lab
	FINANCE	betriber, entreprise, affaires, banque, firm, geschaft, geschäft, finanz, finance, bank, business
	NEWS	neiegkeeten, nouvelles, nachrichten, news
	EDUCATION	schoul, fakult, école, ecole, facult, lycee, lycée, schule, fakultat, fakultät, gymnasium, oberstufe, school, facult
GR	SOCIETY	κοινωνία, koinonia, koinonía, society
	HEALTH	δόντι, donti, dónti, γιατρός, giatrós, giatros, medic, νοσοκομείο, nosokome, health
	INSTITUT	έρευν, erevna, érevna, instit, lab
	FINANCE	ερευνητ, etaire, επιχείρηση, epicheirisi, epicheírisi, chrimatod, χρηματοδ, finance, bank, business
	NEWS	νέα, nea, néa, news
	EDUCATION	Λύκειο, lykeio, lýkeio, schol, σχολ, school, facult
IS	SOCIETY	samfélag, samfelag, society
	HEALTH	tonn, tönn, tannlaek, tannlæk, heilb, læknir, laekn, medic, sjúkrahús, sjukrahus, health
	INSTITUT	ranns, stofnun, institut, lab
	FINANCE	fyrirtaeki, fyrirtæki, viðskipti, vioskipti, fjármál, fjarmal, finance, bank, business
	NEWS	fréttir, frettir, news
	EDUCATION	skóli, skoli, deild, school, facult
LV	SOCIETY	sabiedr, society
	HEALTH	zob, veselīgi, veseligi, arsts, zāles, zales, medic, slimnīca, slimnica, health
	INSTITUT	pētnieks, petnieks, izpēte, izpete, institūt, institut, lab
	FINANCE	uzņēmumiem, uznemumiem, bizness, finans, finance, bank, business
	NEWS	ziņas, zinas, news
	EDUCATION	skola, fakult, school, facult
CY	SOCIETY	toplum, κοινωνία, koinonia, koinonía, society
	HEALTH	dişler, disler, dişç, disc, doktor, tıbbi, hastane, δόντι, donti, dónti, γιατρός, giatrós, giatros, medic, νοσοκομείο, nosokome, health
	INSTITUT	enstit, έρευν, erevna, érevna, instit, lab
	FINANCE	iş, is, maliye, sirket, şirket, ερευνητ, etaire, επιχείρηση, epicheirisi, epicheírisi, chrimatod, χρηματοδ, finance, bank, business

	NEWS	haber, νέα, nea, néa, news
	EDUCATION	okul, fakült, fakult, lise, Λύκειο, lykeio, lýkeio, schol, σχολ, school, facult
MT	SOCIETY	soċjetà, socjeta, society
	HEALTH	sinna, dentis, saħħtu, sanntu, tabib, medik, sptar, medic, health
	INSTITUT	riċerka, ricerka, institut, lab
	FINANCE	kumpani, negozju, finanz, finance, bank, business
	NEWS	aħbarijiet, anbarijiet, news
	EDUCATION	skola, fakult, school, facult

B.2 Top Three Insecure Web Servers per Country

COUNTRY	SERVER	total	insecure	secure	unknown	total all
AT	**microsoft**	429.00	47.55	19.35	33.10	0.57
	litespeed	330.00	37.88	31.82	30.30	0.44
	apache	64157.00	36.17	30.71	33.13	84.83
BE	**apache**	72275.00	37.95	27.47	34.58	77.83
	litespeed	1014.00	35.90	21.20	42.90	1.09
	microsoft	1059.00	33.24	16.71	50.05	1.14
BG	**microsoft**	25.00	48.00	20.00	32.00	0.34
	apache	5164.00	45.18	25.56	29.26	71.04
	litespeed	171.00	43.86	22.81	33.33	2.35
CH	**microsoft**	2707.00	40.93	31.55	27.52	1.77
	litespeed	3050.00	38.69	34.26	27.05	2.00
	other	14063.00	33.71	37.47	28.82	9.22
CY	**litespeed**	67.00	47.76	17.91	34.33	7.75
	nginx	209.00	40.67	19.14	40.19	24.19
	apache	444.00	40.09	26.58	33.33	51.39
CZ	**litespeed**	137.00	51.09	17.52	31.39	0.16
	ats	4778.00	44.58	26.20	29.22	5.59
	apache	45523.00	41.70	26.89	31.41	53.26
DE	**microsoft**	2537.00	38.51	29.29	32.20	0.32
	openresty	1346.00	35.66	31.58	32.76	0.17
	flywheel	268.00	34.70	34.70	30.60	0.03
DK	**microsoft**	1248.00	39.02	22.44	38.54	1.40
	flywheel	182.00	32.42	54.40	13.19	0.20
	apache	60241.00	31.58	33.84	34.57	67.46
EE	**apache**	13540.00	41.45	31.68	26.87	91.10
	nginx	986.00	36.92	26.98	36.11	6.63
	litespeed	71.00	35.21	21.13	43.66	0.48
ES	**microsoft**	770.00	50.13	15.06	34.81	0.75
	flywheel	27.00	44.44	29.63	25.93	0.03
	apache	61840.00	42.08	26.78	31.14	59.91
FI	**microsoft**	295.00	39.66	21.36	38.98	0.51
	apache	38633.00	34.98	29.02	36.00	67.33
	litespeed	3823.00	26.47	38.43	35.10	6.66
FR	**microsoft**	1032.00	58.04	12.40	29.55	0.37
	apache	207737.00	39.50	32.10	28.40	73.89

	nginx	38546.00	37.59	23.75	38.66	13.71
GB	**apache**	324869.00	41.69	29.31	29.00	61.15
	microsoft	7598.00	35.54	18.75	45.71	1.43
	other	26882.00	35.33	25.40	39.27	5.06
GR	**microsoft**	544.00	54.41	19.12	26.47	1.29
	apache	18049.00	44.78	24.41	30.82	42.81
	nginx	15202.00	41.74	23.48	34.78	36.06
HR	**openresty**	29.00	62.07	13.79	24.14	0.20
	microsoft	253.00	51.78	21.74	26.48	1.74
	apache	11078.00	44.12	28.23	27.65	76.04
HU	**microsoft**	219.00	50.68	17.35	31.96	0.53
	apache	30905.00	40.37	26.27	33.36	74.31
	nginx	7267.00	40.15	20.97	38.87	17.47
IE	**other**	1889.00	55.43	9.42	35.15	7.40
	microsoft	248.00	52.42	14.11	33.47	0.97
	apache	15597.00	46.70	23.78	29.52	61.14
IS	**microsoft**	54.00	48.15	7.41	44.44	0.98
	apache	3317.00	41.00	26.29	32.71	60.35
	litespeed	744.00	39.11	31.85	29.03	13.54
IT	**microsoft**	5079.00	57.49	15.48	27.03	2.28
	apache	118007.00	46.68	29.72	23.61	53.03
	other	34074.00	35.20	32.77	32.03	15.31
LT	**microsoft**	42.00	52.38	11.90	35.71	0.39
	apache	7793.00	48.85	22.69	28.46	71.81
	nginx	2132.00	45.83	21.86	32.32	19.64
LU	**microsoft**	18.00	50.00	5.56	44.44	0.39
	nginx	1324.00	48.72	16.62	34.67	28.96
	apache	2904.00	36.98	28.82	34.19	63.52
LV	**nginx**	2450.00	42.86	21.76	35.39	34.58
	apache	3460.00	41.99	20.66	37.34	48.83
	other	622.00	40.68	29.58	29.74	8.78
NL	**apache**	327735.00	34.81	36.34	28.85	70.89
	microsoft	12065.00	33.47	29.76	36.77	2.61
	litespeed	8542.00	31.37	37.05	31.57	1.85
NO	**ats**	24.00	58.33	33.33	8.33	0.05
	apache	25118.00	35.61	28.42	35.97	51.11
	nginx	15214.00	32.03	26.32	41.65	30.96
PL	**microsoft**	426.00	51.64	21.36	27.00	0.20
	openresty	220.00	47.27	21.36	31.36	0.10
	other	32986.00	46.36	25.04	28.59	15.14
PT	**microsoft**	323.00	50.46	21.98	27.55	1.82
	litespeed	217.00	42.40	23.04	34.56	1.23
	apache	11647.00	39.88	27.27	32.85	65.77
RO	**microsoft**	272.00	52.21	16.18	31.62	0.43
	apache	29380.00	43.24	25.86	30.89	46.61

	litespeed	22586.00	41.17	28.39	30.44	35.83
SE	**microsoft**	4934.00	48.09	18.04	33.87	2.33
	flywheel	406.00	40.15	36.45	23.40	0.19
	litespeed	1504.00	38.63	31.91	29.45	0.71
SI	**microsoft**	113.00	57.52	15.04	27.43	0.75
	apache	12564.00	42.59	26.68	30.73	83.13
	openresty	15.00	40.00	26.67	33.33	0.10
SK	**ats**	194.00	50.00	23.71	26.29	0.64
	microsoft	156.00	44.87	21.79	33.33	0.52
	apache	8036.00	43.74	27.51	28.75	26.70

B.3 Net Indicators Sorted by Digital Skill

country	IU	IU_under45	IU_over45	FoIU	HH	DS
IS	99	100	98	99	99	85
LU	97	100	91	92	93	85
NL	95	97	93	94	98	79
NO	98	99	95	97	96	77
SE	93	95	91	91	92	77
FI	95	99	90	93	94	76
CH	97	99	93	91	93	76
GB	95	100	89	94	95	71
DK	98	99	96	95	94	71
DE	93	99	86	90	94	68
AT	88	97	75	85	89	67
BE	90	97	80	87	87	61
EE	90	99	78	87	90	60
CZ	87	98	74	84	86	60
SK	83	97	65	78	81	59
FR	89	96	81	85	89	57
ES	87	98	61	83	86	55
LT	81	97	63	78	78	55
SI	81	96	66	79	87	54
PT	75	96	56	71	79	50
CY	85	98	66	84	86	50
HU	84	95	63	75	83	50
LV	79	99	69	81	82	48
IE	73	98	70	80	89	48
PL	79	97	57	75	84	46
GR	73	95	52	70	82	46
IT	77	90	62	72	84	44
HR	76	96	58	73	82	41
BG	67	88	48	64	72	29
RO	77	92	59	68	81	29

References

Akrout, R., Alata, E., Kaaniche, M., & Nicomette, V. (2014). An automated black box approach for web vulnerability identification and attack scenario generation. *Journal of the Brazilian Computer Society*, (p. 4).

Alexa. (2019). *Alexa database*. Retrieved from Alexa Internet: https://www.alexa.com

Alsaleh, M., Alomar, N., Alshreef, M., Alarifi, A., & Al-Salman, A. (2017). Performance-based comparative assessment of open source web vulnerability scanners. *Security and Communication Networks*.

Amin, A., Eldessouki, A., Magdy, M. T., Abdeen, N., Hindy, H., & Hegazy, I. (2019). Androshield: Automated android applications vulnerability detection, a hybrid static and dynamic analysis approach. *Information*, 326.

Ammann, P., & Offutt, J. (2016). *Introduction to software testing.* Cambridge University Press.

Arnaert, M., Bertrand, Y., & Boudaoud, K. (2016). Modeling vulnerable Internet of Things on SHODAN and CENSYS: An ontology for cyber security. *Proceedings of the Tenth International Conference on Emerging Security Information, Systems and Technologies (SECUREWARE 2016)*, (pp. 24-28).

Beale, J., Meer, H., van der Walt, C., & Deraison, R. (2004). *Nessus Network Auditing: Jay Beale Open Source Security Series.* Elsevier.

Bird, J., & Kim, F. (2014). Survey on application security programs and practices. *A SANS Analyst Survey; SANS Institute: Bethesda, MD, USA*, 1-24.

Blažič, B. J. (2021). *Cybersecurity Skills in EU: New Educational Concept for Closing the Missing Workforce Gap.* Cybersecurity Threats with New Perspectives. IntechOpen. (in press).

Bodenheim, R. C. (2014). *Impact of the Shodan computer search engine on internet-facing industrial control system devices.* Ohio: AIR FORCE INSTITUTE OF TECHNOLOGY.

Bou-Harb, E., Debbabi, M., & Assi, C. (2013). A statistical approach for fingerprinting probing activities. *2013 International Conference on Availability, Reliability and Security* (pp. 21-30). IEEE.

Breusch, T. S., & Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica: Journal of the Econometric Society*, 1287-1294.

Campbell, A. D. (2017, January). *Wordpress.org*. Retrieved from WordPress 4.7.2 Security Release: https://wordpress.org/news/2017/01/wordpress-4-7-2-security-release/

Catakoglu, O., Balduzzi, M., & Balzarotti, D. (2017). Attacks landscape in the dark side of the web. *In Proceedings of the Symposium on Applied Computing* (pp. 1739-1746). Association for Computing Machinery.

Censys. (2019). *Censys*. Retrieved from Censys: https://censys.io

Cernica, I. C., Popescu, N., & Tiganoaia, B. (2019). Security evaluation of wordpress backup plugins. *Security evaluation of wordpress backup plugins* (pp. 312-316). IEEE.

Chen, C., Cui, B., Ma, J., Wu, R., Guo, J., & Liu, W. (2018). A systematic review of fuzzing techniques. *Computers & Security*, 118-137.

Chen, P., Desmet, L., Huygens, C., & Joosen, W. (2016). Longitudinal study of the use of client-side security mechanisms on the European Web. *Proceedings of the 25th International Conference Companion on World Wide Web*, (pp. 457-462).

Chiba, D., Tobe, K., Mori, T., & Goto, S. (2012). Detecting malicious websites by learning IP address features. *2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet* (pp. 29-39). IEEE.

Cimpanu, C. (2020). *Millions of WordPress sites are being probed and attacked with recent plugin bug*. Retrieved from ZDnet: https://www.zdnet.com/article/millions-of-wordpress-sites-are-being-probed-attacked-with-recent-plugin-bug/

Crawl, C. (2019). *Common Crawl*. Retrieved from https://commoncrawl.org

da Fonseca, J. M., & Vieira, M. A. (2014). A practical experience on the impact of plugins in web security. *2014 IEEE 33rd International Symposium on Reliable Distributed Systems* (pp. 21-30). IEEE.

Database, E. (2019). *Exploit-DB*. Retrieved from https://www.exploit-db.com

De Laat, P. B. (2005). Copyright or copyleft?: An analysis of property regimes for software development. *Research Policy*, 1511-1532.

Delta, T. H. (2018, Oct). Retrieved from The Hague Security Delta: https://www.thehaguesecuritydelta.com/news/newsitem/976

Desikan, S., & Ramesh, G. (2007). *Software Testing*. Pearson Education India.

Doupé, A., Cavedon, L., Kruegel, C., & Vigna, G. (2012). Enemy of the state: A state-aware black-box web vulnerability scanner. In *Presented as part of the 21st $\{$USENIX$\}$ Security Symposium ($\{$USENIX$\}$ Security 12)* (pp. 523--538).

Durbin, J., & Watson, G. S. (1971). Testing for serial correlation in least squares regression. *III. Biometrika*, 1-19.

Durumeric, Z., Adrian, D., Mirian, A., Bailey, M., & Halderman, A. J. (n.d.). A search engine backed by Internet-wide scanning. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, (pp. 542-553).

Durumeric, Z., Bailey, M., & Halderman, A. J. (2014). An internet-wide view of internet-wide scanning. *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, (pp. 65-78).

Durumeric, Z., Wustrow, E., & Halderman, J. A. (2013). ZMap: Fast Internet-wide scanning and its security applications. *22nd {USENIX} Security Symposium ({USENIX} Security 13)*, (pp. 605-620).

EU-Commision. (2016). *DESI 2016 Country Profiles*. Retrieved from EC Europa: https://ec.europa.eu/digital-single-market/en/news/desi-2016-country-profiles

Eurostat. (2019). *European statistics*. Retrieved from https://ec.europa.eu/eurostat

Eurostat. (2019). *Individuals who have basic or above basic overall digital skills*. Retrieved from European Statistics: https://ec.europa.eu/eurostat/web/products-datasets/product?code=tepsr_sp410

FIRST. (2019). *A Complete Guide to the Common Vulnerability Scoring System*. Retrieved from Forum of Incident Response and Security Teams: https://www.first.org/cvss/specification-document

FIRST. (2020). *Common Vulnerability Scoring System - User Guide*. Retrieved from Forum of Incident Response and Security Teams: https://www.first.org/cvss/user-guide

Ghaleb, T. A. (2015). Website fingerprinting as a cybercrime investigation model: Role and challenges. *First International Conference on Anti-Cybercrime (ICACC)* (pp. 1-5). IEEE.

Gloor, P. A. (2006). *Swarm creativity: Competitive advantage through collaborative innovation networks.* Oxford University Press.

Goethem, T. V., Chen, P., Nikiforkais, N., Desmet, L., & Joosen, W. (2014). Large-scale security analysis of the web: Challenges and findings. *International Conference on Trust and Trustworthy Computing* (pp. 110-126). Cham: Springer.

Graham, R. D., & Johnson, P. C. (2014). Finite state machine parsing for internet protocols: Faster than you think. *IEEE Security and Privacy Workshops* (pp. 185-190). IEEE.

Hammond, S., & Umphress, D. (2012). Test driven development: the state of the practice. *In Proceedings of the 50th Annual Southeast Regional Conference*, (pp. 158-163).

Harvey, A. C., & Collier, P. (1977). Testing for functional misspecification in regression analysis. *Journal of Econometrics*, 103-119.

Hassan, M., Sarker, K., Biswas, S., & Sharif, H. (2017). Detection of Wordpress Content Injection Vulnerability. *arXiv preprint arXiv:1711.02447*.

Hastie, J., & Chambers, J. M. (1992). *Statistical models in S.* Wadsworth & Brooks.

Heaton, J. (2006). *Programming spiders, bots, and aggregators in Java.* John Wiley & Sons.

Huang, C. H., Zhang, K. Z., Cheng, W. H., & Shieh, W. S. (2017). Web application security: Threats, countermeasures, and pitfalls. *Computer*, 81-85.

ITU. (2019). *Digital skills insights.* Retrieved from https://academy.itu.int/digital-skills-insights-2019

Jang-Jaccard, J., & Nepal, S. (2014). A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences*, 973-993.

Jarque, C. M., & Bera, A. K. (1980). Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics letters*, 255-259.

Jorgensen, P. C. (2013). *Software Testing, 4th Edition.* Auerbach Publications.

Jorgensen, P. C. (2018). *Software testing: a craftsman's approach.* CRC press.

Kaner, C. (2006). Quality assurance institute worldwide annual software testing conference. *Exploratory Testing.*

Khan, N., Yaqoob, I., Hashem, I. A., Inayat, Z., Mahmoud Ali, W. K., Alam, M., . . . Gani, A. (2014). Big data: survey, technologies, opportunities, and challenges. *The scientific world journal.*

Kim, H., Kim, T., & Jang, D. (2018). An intelligent improvement of Internet-wide scan engine for fast discovery of vulnerable IoT devices. *Symmetry*, 151.

Koskinen, T., Ihantola, P., & Karavirta, V. (2012). Quality of WordPress plug-ins: an overview of security and user ratings. *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, (pp. 834-837).

Krsul, I. V. (1998). *Software vulnerability analysis.* Purdue University.

Laitinen, P. (2018). Vulnerabilities in the wild: Detecting vulnerable Web applications at scale.

Lalet, P. (2020). Retrieved from IVRE: https://ivre.rocks/

Lewis, W. E. (2017). *Software testing and continuous quality improvement.* CRC press.

Li, F., Durumeric, Z., Czyz, J., Karami, M., Bailey, M., McCoy, D., & Paxson, V. (2016). You've got vulnerability: Exploring effective vulnerability notifications. *25th {USENIX} Security Symposium ({USENIX} Security 16)*, (pp. 1033-1050).

Lutz, M. (2013). *Learning python: Powerful object-oriented programming.* O'Reilly Media, Inc.

Mansfield, M. (2020, May). *Cyber Security Statistics: Numbers Small Businesses Need to Know.* Retrieved from Small Business Trends: https://smallbiztrends.com/2017/01/cyber-security-statistics-small-business.html

Maunder, M. (2017). *Wordfence.* Retrieved from Wordfence: https://www.wordfence.com/blog/2017/02/rapid-growth-in-rest-api-defacements/

Maunder, M. (2019, May). *WordFence.* Retrieved from Rapid Growth in Defacements, Who was Hit, Who is Attacking: https://www.wordfence.com/blog/2017/02/rapid-growth-in-rest-api-defacements/

McGraw, G. (2006). Software Security: Building Security In. *Datenschutz und Datensicherheit*, 662-665.

Moen, D. (2016, May). *WordFence.* Retrieved from How Attackers Gain Access to WordPress Sites: https://www.wordfence.com/blog/2016/03/attackers-gain-access-wordpress-sites/

Nappa, A., Rafique, Z. M., Caballero, J., & Gu, G. (2014). CyberProbe: Towards Internet-scale active detection of malicious servers. *Proceedings of the 2014 Network and Distributed System Security Symposium (NDSS 2014)*, (pp. 1-15).

Nessus. (2019). *Nessus.* Retrieved from Nessus: https://www.tenable.com/products/nessus

Nguyen, T. K., & Hwang, S. O. (2016). Large-Scale Detection of DOM-Based XSS Based on Publisher and Subscriber Model. *2016 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 975-980). IEEE.

NIST. (2019, May). *National Vulnerability Database.* Retrieved from https://nvd.nist.gov

Nmap. (2020). Retrieved from Nmap: https://nmap.org

Opensource. (2019). *Opensource.* Retrieved from Opensource: https://opensource.org/licenses

OWASP. (2013). *OWASP 2013.* Retrieved from OWASP Presentation 2013: https://wiki.owasp.org/images/1/17/OWASP_Top-10_2013--AppSec_EU_2013_-_Dave_Wichers.pdf

OWASP. (2019). *OWASP Top Ten.* Retrieved from OWASP Top Ten: https://owasp.org/www-project-top-ten/

OWASP. (2020). *OWASP.* Retrieved from Sensitive Data Exposure: https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure.html

OWASP ZAP. (2021). Retrieved from OWASP ZAP: https://owasp.org/www-project-zap/

Patel, S. K., Rathod, V. R., & Prajapati, J. B. (2013). Comparative analysis of web security in open source content management system. *International Conference on Intelligent Systems and Signal Processing (ISSP)* (pp. 344-349). IEEE.

Potter, B., & McGraw, G. (2004). Software security testing. *IEEE Security & Privacy*, 81-85.

Ransbotham, S. (2010). An Empirical Analysis of Exploitation Attempts Based on Vulnerabilities in Open Source Software. In Weis.

Rosso, S., Abrahamson, B., Adams, M., Cave, J., HouSandí, H., Hulse, D., & Maiorana, P. (2015). *WordPress Security White Paper.*

Ruohonen, J. (2019). A Demand-Side Viewpoint to Software Vulnerabilities in WordPress Plugins. *Proceedings of the Evaluation and Assessment on Software Engineering*, (pp. 222-228).

Sahatqija, K., Ajdari, J., Zenuni, X., Raufi, B., & Ismaili, F. (2018). Comparison between relational and NOSQL databases. *41st international convention on information and communication technology, electronics and microelectronics (MIPRO)* (pp. 0216-0221). IEEE.

Salem, O., Hossain, A., & Kamala, M. (2010). Awareness program and ai based tool to reduce risk of phishing attacks. *In 2010 10th IEEE International Conference on Computer and Information Technology* (pp. 1418-1423). IEEE.

Samtani, S., Yu, S., Zhu, H., Patton, M., & Chen, H. (2016). Identifying SCADA vulnerabilities using passive and active vulnerability assessment techniques. *Conference on Intelligence and Security Informatics (ISI)* (pp. 25-30). IEEE .

Schagen, N., Koning, K., Bos, H., & Giuffrida, C. (2018). Towards automated vulnerability scanning of network servers. *Proceedings of the 11th European Workshop on Systems Security* (pp. 1-6). ACM.

Schryen, G. (2011). Is open source security a myth? *Communications of the ACM*, 130-140.

Sfakianakis, A., Douligeris, C., Marinos, L., Lourenço, M., & Raghimi, O. (2019). *Enisa threat landscape report 2018 15 top cyberthreats and trends.* Enisa.

Shodan. (2019). *Shodan.* Retrieved from Computer Search Engine: https://www.shodan.io

Silaen, K. E., & Lim, C. (2016). A novel countermeasure to prevent XMLRPC WordPress attack. *In 2016 International Conference on Data and Software Engineering (ICoDSE)* (pp. 1-6). IEEE.

Son, S., & Shmatikov, V. (2013). The Postman Always Rings Twice: Attacking and Defending postMessage in HTML5 Websites. NDSS.

Soska, K., & Christin, N. (2014). Automatically detecting vulnerable websites before they turn malicious. *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, (pp. 625-640).

Soussi, W., Korczynski, M., Maroofi, S., & Duda, A. (2020). Feasibility of Large-Scale Vulnerability Notifications after GDPR. *2020 IEEE European Symposium on Security and Privacy Workshops*, 532-537.

Stock, B., Pellegrino, G., Li, F., Backes, M., & Rossow, C. (2018). Didn't you hear me?—Towards more successful Web vulnerability notifications. *NDSS Network and Distributed System Security Symposium.*

Stock, B., Pellegrino, G., Rossow, C., Johns, M., & Backes, M. (2016). Hey, you have a problem: On the feasibility of large-scale web vulnerability notification. *25th {USENIX} Security Symposium ({USENIX} Security 16)*, (pp. 1015-1032).

Stuttard, D., & Pinto, M. (2011). *The web application hacker's handbook: Finding and exploiting security flaws.* John Wiley & Sons.

Summerfield, M. (2013). *Python in practice: create better programs using concurrency, libraries, and patterns.* Addison-Wesley.

Symantec. (2019). Retrieved from Internet Security Threat Report: https://www.broadcom.com/support/security-center/publications/archive

Tajalizadehkhoob, S., Van Goethem, T., Korczyński, M., Noroozian, A., Böhme, R., Moore, T., . . . van Eeten, M. (2017). Herding vulnerable cats: a statistical approach to disentangle joint responsibility for web security in shared hosting. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 553-567). ACM.

Trunde, H., & Weippl, E. (2015). WordPress security: an analysis based on publicly available exploits. *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*, (pp. 1-7).

Tsotsis, A. (2011, March 3). *Wordpress.com Suffers Largest DDoS Attack In Its History.* Retrieved from Tech Crunch: https://techcrunch.com/2011/03/03/wordpress-com-suffers-major-ddos-attack/

Uygur, S. U. (2014). *Penetration testing with BackBox.* Packt Publishing Ltd.

Vasek, M., & Moore, T. (2014). Identifying risk factors for webserver compromise. *International Conference on Financial Cryptography and Data Security* (pp. 326-345). Berlin, Heidelberg: Springer.

Vasek, M., Wadleigh, J., & Moore, T. (2015). Hacking is not random: a case-control study of webserver-compromise risk. *IEEE Transactions on Dependable and Secure Computing* (pp. 206-219). IEEE.

Viva Differences. (2020). Retrieved from Viva Differences: https://vivadifferences.com/difference-between-server-side-scripting-and-client-side-scripting/

W3. (2019). *W3.* Retrieved from Status Code Definitions: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

W3C. (2020). *HTML & CSS.* Retrieved from W3C: https://www.w3.org/standards/webdesign/htmlcss.html

W3C. (2021). *Mobile Web Application Best Practices.* Retrieved from W3C: https://www.w3.org/TR/mwabp/

W3Tech. (2019). *W3Tech.* Retrieved from W3Tech: https://w3techs.com

Walden, J., Doyle, M., Lenhof, R., Murray, J., & Plunkett, A. (2010). Impact of plugins on the security of web applications. *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, (pp. 1-8).

Welling, L., & Thomson, L. (2003). *PHP and MySQL Web development.* Sams Publishing.

Wikipedia. (2019). *Wikipedia.* Retrieved from Content management system: https://en.wikipedia.org/wiki/Content_management_system

Wikipedia. (2019). *Wikipedia.* Retrieved from Web content management system: https://en.wikipedia.org/wiki/Web_content_management_system

WordPress. (2019). Retrieved from WordPress Stats: https://wordpress.com/stats/

WPScan. (2019, May). *WPScan WordPress Security Scanner.* Retrieved from WPScan WordPress Security Scanner: https://wpscan.com/wordpress-security-scanner

Zaccone, G. (2015). *Python parallel programming cookbook.* Packt Publishing Ltd.

Zalewski, M. (2020). *Skipfish.* Retrieved from https://code.google.com/archive/p/skipfish/

ZoomEye. (2020). Retrieved from https://www.zoomeye.org/

Bibliography

Publications Related to the Thesis

Journal Articles

Cigoj, P., & Blažič, B. J. (2019). An Intelligent and Automated WCMS Vulnerability-Discovery Tool: The Current State of the Web. *IEEE Access*, *7*, 175466-175473.

Conference Paper

Cigoj, P., Stepančič, Ž., & Blažič, B. J. (2020). A Large-Scale Security Analysis of Web Vulnerability: Findings, Challenges and Remedies. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2020. ICCSA 2020. Lecture Notes in Computer Science, vol 12253. Springer, Cham. https://doi.org/10.1007/978-3-030-58814-4_64.

Blažič, B. J., & Cigoj, P. (2021). Web vulnerability in 2021 : large scale inspection, findings, analysis and remedies. In: Borcoci, E. (Ed.), Thulasiraman, R. P. (Ed.) Internet 2021, 24–30, IARIA, Best paper award.

Other Publications

Cigoj, P., & Blažič, B. J. (2016). An Advanced Educational Tool for Digital Forensic Engineering. International Journal of Emerging Technologies in Learning, 11(3).

Cigoj, P., & Blažič, B. J. (2016). Innovative Solution for an eID Secure Management in a Multi-platform Cloud Environment. In *Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015* (pp. 529-537). Springer, Cham.

Cigoj, P., & Blažič, B. J. (2015). An Innovative Approach in Digital Forensic Education and Training. In *IFIP World Conference on Information Security Education* (pp. 101-110). Springer, Cham.

Cigoj, P., & Blažič, B. J. (2015). An authentication and authorization solution for a multiplatform cloud environment. *Information Security Journal: A Global Perspective*, *24*(4-6), 146-156.

Cigoj, P., & Klobučar, T. (2012). Cloud security and OpenStack. In *Proceedings of the 1st International Conference on Cloud-Assisted Services, (AS'12)* (pp. 20-106).

Biography

Primož Cigoj was born on 24 April 1984 in Ljubljana, Slovenia, where he finished the primary (OŠ Mirana Jarca) and secondary school (Vegova) and bachelor's study of computer science at the Faculty of Computer and Information Science, University of Ljubljana. Since 2011, he has been employed at the Jožef Stefan Institute in Ljubljana, Slovenia. He has completed his master's degree dissertation in information and communication technologies at the Jožef Stefan International Postgraduate School and is now holding the Young Researchers award from the National Research Agency.

He is also an Ethical Hacker expert with over 20 years of experience in carrying out security checks and system penetration tests. His main areas of interest are information security, digital forensics, cybercrime prevention, and cloud computing. He has been a member of the board of the Internet Society (ISOC) Slovenia since 2011. His previous research work was related to the European project Dynamic Forensics Evaluation and Training (DFET) funded by EU DG Home. The goal of this project was to develop a cloud-based cybercrime training environment that included real-life simulation and scenario analysis of committed crime attacks. Within this project, he has developed an advanced educational tool for digital forensic engineering named EDUFORS. This innovative tool was presented as a journal article with IF and at international conferences. Within the EU FP7 Courage project, he contributed to the development of a research and development agenda for fighting cybercrime and cyberterrorism.

Primož Cigoj participated also in the EU DG JUST project LIVE-FOR (Criminal Justice Access to Digital Evidence in the Cloud – LIVE-FORensics). The main goal of the project was to upgrade the competence of judicial authorities in the European investigation order implementation, the awareness about cross-border electronic evidence collection, and education actions in cloud forensics. His participation included analytical activities about target groups of legal practitioners, the organization of joint meetings, the contribution of material to educational and training workshops, and assisting with awareness-raising and dissemination activities. He was also involved in the activities of the SENTER project funded by DG Home (Strengthening European Network Centres of Excellence in Cybercrime) that was creating a cybercrime awareness landscape and building the single point of reference for EU national cybercrime centres of excellence (CoE). His main activities here included an analysis of existing educational resources and IT tools in the fight against cybercrime, digital forensics, and open-source intelligence.

In 2017, Primož Cigoj won the Cyber security challenge competition organized by Europol HSD and NATO by scoring 100% of the solved tasks.