Cybersecurity of WordPress Platforms. An Analysis Using Attack-Defense Trees Method

Gabriel PETRICĂ, PhD
Faculty of Electronics, Telecommunications and Information Technology,
University POLITEHNICA of Bucharest, Romania
gabriel.petrica@upb.ro

Abstract

The aim of this paper is to analyze the techniques for securing a Content Management System, highlighting the vulnerabilities of the WordPress platform. The study includes qualitative and quantitative analyzes on the resilience of CMS platforms to cyber-attacks, simulated by the AD Trees methodology. The data provided by CVE is used to build possible attack scenarios that could compromise the cybersecurity of the web application. At the end of the paper, in order to minimize the impact of these attacks, solutions are proposed as sets of countermeasures within the Attack-Defense Trees.

Index terms: Attack-Defense Tree, CMS, cyber-attacks, software vulnerabilities, WordPress

1. Introduction

In today's information society, a Content Management System (CMS) is a modern and easy-to-use tool that accelerates an organization's ability to create, disseminate, and update various types of content. Current web technologies enable organizations to deliver consistently updated, real-time content and meet clients' dynamic demands. CMS is the solution for automated control of information collection, management, and distribution [1].

Among the features of a Content Management System, we can mention:

- easy to install and adapt to various types of websites;
- the graphical interface is simple and facilitates content management (text, images, multimedia objects);
- is suitable for different types of content;
- benefit from regular updates.

The functionalities of a CMS consist mainly of 3 aspects: *data collection*, *components management* and *content publishing*. *The collection* is the process of creating or acquiring data, automatically converting the format, the aggregation process, reusing information, and content segmenting into metadata. *Components management* can refer to several aspects regarding [2]:

- the system (components security, automatic indexing, management dashboard, scalability, flexibility and simplicity);
- document management (keeping document versions, several types of formats, automatic updating and deleting, compatibility with other applications);
- publishing (using a template for a consistent layout, automatic content conversion, advanced customization system, multilingual interface).

In the content lifecycle management process (fig. 1) the following steps are considered [3]:

- content organization and planning;

- defining the workflow;
- content creation;
- data storage and availability in case of emergency;
- professional content editing;
- checking the content before its publication;
- deleting / archiving the content or updating it to bring it back to the attention of the audience.

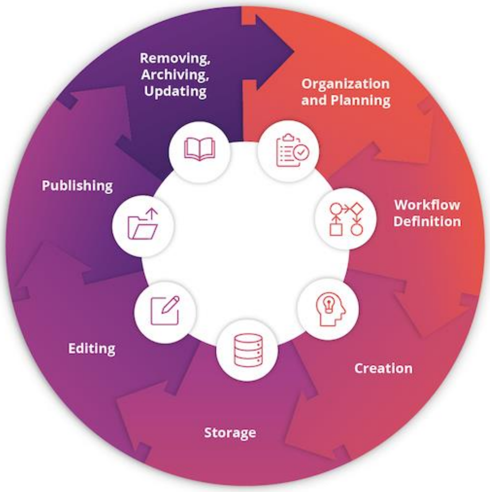


Fig. 1. Content lifecycle management steps [3]

2. WordPress vulnerabilities

Content management systems are becoming more and more popular and, thanks to the features offered, the number of websites using CMS is growing exponentially. That's why websites developed with a content management system are becoming attractive targets for cybercriminals.

The security of the infrastructure that connects to the Internet environment is essential for organizations when considering the security of their network. Even if the infrastructure that connects to the outside world does not store sensitive information, there is a significant risk to the reputation of organizations in the event of cyber-attacks. Security vulnerabilities in content management systems installed on organizations' web servers are often exploited by cyber attackers. Once such a system has been compromised, the web server can be used as a tool to facilitate intrusion attempts or to launch other cyber-attacks [4].

An attacker can use automated tools to find security vulnerabilities in web servers or CMS platforms. Usually, these intrusions are the result of a low level of security of the victim's computer or server rather than deliberate attacks. If a software vulnerability is found or a CMS has been compromised, the attacker can:

- gain access to the authenticated and privileged areas of a web application;
- upload malware to the web server to facilitate access through Remote Administration Tool (RAT);
- inject malicious content into legitimate web pages.

The different ways of identifying and noting software vulnerabilities and the lack of interoperability between databases and tools related to the same vulnerabilities led to the launch of the Common Vulnerabilities and Exposures (CVE) project in 1999. Designed and coordinated by Mitre Corporation, CVE is a tool for monitoring and standardizing the most common vulnerabilities, which ensures trust between the parties when is used to discuss or share information about a unique vulnerability of an application, operating system, service or firmware [5]. Each known vulnerability is uniquely identified in the National Vulnerability Database (NVD) and is also available in the CVE list, with detailed descriptions for each vulnerability, as well as a Common Vulnerability Scoring System (CVSS) that quantifies (on a scale from 0 to 10 - maximum severity) the impact of that vulnerability [6].

For a WordPress platform, the distribution of software vulnerabilities in 2021, based on data collected by CVE Details [7], is described numerically in the next table and graphically represented in fig. 2.

Table 1. WordPress software vulnerabilities reported in 2021 [7]

#	CVE ID	Vulnerability Type(s)	Score	Affects Confidentiality?	Affects Integrity?	Affects Availability?
1	CVE-2021-44223	Exec Code	7.5	Partial	Partial	Partial
2	CVE-2021-39203	Bypass	6.0	Partial	Partial	Partial
3	CVE-2021-39202	XSS	3.5	None	Partial	None
4	CVE-2021-39201	XSS, Bypass	3.5	None	Partial	None
5	CVE-2021-39200	+Info	4.3	Partial	None	None
6	CVE-2021-29450	+Info	4.0	Partial	None	None
7	CVE-2021-29447	undefined	4.0	Partial	None	None
8	CVE-2020-36326	undefined	7.5	Partial	Partial	Partial

We note that these software vulnerabilities generally have a medium impact on the security of the WordPress platform (with a score between 3.5 and 7.5 points). However, there are three exceptions, which partially affect the components of the C-I-A (Confidentiality, Integrity, and Availability) triad:

- CVE-2021-44223, “Execute Code” type vulnerability: “WordPress before 5.8 lacks support for the Update URI plugin header. This makes it easier for remote attackers to execute arbitrary code via a supply-chain attack against WordPress installations that use any plugin for which the slug satisfies the naming constraints of the WordPress.org Plugin Directory but is not yet present in that directory.” [8];
- CVE-2020-36326: “PHPMailer 6.1.8 through 6.4.0 allows object injection through Phar Deserialization via addAttachment with a UNC pathname” [9];
- CVE-2021-39203, “Bypass” type vulnerability: “Authenticated users who don't have permission to view private post types/data can bypass restrictions in the block editor under certain conditions” [10].

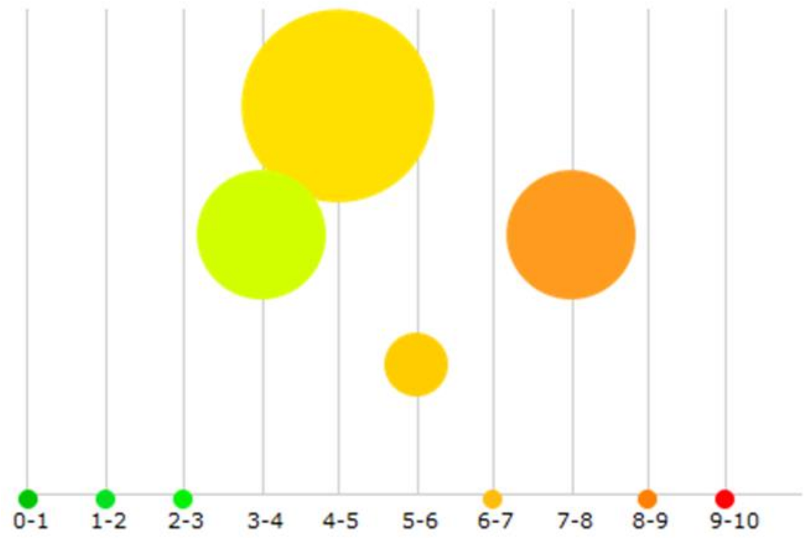


Fig. 2. WordPress software vulnerabilities distribution in 2021, by CVSS score (according to CVE Details [7])

3. Exploiting WordPress vulnerabilities. Attack-Defense Trees

Attack-Defense Trees (AD Trees) is a graphical methodology used by both system designers and personnel specialized in ensuring information security in various fields (information technology, military or aerospace industry, banking or business, etc.) to model and analyze system security threats. This method, introduced by Bruce Schneier, provides a graphical analysis of the ways in which a target (computer, network, organization) can be attacked (a goal can be reached by a potential threat) and indicates the measures taken by the defender to prevent achieving that goal by the attacker [11].

An AD Tree can have two types (categories) of nodes: attack nodes (represented by circles) and defense nodes represented by rectangles, while the *root node* of the tree describes the main target (objective) of one of the players (attacker or defender, depending on the role analyzed). The *root* node represents a state of success (from the point of view of the attacker), while the nodes of the tree express different ways to achieve the main goal [12]. An Attack Tree is a subclass of AD Trees in which only the actions of the attacker are considered.

In this hierarchical structure, each node can have one or more children of the same type, which detail the purpose of the node into several sub-objectives. Thus, each path represents a unique type of attack (a possible scenario). The relations between nodes can be conjunctive or disjunctive, being defined as follows:

- the objective of a conjunctive node is reached when all its objectives directly connected from the lower level are reached (AND logical function) - AND (N_1, N_2) - fig. 3.a;

- the objective of a disjunctive node is reached when at least one of its objectives directly connected from the lower level are reached (OR logical function) - fig. 3.b.

root
N_1
N_2
(a)
root
N_1
N_2
(b)

Fig. 3. The *root* and *N_1*, *N_2* nodes in an AD Tree: (a) AND function; (b) OR function

In this paper we used the ADTool application [13] because it allows the construction of the attack / defense scenarios in a descriptive way, based on a language consisting of six operators that can have as arguments labels or other operators:

- op() - *or-proponent* (OR operator for the attacker), $op(x, y) = x \cap y$;
- oo() - *or-opponent* (OR operator for the defender);
- ap() - *and-proponent* (AND operator for the attacker), $ap(x, y) = x \cup y$;
- ao() - *and-opponent* (AND operator for the defender);
- cp() - *countermeasure-proponent* (operator for the attacker), $cp(x, y) = x \cap y'$;
- co() - *countermeasure-opponent* (operator for the defender).

We conducted a case study based on the security analysis of a web application built on a WordPress platform. After installing the necessary tools and the actual implementation of the Web application, we built the Attack Tree based on the ADT methodology with the main objective "WordPress security breach" by exploiting software vulnerabilities of the WordPress platform identified in 2021. Delaying response or blocking operation of application, unauthorized data extraction, monitoring activity or inserting malicious code are other objectives of a potential attacker.

Possible attacks will exploit the eight software vulnerabilities reported in 2021 for WordPress platform [14], indicated in tab. 2 and represented graphically in fig. 4.

Table 2. WordPress vulnerabilities reported in 2021 [14]

WordPress	**TOTAL**	**Execute Code**	**Bypass**	**XSS**	**Gain Information**	**Other types**
	8	1	1	2	2	2

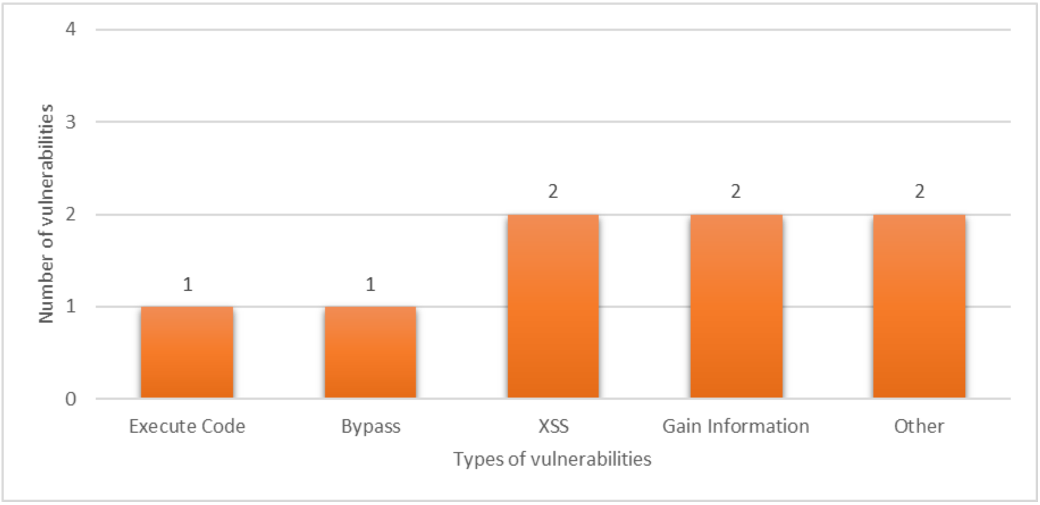


Fig. 4. Vulnerabilities in WordPress platform (according to CVE [14])

The code of the Attack Tree (represented in fig. 5) is indicated below: the achievement of the proposed objective (“*WordPress security breach by exploiting software vulnerabilities*”) will be possible by exploiting any of the eight vulnerabilities identified (CVE-2021-44223 *or* CVE-2021-39203 *or… or* CVE-2020-36326).

```
op(CVE-2021-44223, CVE-2021-39203, CVE-2021-39202, CVE-2021-39201, CVE-2021-39200, CVE-2021-29450,
CVE-2021-29447, CVE-2020-36326)
```

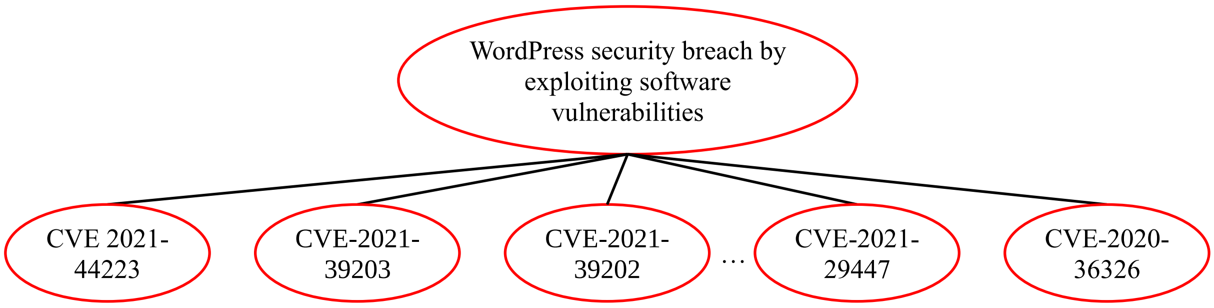


Fig. 5. Attack Tree - the objective “*WordPress security breach by exploiting software vulnerabilities*”

The countermeasures adopted by the defender refer to the following categories of corrective actions (see tab. 3 and fig. 6 representing the AD Tree with the implemented countermeasures):

- fixing CVE vulnerabilities;
- secure management of passwords;
- validate the data entered for authentication and limit the number of login attempts;
- other measures (the countermeasures N_4).

Table 3. Countermeasures / corrective actions

Countermeasures	**Corrective actions**
N_1	Update WordPress components: platform (core) / plugins / themes Delete unused plugins / themes

Countermeasures	Corrective actions
	Installation from secure sources Rules in .htaccess (configuration file used by Apache web server)
N_2	Requesting a minimum length password Weak passwords warnings Mechanism for generating random passwords Controls for complex passwords Not allowing passwords with a typical content Password strength indicator Encrypt stored passwords Sending passwords only through secure protocols Provide a reset procedure if the password has been forgotten Require a periodic password change
N_3	Limit login attempts for a user / IP address Using CAPTCHA technology Implementing 2-factor authentication Full input validation Using Web Application Firewall Creating an unpredictable account name and granting administrator rights, followed by deleting the default administrator account ("admin") Disconnect idle accounts Show minimal information if login failed (e.g. "Incorrect login")
N_4	Limiting responses to queries Disable XML-RPC protocol Changing the default tables prefix, at WordPress setup or later (manually or by using a plugin) Request re-authentication before changing sensitive settings

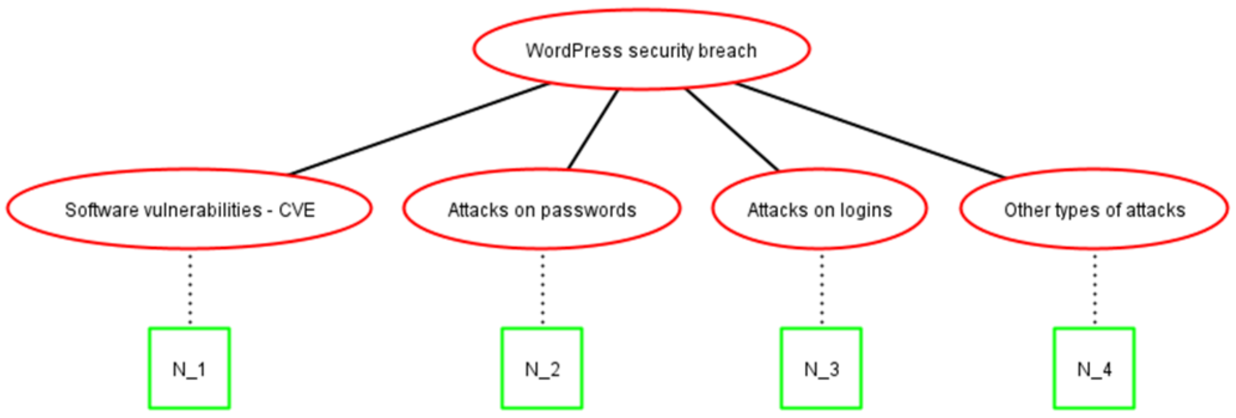


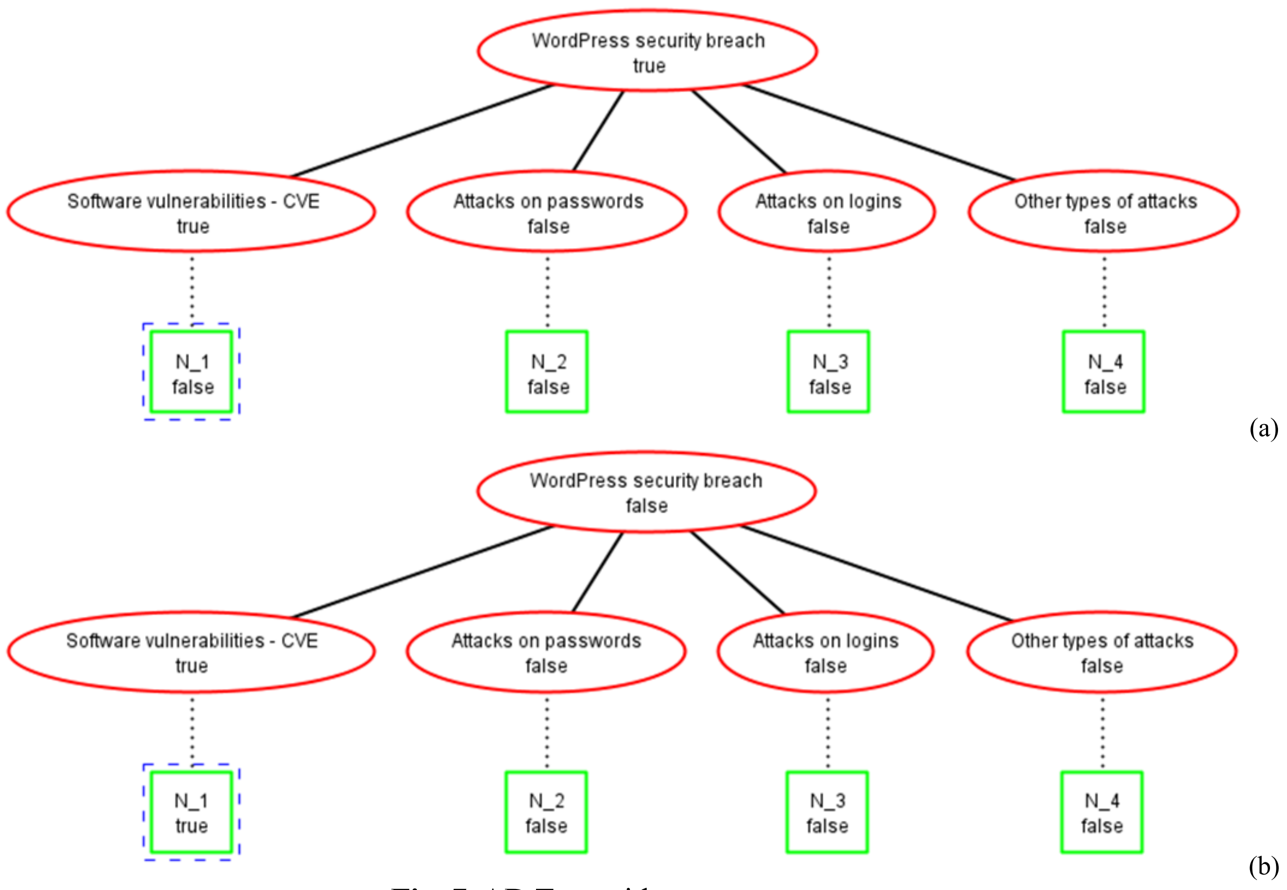
Fig. 6. AD Tree with the implemented countermeasures

The code of this AD Tree will be based on the operators op() and cp() and will have the following expression:

```
op(
 cp(Software vulnerabilities - CVE, N_1),
 cp(Attacks on passwords, N_2),
 cp(Attacks on logins, N_3),
 cp(Other types of attacks, N_4)
)
```

The AD Tree with the implemented countermeasures will have the configurations presented in fig. 7: the logical values *true* / *false* were inserted for the labels corresponding to the attack types, respectively to the countermeasures. The true / false quantitative analysis for the "Software vulnerabilities - CVE" label and N_1 countermeasures allow the following conclusions:

- if the label "Software vulnerabilities - CVE" is *true* and N_1 set of countermeasures is *false* (not implemented), the security is compromised (the top objective "WordPress security breach" is *true*) - see fig. 7.a;
- if the label "Software vulnerabilities - CVE" is *true* and N_1 set of countermeasures is *true*, the attacker does not achieve his goal ("WordPress security breach" is *false*) - see fig. 7.b.



WordPress security breach true
Software vulnerabilities - CVE true
Attacks on passwords false
Attacks on logins false
Other types of attacks false
N_1 false
N_2 false
N_3 false
N_4 false
(a)
WordPress security breach false
Software vulnerabilities - CVE true
Attacks on passwords false
Attacks on logins false
Other types of attacks false
N_1 true
N_2 false
N_3 false
N_4 false
(b)

Fig. 7. AD Tree with countermeasures:
(a) N_1 set of countermeasures is disabled (false); (b) N_1 set of countermeasures is activated (true)

4. Conclusions. Further developments

Apart from the four types of software vulnerabilities presented in tab. 1, over time there have been other types of WordPress vulnerabilities that have allowed various types of cyber-attacks on web applications: SQL injection, Denial of Service, Directory Traversal, Cross Site Request Forgery (CSRF) or Gain Privilege (see CVE database). Among the solutions for protection against these types of attacks we can consider [15]:

- use of input validation and isolation techniques against injection attacks;
- implementation of authorization and authentication mechanisms to prevent security breaches;
- creating backup copies of data;
- use of secure connections.

Security controls that assess vulnerabilities during the installation of a CMS can include:

- use of specific tools to scan for security vulnerabilities, such as WPScan for WordPress;
- performing vulnerability assessments of the custom code or plugins used to enhance some functionalities of CMS.

Further developments of this study consist in extending the analysis to other software components of a complex web application. Vulnerabilities can be identified in the operating system, web server, PHP version, databases, or other software components. Other situations that may lead to application malfunctions include physical attacks (steal or destroy of equipment and access to the server room).

This paper is a starting point in studying the security of a web application built on a WordPress platform. Defending against cybersecurity threats is an ongoing process that must involve awareness of the latest techniques and tools, correlated with the hardware and software environment in which the application operates, but also with user training, the most vulnerable component in ensuring security in an organization.

References

[1]. Adobe, Glossary term ”Content management”, https://business.adobe.com/sg/glossary/content-management.html.

[2]. C. Benevolo, Evaluation of Content Management Systems (CMS): a Supply Analysis, 2017.

[3]. Contentstack, Content Lifecycle Management for the Modern Enterprise, https://www.contentstack.com/blog/all-about-headless/content-lifecycle-management/.

[4]. Securing Content Management Systems, 2020, [Online] https://www.cyber.gov.au/sites/default/files/2020-06/PROTECT%20-%20Securing%20Content%20Management%20Systems%20%28June%202020%29.pdf.

[5]. CVE - Common Vulnerabilities and Exposures, https://cve.mitre.org/.

[6]. NIST Common Vulnerability Scoring System Calculator Version 3, https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator.

[7]. CVE Details - CVSS Scores for WordPress, https://www.cvedetails.com/cvss-score-charts.php?product_id=4096.

[8]. Vulnerability Details: CVE-2021-44223, https://www.cvedetails.com/cve/CVE-2021-44223/.

[9]. Vulnerability Details: CVE-2020-36326, https://www.cvedetails.com/cve/CVE-2020-36326/.

[10]. Vulnerability Details: CVE-2021-39203, https://www.cvedetails.com/cve/CVE-2021-39203/.

[11]. B. Schneier, Attack Trees, Dobb's Journal, 1999, https://www.schneier.com/academic/archives/1999/12/attack_trees.html.

[12]. G. Petrică, S.D. Axinte, I.C. Bacivarov, Dependabilitatea sistemelor informatice, Matrix Rom, București, 2019, ISBN 978-606-25-0529-5.

[13]. B. Kordy, P. Kordy, S. Mauw, P. Schweitzer, ADTool: Security Analysis with Attack-Defense Trees, Proceedings of the 10th International Conference on Quantitative Evaluation of Systems, 2013, DOI: 10.1007/978-3-642-40196-1_15.

[14]. CVE Details, WordPress: Security Vulnerabilities Published In 2021, https://www.cvedetails.com/vulnerability-list/vendor_id-2337/product_id-4096/year-2021/WordPress-WordPress.html.

[15]. G. Petrică a.o., Cybersecurity Guide, 2021, ISBN 978-973-0-33645-0, DOI: 10.19107/CYBERSEC.2021.EN, https://www.cyberlearning.ro/cybersecurity-guide/.