Web Application Honeypot Published in the Wild

Octavian Grigorescu
Computer Science
University Politehnica of Bucharest
octavian.grigorescu@upb.ro

Cristian Săndescu
Computer Science
University Politehnica of Bucharest
cristian.sandescu@upb.ro

Alexandru Caba
Computer Science
University Politehnica of Bucharest
alexandru.caba@stud.acs.upb.ro

***Abstract*—The goal of this project is to create an intelligent system that will be installed under a cybernetic infrastructure, with the purpose of detecting cyber-attacks and learning about their methods of scanning and penetrating. For this project, we aim to examine data generated by the attacker's actions and highlight the value of information provided by them. Thus, we create honeypots that hold "Capture the Flag" type of games in the area of web applications. These games have multiple challenges of varying difficulties aiming to keep the attacker busy in order to learn his behavior. We created a web application with multiple vulnerabilities and published the honeypot in the exposed internet for about 2 months. During this period, we discovered a lot of noise produced by the bots and scrappers that tried to find out relevant information from the machine. Besides this, we found 2 attempts of human interaction that succeeded to solve a couple of vulnerabilities and gave up at one point leaving the challenges unfinished.**

Keywords—honeypot, web application, bots, capture the flag

I. Introduction

Web applications allow users to send, receive and retrieve data from a database on the Internet using browsers. Despite all the advantages that web applications bring, they can raise many security issues resulting from improper programming or wrong configurations. Serious weaknesses or vulnerabilities allow attackers access to databases, thus giving them the opportunity to collect confidential information. It is often the case that valuable data (for example: personal and financial details) becomes the target of attackers. Although the purpose of such attacks often is to cause reputational damage, an increasing number of attackers seek to access data in order to make a profit from their sale.

The main objective of this project is to create and implement a honeypot system that is able to detect as early as possible the chances of an attack security on a web application and keep the attacker busy in a controlled environment. In this sense, such a system must provide useful information on vulnerable components and bring to knowledge new vulnerabilities that were not known until then. To accomplish this, the honeypot module must not be identifiable by the attacker, but at the same time must present itself as a real application to arouse interest and occupy as much of the time used for the attack.

The objective of this project is solving the web application attacks problem. More specifically, we aim to provide a solution that can stop or diminish the risk of cyber attacks on web applications. To address this situation, a"honeypot" can redirect the attacker in question in a controlled environment where he or she can not affect real data or cause damage of any kind. These systems are used to provide protection to real systems and gather insights regarding the behavior of an attacker to make further improvements and recognize future similar attacks.

At the same time, building honeypot systems requires extensive knowledge and thorough testing, as they can introduce more problems. If they are not properly configured or controlled there is a risk that the entire network will be compromised. Therefore, the problem of "honeypots" as risk sources is also addressed by the current project.

Section II describes the state of the art regarding honeypots. Recent developments regarding honeypots implementation in IoT environments are placed under observation due to an increasing number of such devices using web applications to communicate with end users. Section III presents related solutions that address the challenges and opportunities for honeypots in security systems. Section IV discusses the design and architecture whereas the section V the implementation. Next section contains the evaluation and the two final sections present the conclusions and the future work of the current research project.

II. State of the Art

The massive use of computers today has led to one of the major current issues, namely information security. This field is in a continuous change due to vulnerabilities that inevitably arise, and existing solutions must constantly be modified and developed to solve problems as quickly as possible. This creates a need for research studies on new technologies and architectures in order to find ways and ideas of attack prevention and mitigation.

An area that is especially vulnerable to cyber risks given its recent emergence is the Internet of Things (IoT) [1], [2]. As such, the following lines start by introducing the state-of-the art landmarks regarding honeypots in cybersecurity systems. An emphasis is placed on state of the art research regarding honeypots implementation in IoT [3] infrastructures as most of them use web applications to communicate with the end-user.

A. Honeytoken, Honeypot, Honeynet

A honeypot is a computer used with the intention of mimicking a probable target of cyber threats in which vulnerabilities have been intentionally added. Its role is to detect attacks, redirect them and monitor the traffic generated by such accesses within the network. The semantic field of the term honeypot reveals two closely related terms which are further discussed: honeytoken and honeynets [1].

A honeytoken is an information resource of a system placed in a public environment to detect attacks when it is accessed and used. It is not represented by a physical machine, but rather by a type of digital entity with no real purpose than to draw the attackers' attention and monitor whether unauthorized access has taken place [2].

A honeynet is a network of one or more honeypots. The purpose it meets is to collect information about the type of

2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet) | 978-0-7381-1265-7/20/$31.00 ©2020 IEEE | DOI: 10.1109/ROEDUNET51892.2020.9324870

attacks, how to operate, but also to divert unauthorized access to the real network and its resources. There are two types of honeynet systems: autonomous honeynet systems and hybrid honeynet systems [3].

B. *Low and High Interaction Honeypot*

Honeypots differ in the way they are launched and their complexity. A metric to characterize their types is given by the way they interact. The principle can be reduced to being a low, medium or high level depending on the purpose it wants to be fulfilled.

A low-interaction system will give the attacker limited access to the system operations. In this case it is not intended to analyze the system in detail, because it represents more of a static environment. It must provide a minimum of services and protocols enough to trick the attacker into believing that he is connected to a real system and not to a snare machine [6].

As an advantage, low-interaction honeypots are easy to start and configure, but they do not allow access to a real terminal with administrator rights. Therefore, such a honeypot cannot be very effective as it is a simple simulation of a system and there is a chance not to attract attackers since it is not complex and the information extracted from it its use does not bring impressive results.

A system with a high interaction is at the opposite pole of misleading. Instead of simulating simple services or protocols, the attacker receives a real system to attack, making it much harder to spot the fact that he accessed a controlled environment. By using such a high-interaction honeypot, one can learn methods or tools used to attack and escalate privileges [4]. The feature of adapting to each incident makes it much harder to detect that the system itself is a clone and brings much improved results.

Medium interaction honeypots combine both advantages of low and high interaction honeypots as it brings less risk than creating an entire physical machine and a higher performance than a virtual machine. However, for complex threats such as "zero-day exploits", such systems are not preferable [4].

III. Related work

A honeypot system can be implemented in various ways. It can be used for prevention, detection or collection of information related to attacks, attackers and vulnerabilities. The value of a honeypot depends on the interaction it manifests with the exterior and architecture of the system. Existing solutions do not provide specific information at the architectural level, in order to prevent disclosure that can help attackers to bypass such a system or to identify it when dealing with such a situation.

A. *Symantec Honeypot*

In 2015, Symantec set up a honeypot to attract targeted attacks on Internet of Things devices. These are interconnected products such as routers, video players or cameras. The architectural patterns have not been made public so as not to reveal clues to the attackers that they are in a honeypot, the devices being implemented services to imitate the real ones. The report provided by they support a double increase in the number of attacks only in 2016 [5].

Most of the attacks came from countries such as China, the United States, Russia, Germany and Vietnam representing the first 5 places in the ranking, demonstrating the urgent need for the existence of security standards in this area to make them more less vulnerable to attacks [5].

B. *Honeyd*

Honeyd is a virtual honeypot that creates virtual network topologies computers, being used against spam generated in the network following the attacks. This manages to mislead programs that want to gather information from the network such as Nmap, or ICMP scanner [6].

It does not require many resources for commissioning, and all harmful traffic can be redirected to it to mislead the tools of reconnaissance attackers and evaluation of a network with the data of a virtual one.

The honeypot listens to and interacts with targeted traffic exactly like a real server. ICMP, TCP or UDP traffic is processed within application, so that the entire system behind it is not affected by the attack [7].

The results provided are fast, as all the traffic destined for the virtual machines belonging to the honeypot can be considered belonging to an attack, and the attacker is slowed down and possibly redirected to another honeypot from which it cannot reach the system providing real server protection.

IV. Design and Arhitecture

In the subsequent form of the architecture, the structure of the project was based on several machines and servers, namely: a load balancer, the server that contains the actual web application, and another server that represents the honeypot In order to build this honeypot system, we emulated the Web server based on Wordpress technology and created multiple vulnerabilities having different levels of difficulty. The Figure 1 below illustrates the architecture of the project.

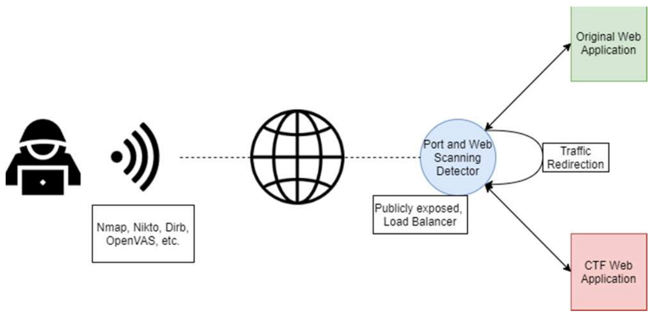

Original Web Application
Port and Web Scanning Detector
Traffic Redirection
Publicly exposed, Load Balancer
Nmap, Nikto, Dirb, OpenVAS, etc.
CTF Web Application

Figure 1 - Honeypot System Architecture

A. *Load Balancer*

The load balancer is placed on a single machine and it can only be seen from outside the network. Its role is to connect users with the web application. The main purpose is to redirect traffic to the actual server or to the honeypot, depending on the type of user. As soon as malicious traffic was detected (the use of programs for scanning, access test unauthorized or numerous malicious requests), the load balancer must interrupt access to the real Web application and direct the suspicious user towards the controlled environment. In this sense, the load balancer works like a router that sends traffic received from the external network to the server inside.

B. *Web Server*

The server with the real application is configured inside the internal network of the infrastructure and contains the resources that are delivered to the user. It is based on the

LAMP architecture (Linux as operating system, Apache as a web server for building a site, MySQL for database relational data containing site information and PHP as interpreted programming language used for Web application development) [8].

Apache v2.4.34 was used as an HTTP server due to its ease of use and flexibility. The service is "open-source" and used by approximately 44% of the sites present in the internet [9]. As one of the oldest and most reliable Web servers, Apache allowed "out of the box" operations with chosen WordPress site being one of the most popular CMS. The web has seen a high popularity of the WordPress application so it was chosen due to the major percentage of sites that use it on the Internet: (34%) and due to its popularity as a content management system: 60% [10].

MySQL v5.7 was used to build a WordPress compatible database to store application information. MySQL is a relational database, easy to use and integrate with web applications and was chosen because WordPress uses this database in its implementation. Therefore, the configuration of another technology could have led the attacker to suspicions.

Although it has multiple advantages, the complicated architecture of WordPress has introduced countless vulnerabilities over time and has turned web applications based on it in major target for attackers. These may include SQL attacks injection, XSS (cross-site scripting), reverse shell, cross-site scripting directories ("directory traversal") and the list can continue [11].

C. *CTF Honeypot Server*

The honeypot server must imitate as much as possible the real machine so that the attacker becomes interested in its content. Therefore, the design of the honeypot must ensure that the information provided to the attacker is as real and credible as possible. As such, the CTF Honeypot server must be a replica of the WordPress Server. However, vulnerabilities were purposefully introduced in the CTF Honeypot Server on four levels.

The implementation started with cloning the machine on which the real Web server was located, but no vulnerabilities introduced. Its purpose is to deal with access detected as unauthorized and catch the user in a "Capture the flag" game in order to obtain administration rights on the machine.

V. Implementation

The chosen architecture is based on a modular approach, each component being able to be developed and configured separately, so that in the end to form a complete module that fulfills the desired functionality.

In the initial phase, the architecture of the project consisted of two virtual machines consisting of the Web server running on a Linux-based operating system, Ubuntu distribution. On the second machine, Kali distribution was used as it offers a multitude of tools it offers in terms of penetration testing and security assessment.

This step has helped to gather information that can be withdrawn from scans on how the architecture should be improved to disguise the real machine.

In the subsequent form the structure of the project is based on several machines / servers and namely a load balancer, the server that contains the actual web application, another server that represents the honeypot and also one for the client who wants to use the application resources.

In computer networks, the DMZ ("demilitarized zone") is a logical or physical subnet that separates a local network from another one outside, usually the internet. Machines that communicate with the outside, services or resources are placed in this area to be accessible from the outside, with the rest of the local network remaining untouchable. This provides an additional layer of security to the internal components as it restricts attackers from directly accessing servers or data through the framework from the Internet [12].

The load balancer is placed on a single machine and it can be only seen from outside the network. The main purpose is to redirect traffic either to the actual server or to the trap machine, depending on the type of user. As soon unusual traffic to the application was detected, such as usage of scanning applications, unauthorized access tests or numerous malicious requests the load balancer must interrupt access to the Web application Web in order to protect it and place the attacker in a controlled environment where where no damage can be caused. Basically it works like a router that sends the traffic received from the external network to the server inside.

This way you can see which pages you are trying to access, and if the requests raise suspicions such as: the header of the request is altered, the requests are too many in a short time or are access non-existent pages repeatedly, new iptables rules can be introduced for the user concerned.

To detect scans on the system containing the web application, we used a program that recognizes TCP scans. Most tools for gathering information used in the attack recognition phase are based on certain scans carried out at the beginning, succeeding in making this program suitable for the detection of a general use.

A. *First Level*

The first vulnerability is to find a page that has been configured improperly. A cyberattack usually starts by using recognition tools such as Nmap, Dirb or Niktothe usage of this kind of scanning tools provides the attacker information about the system and ways to start exploiting it. Therefore, the attacker reaches the first level of the honeypot machine.

A scan tries to determine the OS, serivces that are running and their versions by trying to access the most common TCP or UDP ports. The purpose of the proxy is to detect requests from such programs and to serve back the client a link to the entry page within the CTF game, which has been intentionally left vulnerable to be able to retrieve information about how an attacker behaves in this kind of situation

The page in question does not have a direct link, which is why the initial scan is required in order to discover the web app. This page contains mainly logs without relevant information, and it aims to keep the attacker busy for a longer period of time.

The first level was configured using a WordPress plugin that allows viewing uploaded or downloaded files. The vulnerability occurs when a download attempt is made, because the service uses an older version which does not have sufficient security checks, and it permits directory traversal.

The download address discloses information that should not be included in the link, namely the absolute path to this file, which helps understanding how the web application is structured and allows downloading other files from the remote machine. By accessing the page described above, the attacker receives the configuration data of the WordPress application, including the username and password of the database used.

B. *Second Level*

For the second level, port 3306 for the MySQL service has been left open to allow remote access, so it can be exploited by the attacker in order to extract data from the Wordpress application. The attacker must decrypt a password stored in MD5 format, having the chance to receive high privileged access to the site. MD5 is a function hashing cryptography that receives an input of variable length and generates a string of fixed-length characters used to authenticate the original message. The algorithm ensures that for a string, there is a single "hash" that cannot be generated by another string.

The web application user has also been configured with administrator rights, and using an online database containing the usual strings whose "hash" it was calculated, the attacker can find out the password and can connect with the new credentials found.

C. *Third Level*

Once the WordPress application is accessed with the data obtained in the previous level, the attacker has more options to get to the machine on which it is located. This level provides the opportunity to collect information related to how it acts and what its intentions are, such as: editing web pages, changing settings, uploading files, adding users or plugins that provide access to a terminal or uploading files on the machine in question.

One way to get a terminal to the server is by editing an existing plugin in the application. In this case, the attacker can use a 'php-reverse-shell', which is an utility used in security testing for situations where it can be uploaded to a PHP running Web server [13]. The script will open a TCP connection from the web server to an IP and a port of choice. Attached to this connection, there will be an interactive terminal that will allow running of commands or programs such as telnet, ssh or su. The advantage is that it differs from a terminal based on a web form, which allows a single one to be sent orders. To make the TCP connection, a TCP listener on the machine which accesses the server using the port set in the script is required.

D. *Fourth Level*

In this phase, the attacker gained access to the physical machine representing a clone of the Web server. The purpose of the fourth level is to increase the user rights on the server in question. This can be exploited through various vulnerabilities including SUID (Set User ID) executables.

For this level we have created a vulnerability within an executable, that allows exceeding the memory on the stack ("buffer overflow") in order to execute arbitrary code entered in the payload. The executable was compiled with the "-fno-stack-protector" option to disable stack protection at program execution, but with ASLR ("Address space layout randomization") activated on the machine in question. ASLR (Address Space Randomization) is a security technique used to protect the system from "buffer overflow" attacks. ASLR has to randomly arrange the area of the address spaces for the programs executed, thus making it difficult to find the exact memory area where the execution takes place [14].

In this case, we can make use of Linux environment variables. The code that provides access to an administrator terminal can be placed in a variable preceded by many NOP ("no-operation") instructions, so that at the end of the program the return address used to jump to another position hits the launching hexa code sequence that opens a high privileged terminal.

The command represents the setting of the environment variable inside the system terminal operating system containing a series of NOP instructions and the hexa code representing the machine code interpreted by the processor that will launch a terminal with the rights it owns the executable holder. The NOP instructions are used to occupy a larger memory area within the stack, and at the processor level they will not influence execution flow, as the processor proceeds to the next instruction without modification of memory registers.

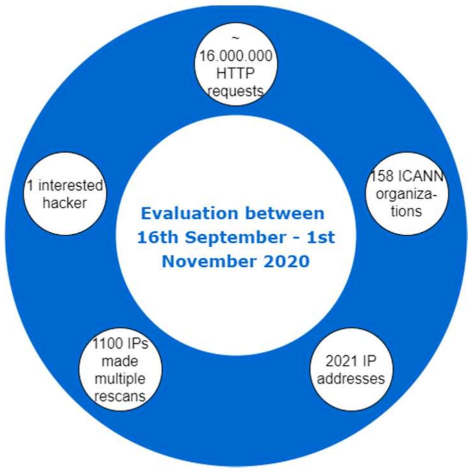
To increase the chances of successful operation, the execution of the program can be done repeatedly, so that at some point the random execution addresses of the program gets to execute the hexa code inserted in the environment variable. After performing the step from the attack vector described above, the attacker obtains administration rights and acomplished his purpose of gaining control over the machine. In order to preserve the machine and the data on it, subsequent access must be blocked, which can be done by stopping the machine or blocking the incoming traffic to the load balancer.

VI. Evaluation

In the first phase of the experiment, before publishing the honeypot, we tested the entire system using a couple of scanning tools such as Nmap, Dirb, Nikto and OpenVAS in order to evaluate the way it handles them.

Afterwards, we deployed and exposed the honeypot to the world starting mid September and ending early November in the Amazon Web Services cloud. We wanted to get the behavior for a short period of time. Based on collected data from the Honeypot system between 16th September 2020 and 1st November 2020, we discovered the followings (also presented in Figure 2.

- There were attempts made from around 158 ICANN organizations.
- There were around 2021 different IP addresses. Around 1100 of them had returned after the first contact and rescanned the honeypot in multiple days.
- There were around 16.000.000 HTTP requests especially generated by automated bots and scrapers that used multiple security tools such as Nmap, Nikto, Dirb, OpenVAS and others unknown.
- There were traces of at least one human interaction, detected in the beginning of the period of interest. The attack targeted the first two vulnerabilities, after which it had stopped.



~ 16.000.000 HTTP requests
158 ICANN organiza-tions
2021 IP addresses
1100 IPs made multiple rescans
1 interested hacker
Evaluation between 16th September - 1st November 2020

Figure 2 - Honepot in the wild results

We suspect the user ended his attack as soon as he realized he was not dealing with a real web application or nothing attractive was to found inside the application. After this event, there wasn't other human interaction detected.

Thus, we take into account the existence of a shared database between organizations of hackers such as a black list created and owned by them where they register diverse data including IP addresses that can have fake applications or services such as honeypots or honeynets. Also, we plan to leave the honeypot a longer time in the Internet to track another possbile attackers.

In order to have multiple visitors to our honeypot we could use a different approach, as mirroring an important or popular web service, which could be more attractive than a regular/trivial web application.

Having the honeypot deployed in the Amazon Web Services cloud, where lots of scrapers and bots scan the network and assets, there are big chances to be found by the attackers. The important part is to build an attractive honeypot context that lures the black hackers into playing the Capture The Flag rather than giving up quickly.

VII. Conclusions

This project presents a solution for honeypot system implementation within web application, based on identifying attackers, redirecting them to controlled environment and analyzing their behavior. The approach presented in this paper was successfully implemented and our objectives have been achieved, the module succeeding in identifying tools used for scanning or attacks targeted to the system or web application.

The approached area proved to be a problematic one and has noticed a great increase lately, the need to develop and research solutions being given by the large number of attacks and exploitations of vulnerabilities. There is a need for solutions to ensure web application protection systems, as many popular web applications still run with known and unsolved vulnerabilities.

A honeypot is generally challenging to implement, as we cannot find easily information on this topic and architectural ideas are kept secret in order to avoid attackers detecting and bypassing this kind of systems. Honeypot site comes as a complementary solution to what currently exists in terms of security, in addition to intrusion detection systems or firewalls.

The advantages of such a project consist in information and statistics collected from captured attacks, showing insights on tools and behavior that an attacker develops. Configuring the honeypot with new released versions of applications known to be secure, without having any associated CVE (Common Vulnerability Exposure), we increase the chances to detect a zero-day vulnerability that is not public, but exists and is exploited in the wild.

The honeypot shouldn't be easy to exploit by automated tools, but it also has to be easy to discover and we have to make sure that there is a human attacker in the background. Starting from an appropriate environment with a honeypot built on a Capture the Flag game composed of multiple levels of difficulty we can evaluate and determine the proficiency level of the attackers that try to break the honeypot.

VIII. Future Work

As future work we want to obtain multiple human interactions with the honeypot in order to collect more useful data for our research. Thus, we will create a honeypot based on a well know web application that already has lots of daily users and which is in the attackers target.

Following the structure change, we propose to implement a set of different improvements and evaluate the results. First of all, we want to create multiple exploring paths for the attackers containing Top 10 Web Application Security Risks [15] vulnerabilities and derivatives in order to discover novel methods of hacking. Secondly, we propose that certain aforementioned paths lead the attacker to the possibility of a better interaction with a shell. Thus, we can use special tools for monitoring his actions, and therefore to establish some information about his approaches and techniques that could be useful for our evaluation and for our next honeypot generation.

References

[1] C. R. a. R. S. A. Radovici, „A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures," *17th RoEduNet Conference: Networking in Education and Research RoEduNet,* pp. 1-5, 2018.

[2] L. C. R. a. R. I. Florea, „Challenges in security in Internet of Things," *16th RoEduNet Conference: Networking in Education and Research (RoEduNet), Targu Mures,* nr. 10.1109/ROEDUNET.2017.8123739, pp. 1-5, 2017.

[3] R. R. L. R. a. D. D. I. Florea, „Survey of Standardized Protocols for the Internet of Things," *21st International Conference on Control Systems and Computer Science,* pp. 190-196, doi: 10.1109/CSCS.2017.33, 2017.

[4] I. Livshitz, "What's the Difference Between a High Interaction Honeypot and a Low Interaction Honeypot?," 3 January 2019. [Online]. Available: https://www.guardicore.com/2019/1/high-interaction-honeypot-versus-low-interaction-honeypot/. [Accessed 1 11 2020].

[5] „What is a honeypot? How it can lure cyberattackers," 26 May 2020. [Interactiv]. Available: https://us.norton.com/internetsecurity-iot-what-is-a-honeypot.htm. [Accesat 1 11 2020].

[6] „Know Your Enemy: Defining Virtual Honeynets," September 2002. [Interactiv]. Available: http://ivanlef0u.fr/repo/madchat/reseau/defense/DefiningVirtualHoneynets.pdf. [Accesat 14 11 2020].

[7] „Low, Medium and High Interaction Honeypot," 3 January 2019. [Interactiv]. Available: https://www.guardicore.com/2019/1/high-interaction-honeypot-versus-low-interaction-honeypot/. [Accesat 14 11 2020].

[8] „What is a honeypot? How it can lure cyberattackers," 26 March 2020. [Interactiv]. Available: https://us.norton.com/internetsecurity-iot-what-is-a-honeypot.html. [Accesat 14 11 2020].

[9] „Developments of the Honeyd Virtual Honeypot," [Interactiv]. Available: http://www.honeyd.org/. [Accesat 14 11 2020].

[10] „Using HoneyD configurations to build honeypot systems," [Interactiv]. Available: https://searchsecurity.techtarget.com/Using-HoneyD-configurations-to-build-honeypot-systems. [Accesat 14 11 2020].

[11] „LAMP (Linux, Apache, MySQL, PHP)," [Interactiv]. Available: https://whatis.techtarget.com/definition/LAMP-Linux-Apache-MySQL-PHP. [Accesat 14 11 2020].

[12] „Usage statistics of Apache," [Interactiv]. Available: https://w3techs.com/technologies/details/ws-apache. [Accesat 14 11 2020].

[13] „WordPress Market Share," [Interactiv]. Available: https://kinsta.com/wordpress-market-share/. [Accesat 14 11 220].

[14] „WordPress: List of Security Vulnerabilities," [Interactiv]. Available: https://www.cvedetails.com/vulnerability-list/vendor_id-2337/product_id-4096/. [Accesat 14 11 2020].

[15] „Demilitarized Zone," [Interactiv]. Available: https://searchsecurity.techtarget.com/definition/DMZ. [Accesat 14 11 2020].

[16] „PHP Reverse Shell," [Interactiv]. Available: http://pentestmonkey.net/tools/web-shells/php-reverse-shell. [Accesat 1 11 2020].

[17] „What Is ASLR, and How Does It Keep Your Computer Secure?," 2016. [Interactiv]. Available: https://www.howtogeek.com/278056/what-is-aslr-and-how-does-it-keep-your-computer-secure/. [Accesat 14 11 2020].

[18] „OWASP Top Ten," 2020. [Interactiv]. Available: https://owasp.org/www-project-top-ten/. [Accesat 14 11 2020].