International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-5, June 2019

Vulnerability Modelling to Improve performance of web application vulnerability scanners

Siham El Idrissi, Ichrak Lafram, Naoual Berbiche, Fatima Guerouate, Mohamed Sbihi

***Abstract*: *Although web vulnerability scanners are valuable components when auditing the security of an application or website, they largely lack the ability to identify important vulnerability classes in advance. Therefore, a scanner is needed to help cover a wide range of vulnerability types. A new modelling of Web vulnerabilities is proposed in this article to highlight the input vectors that can convey them in Web applications. The modelling will then be used in the dataset creation phase based on the input vectors that will subsequently be modelled and detailed in this article. The modelling will be considered as the input of the learning system, in order to apply machine-learning techniques later, to optimize the scanners and this by improving the vulnerability detection performances by these tools*.**

Index Terms*: *Web Application, Attacks, Vulnerabilities, Web Vulnerability Scanners, Vulnerability Modelling.

I. INTRODUCTION

The security audit of an application or website requires careful thinking and planning to find the right tool to detect all types of vulnerabilities. Web Application Vulnerability Scanners (WAVS) are defined as scanning tools that examine web applications for potential security vulnerabilities such as Cross-Site Scripting (XSS), SQL Injection, directory traversal, insecure configurations, and remote command execution vulnerabilities. (WAVS) help developers and penetration testers to identify existing vulnerabilities that could compromise the security and privacy of data exchanged between the client and web server during the development and deployment phases.Despite the importance of these tools, they have limitations. These limitations were mentioned in our article [1], tackling the evaluation of web vulnerability free and commercial scanners to understand their functioning and architecture and compare their vulnerability detection performance in web applications based on accuracy, recall and F-measurement. The assessment helped study the architecture and operation of the scanners, specify the degree of effectiveness, draw conclusions about their ability to detect vulnerabilities and prevent others by proposing recommendations on the use of WAVS by companies or organizations. The appropriate scanner should be chosen for each vulnerability based on the assessment.It was concluded from the evaluation we conducted that the tested web scanners were unable to detect certain types of vulnerabilities that were accurate enough and seemed to work reliably for (SQL and XSS).Moreover, there were entire classes of significant vulnerabilities that were not well assimilated and explored. They were not detected either by scanners despite the danger they presented. For example there were other security risks presented in the OWASP TOP 10 2017 or vulnerabilities that target web 2.0 content and use AJAX technology. So, to solve the problem presented by those tools, we focused on the architecture of scanners to detect where those limits came from.The architecture of web application vulnerability scanners (WAVS), detailed in our article [1] and which is based on three components: Crawler module, Attacker module and Analysis module, requires additional research and contributions .This will help detect all the vulnerabilities present in the web application and will allow to improve the detection rate and the performance of scanners in general. This can be achieved by the use of new approaches and methodologies needed to improve the scanner architecture.

We started by the improvement of the crawler component, the first component of the architecture to explore the web application to recover and identify the web pages, the associated input vectors such as fields of input of the HTML forms, request parameters GET and POST and cookies. In the crawling phase, the scanner operations include browsing all possible links and web directories. By the end of this phase, the scanner identifies the entry points that require special input, such as the username and password. The forms and functions such as the GET and POST are also identified in this phase. Besides, the crawler creates an indexed list of all crawled pages. The scanner should also be able to identify the application structure and functionality to extract information that will be useful in the next phase. Most scanners require a more sophisticated Crawling mechanism to ensure that all content in a web application is analyzed. The detection of the presence of a web vulnerability depends essentially on the quality of this component. If the crawler is poor, the scanner will probably miss the vulnerability detection. For this purpose, this article will present a modelling by decision tree to visualize and trace the input vectors that can indicate the presence or absence of any vulnerability and therefore, improve the detection of vulnerabilities by scanners.

Manuscript published on 30 June 2019.
***** Correspondence Author (s)

Siham El idrissi*, LASTIMI Laboratory, Superior School of Technologies of Sale, Mohammadia School of engineering, Mohamed V University city of Rabat, Morocco

Ichrak lafram, LASTIMI Laboratory, Superior School of Technologies of Sale, Mohammadia School of engineering, Mohamed V University city of Rabat, Morocco.

Naoual Berbiche, LASTIMI Laboratory, Superior School of Technologies of Sale, Mohammadia School of engineering, Mohamed V University city of Rabat, Morocco.

Fatima Guerouate, , LASTIMI Laboratory, Superior School of Technologies of Sale, Mohammadia School of engineering, Mohamed V University city of Rabat, Morocco.

Mohamed Sbihi, , LASTIMI Laboratory, Superior School of Technologies of Sale, Mohammadia School of engineering, Mohamed V University city of Rabat, Morocco.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

Our modelling will allow us to modernize the Crawling phase so that it would recover and identify all the existing input vectors targeted by web vulnerabilities. This modelling was based on modelling studies already presented and published by several researchers in this field. The study of this research work allowed us, on the one hand, to raise their limits and on the other hand to propose improvements by modelling more adapted to web vulnerabilities and scanners that detect them.

The modelling (step 1) of our approach aims at extracting the input vectors from which we will build our dataset. The modelling will be the input of the learning system. Step 1 on the following figure shows the position of the modelling in relation to the rest of our work.

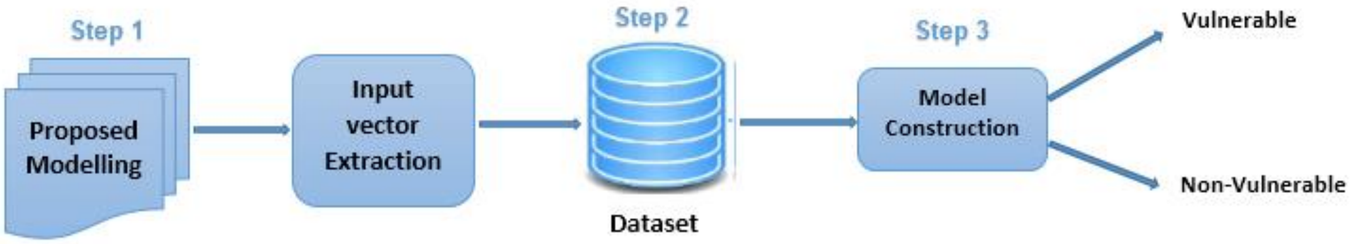

Fig .1. The Proposed Approach

II. RELATED WORK

Analysis of modelizations :

The purpose of this section is to provide an overview of existing classifications and see if they may be relevant for the vulnerability modelling.

Our analysis focused on two types of classifications: classifications derived from the work of academic researchers and classifications derived from the work of institutions and organizations specialized in web application security.

Now we will present the taxonomies that have been made by institutions and organizations specialized in web application security.

According to (WASC Treat Classification, WASC TC) the Web Application Security Consortium (WASC) provides a classification of 49 threats: These threats can be divided into two views : Enumeration view (Weakness threats , attack threats) and Development Phase view (include the source of the threats: Design, implementation or development).

The 49 threats can be grouped into six categories:

Insufficient authentication, Insufficient authorization, Client-side attacks, Command execution, Information leakage and Logical attack, [2].

Unlike WASC, which describes all possible attacks, the Open Web Application Security Project (OWASP) addresses only the top 10 security risks every 3 years. It publishes the ranking of the 10 most dangerous security vulnerabilities in the document "OWASP Top 10" and allows the project team to focus on protecting the web application against the most important threats. The OWAP considers the following attributes: Threat Agents, Attack vector, Weakness Prevalence, Weakness Detectability, Technical Impacts and Business Impacts [3].

The Common Weakness Enumeration (CWE) project aims to name describe and universally classify software security vulnerabilities, whether at the code level, at the design or architecture, such as buffer overflow, definition of password, cookie management, etc. More than 600 software vulnerabilities have already found their place in this new dictionary, in the form of detailed sheets. The vulnerabilities are organized in a taxonomy based on seven dimensions: Weakness, Abstraction, Structure, Description,

Relationships, Modes of Introduction and Common Consequences [4].

In what follows we will present the taxonomies resulting from the work of academic researchers.

Bishop's taxonomy can be interesting as it looks at the attributes that consider: nature of the fault (e. g. Buffer Overflow), phase of vulnerability introduction (e. g. during the design or implementation stage), operating domain (i.e. how to exploit it), effects domain (i.e. what is affected), minimum number of components necessary to exploit the vulnerability and source of its identification (i.e. the site or mailing list where the vulnerability was published) [5].

Alvarez has developed a multidimensional taxonomy .Each dimension represents a particular characteristic of the attack. The taxonomy follows the following rule: Each entry point has a vulnerability that threatens a service exploited by an action using an entry against a target that has a certain magnitude to obtain privileges [6].

Hansman's taxonomy has suggested four dimensions related to attacks: The vector or type (i.e. the means used by the attacker to achieve his goals, such as viruses, worms, denial of service), the target (e.g. operating, network protocol), the effects of the attack and the vulnerability exploited [7].

Simmons has proposed a cyber-attack taxonomy called AVOIDIT (Attack Vector, Operational Impact, Defense, Information Impact, and Target). Five main categories characterizing the nature of an attack have been used:

Attack vector, Attack target, Operational impact, Information impact, Defense mechanism [8].

Weber has presented a taxonomy based on three dimensions: The level of privilege required to conduct the attack, the way used by the attacker (e.g. exploitation of a software bug) and the desired effect (e.g. denial of service) [9].

Unlike the modelizations already mentioned, Howard's taxonomy has focused on the attack process, rather than the attack itself. It has taken into account the attacker, the tool he has used, the vulnerability exploited, the access obtained, the results of the attack found (i.e. disclosure, alteration) and its objectives (obtaining or destroying information)[10].

Before presenting our modelling, the following section will first discuss the limitations of the modelizations presented and analyze the attributes they propose in order to keep only the most relevant ones.

III. DISCUSSION

The different existing taxonomies adopt different perspectives that are based on attributes related to attacks or vulnerabilities. Even if it is impossible to mention all the existing taxonomies, we can globally identify the most important ones:

- Type of attack: virus, worms, Trojan horse, denial of service, etc....
- Attack detection technique: statistical approach, filtering.
- Signature of the attack: pattern or sequence of observed patterns.
- Tool used by the attacker: tool kit, script, user command, etc..
- Target of the attack: operating system, network protocol, application, service.
- Result of the attack: unlawful modification or disclosure of information, denial of service.
- Access targeted by the attack: super-user access, normal user access.
- Vulnerability exploited by the attack: memory overflow, wrong password strategy, wrong configuration, etc.
- Objective of the attack: financial gain, terrorism, self-satisfaction, etc....
- Security property targeted by the attack: confidentiality, integrity, availability.

It can be noted that the existing works lack clarity in the distinction between attributes. We also note that most modelizations focus on attack modelling and neglect vulnerability modelling. At the end of this discussion, it was found that existing taxonomies are not suitable for web vulnerability modelling. The reasons can be summarized in the following points:

- For most of them, they consider the vision of the attack and not that of the vulnerabilities; therefore, it is not surprising that the resulting attributes are less relevant for modelling Web vulnerabilities;

-Taxonomies have neglected a very important attribute, which is the target entry point because the main vector of attack against web applications is their own entries;

- Sometimes, the definition of attributes is a bit ambiguous and sometimes inconsistent;

- The number of resulting classes is sometimes very large, though the resulting complexity is unjustified;

- Unfortunately, these taxonomies are not accompanied by test cases;

In the rest of this study, we propose a new modelling that seeks to avoid these limitations. We will use the attributes we identified when studying existing modelling and eliminate those that are ambiguous and irrelevant to web vulnerability modelling. The selected attributes will be accompanied by a clear definition. Finally, we will combine our modelling with a test case. The ultimate goal is to provide WAVS test cases that are both relevant and representative of the different vulnerabilities.

IV. PROPOSAL OF A NEW MODELLING APPROACH

Let us begin by analyzing the attributes of the taxonomies already studied, in order to select only the ones most relevant from the "evaluator" point of view. Those that are incompatible with WAVS will, therefore, be excluded.

Modelling using dimensions :

A. Step 1 :

Extraction of all dimensions of the attack taxonomies studied.

TABLE 1 Summary of web attack taxonomies

Taxonomy	Dimension of taxonomy
WASC	**Development Phase view Enumeration view**
OWASP	**Threat Agents Attack vector Weakness Prevalence Weakness Detectability Technical Impacts Business Impacts**
CWE	**Weakness Abstraction Structure Description Relationships Modes Of Introduction Common Consequences**
Bishop	**Nature of the fault Phase of vulnerability introduction Operating domain Effects domain Components Source**
Alvarez	**Entry point Vulnerability Service Action Input length Target Scope Privilege**
Hansmann	**Attack vector Attack target Vulnerability Payload**
Simmons	**Attack vector Target Operational impact Informational impact Defense**
Weber	**Privilege Way Effect**
Howard	**Tool Vulnerability Action Target Unauthorized result**

B. Step 2 :

Choice of the most important dimensions:

-Way

-Privilege

-Impact

© Copyright: All rights reserved.

C. Step 3 :

Addition of new dimensions :

- -Target entry point (web application entry).
- - Execution mode.
- -Security service involved.
- -Type of vulnerability.

1) Dimension: target entry point (web application entry).

The main vectors of attack against Web applications are their own inputs, a wrong definition of these inputs can lead to the transition from an attack to the Web application. To protect the Web application from external threats, it is necessary to scan each component of the HTTP protocol constituting an input.

There are two types: http header and http verb

Inputs that may carry attacks on Web applications are shown in Table 2 [11].

TABLE 2
Web application inputs

Alias	General characteristic	Description
GET	**HTTP Query String Parameters**	**Input parameters sent in the URL**
POST	**HTTP Body Parameters**	**Input parameters sent in the HTTP body**
COOKIE	**HTTP Cookie Parameters**	**Input parameters sent in the HTTP cookie**
HEADER	**HTTP Headers**	**HTTP request headers used by the application**
XML	**XML Element Content**	**The content of XML elements**
JSON	**JSON Parameters**	**Parameters sent in JSON format**

a) Http verb :

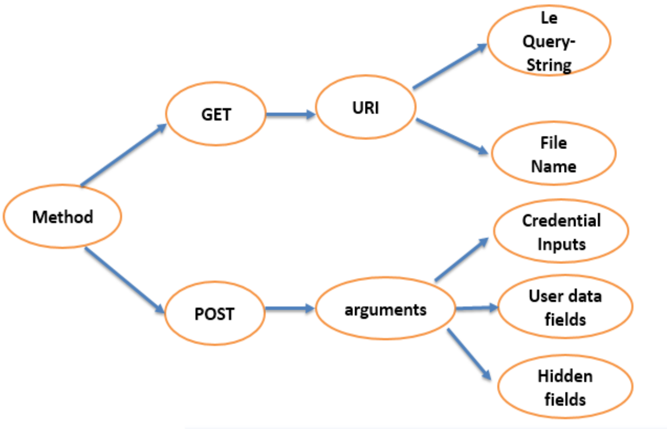

Fig. 2. Http verb

In the URI of a GET method, there are two possible entries [12]:

1. Query-String: It is the string of characters coming behind the ? or # in the URL of an HTTP request. Some web applications, such as search engines, retrieve the data of a query through the Query-string.

2. File Name: File names are also considered as an input parameter in a GET request. For coding ease, some developers prefer to manipulate file names to provide users with services and features without imposing strict control over these file names. This can have negative consequences on the security of the entire Web application ecosystem. Inclusion attacks (remote or local) is the favourite target of this entry. Directory traversals in the file system to access sensitive unsecured files can also be injected via this vector.

The arguments of an HTTP request using the POST method are the variables used by the Web application as input fields. These fields can be [13]:

1. Fields containing security credentials such as login and passwords. The format of these fields can be a plain text, code (base64), or a number.

2. User data entry fields usually in ASCII plain text.

3. Hidden fields (visually) used by developers to retrieve calculated data from a form. However, experienced users or attackers who use simple source code editing tools can modify these fields.

b) Http header :

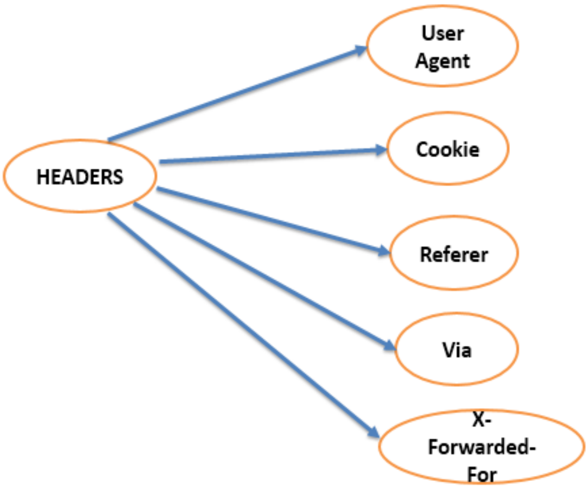

Fig. 3. Http header

HTTP protocol headers can also be retrieved by the web application to perform content formatting operations or for debugging purposes [14].

1. User Agent: It is the footprint of the web browser used by the web application to adapt the reduction (HTML/JavaScript/CSS) according to the version of the User Agent (Smart Phone, tablet, PC). This entry is vulnerable to attacks such as ShellShock, which inserts Shell code and affects the operating system on which the web application runs [15].

2. Cookie: It is used by the application to retrieve a session (authentication) cookie already registered in the cookie database. This entry is potentially vulnerable to attacks by SQL injection [16].

3. Referer: It is a means for the Web application to know the initiator of a request relayed by several proxy systems.

The content of this header is an IP address .However, an attacker can completely modify this address with his own IP address to retrieve the application's response instead of the legitimate initiator of the request. As it can also insert malicious code that will be interpreted by the application [17].

4. Via: The Via header, configured in an HTTP profile, concatenates information for each router in a response or request, separated by commas. [18]

5. X-Forwarded-For: It is an HTTP header inserted by proxies to identify the client's IP address. It can also be added to the request if other proxy servers proxify the application servers themselves. In this case, the requested IP address is always a local one. The client's IP address must be extracted from the request. This header can, therefore, contain several IP addresses.

2) Dimension Execution mode:

Two modes of execution can be distinguished:

- Server side execution :
  - SQL injection.

SQL injection is a class of attacks that exploits the lack of validation of web application entries to attack servers.

- Client-side execution:
  - Cross Site Scripting XSS.
  - Cross Site Request Forgery CSRF.

XSS and CSRF are the two classes of attacks that exploit the lack of validation of web application entries to attack clients.

3) Dimension Targeted security service:

Information system security has the following objectives [19]:

- Availability: the system must operate flawlessly during the intended use periods and guarantee access to the services and resources installed with the expected response time.
- Integrity: the data must be what is expected, and must not be illegal or malicious. In other words, the elements considered must be accurate and complete.
- Confidentiality: only authorized people can have access to the information intended for them. Unwanted access must be prevented.

Other aspects can also be considered as objectives of information security system, such as:

- Traceability: Ensuring that access and attempted access to the elements under consideration are found and that they are preservable and usable.
- Authentication: Identifying users is fundamental to managing access to relevant workspaces and maintaining trust in exchange relationships.
- Non-repudiation and imputation: No user should be able to challenge the operations he or she has performed in the scope of its authorized actions and no third party shall be able to take over the actions of another user.

4) Dimension Type of vulnerabilities :

There are three types [20]:

- Vulnerability in implementation: It is when the system design is secure, but the implementation does not respond, so vulnerabilities are present. For example, a program can be designed safely, while implementation contains bugs that can be exploited.
- Design vulnerability: It is when the fundamental design of the system is incorrect, so even a perfect implementation will have vulnerabilities. For example, a system that allows users to choose weak passwords.
- Configuration vulnerability: The system configuration introduces vulnerabilities. The system itself can be secure but when it is configured incorrectly, it may be vulnerable.

V. NEW MODELLING

1) New Vulnerability Modelling : Source (Figure 6)

Vulnerabilities were modelled in Figure 4 based on the Classification Tree Method (CTM) [21].Decision tree modelling will give us a clearer and more concrete vision of each dimension mentioned above. As shown in Figure 4, our classification is based on the following seven dimensions.

Execution mode, Entry point, Security service, Vulnerability type, Way, Privilege and Impact.

2) Experiments performed: source (Table 3)

In this section, we will test the vulnerabilities modelling we have performed by applying to the 10 most dangerous security vulnerabilities in the OWASP Top 10 2017 classification. The results we have obtained are described in Table 3.

Application of modelling on the following attacks [22]:

A1 - Injection
A2 - Broken Authentication
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

VI. CONCLUSION

We aim to propose new approaches and methodologies to improve the performance of WAVS (vulnerability detection rate). This can be done through improvements in scanner architecture. The detection of the presence of a web vulnerability depends essentially on the quality of the crawling component. If the crawler is poor, the scanner will probably miss the vulnerability detection. Our future work will consist of optimizing the detection rates of these scanners using machine-learning techniques. The use of these techniques will improve detection during the Crawling phase, which will allow us to obtain a more efficient vulnerability scanner capable of detecting all types of vulnerabilities without omitting any.

*Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)*
© Copyright: All rights reserved.

REFERENCES

1. El Idrissi, S., Berbiche, N., Guerouate, F., & Sbihi, M. (2017). Performance evaluation of web application security scanners for prevention and protection against vulnerabilities. International Journal of Applied Engineering Research, 12(21), 11068–11076.
2. The Web Application Security Consortium / Threat Classification,http://projects.webappsec.org/.
3. R. Jnena, 2013 Modern Approach for WEB Applications Vulnerability Analysis.
4. Common Weakness Enumeration Project (CWE), http ://cwe.mitre.org/.
5. Bishop, M. (1999). Vulnerabilities Analysis. Recent Advances in Intrusion Detection 1999, 125–136.
6. Álvarez, G., & Petrović, S. (2007). A Taxonomy of Web Attacks, 295–298.
7. Hansman, S., & Hunt, R. (2005). A taxonomy of network and computer attacks. Computers and Security, 24(1), 31–43.
8. Simmons, C. B., Shiva, S. G., Bedi, H., & Dasgupta, D. (2014). AVOIDIT : a cyber attack taxonomy. 9th annual symposium on information assurance (asia'14), 2–12.
9. Weber, D. J. (1998). A Taxonomy of Computer Intrusions.
10. Howard, J. D. (1997). Form SF298 Citation Data. Public Policy.
11. The Input Vector Support of Web Application Scanners : http://sectoolmarket.com/input-vector-support-unified-list.html.
12. HTTP request methods: GET https://developer.mozilla.org.
13. HTTP request methods : POST https://developer.mozilla.org.
14. HTTP headers https://developer.mozilla.org.
15. HTTP headers User-Agent https://developer.mozilla.org.
16. HTTP headers Cookie https://developer.mozilla.org.
17. HTTP headers Referer https://developer.mozilla.org.
18. HTTP headers Via https://developer.mozilla.org.
19. Information security : https://whatis.techtarget.com.
20. Akrout, R. (2013). Vulnerability analysis and evaluation of intrusion detection systems for Web applications, Rim Akrout to cite this version : HAL Id : tel-00782565.
21. Test design using the classification tree method: https://www.assystem-germany.com.
22. OWASP Top Ten Project: https://www.owasp.org.
23. Lahlayl, W., Berbiche, N., Guerouate, F., & Sbihi, M. (2017). Solving the Interoperability problem between UML modelling tools : Modelio and ArgoUML, 12(19), 8632–8641.

© Copyright: All rights reserved.

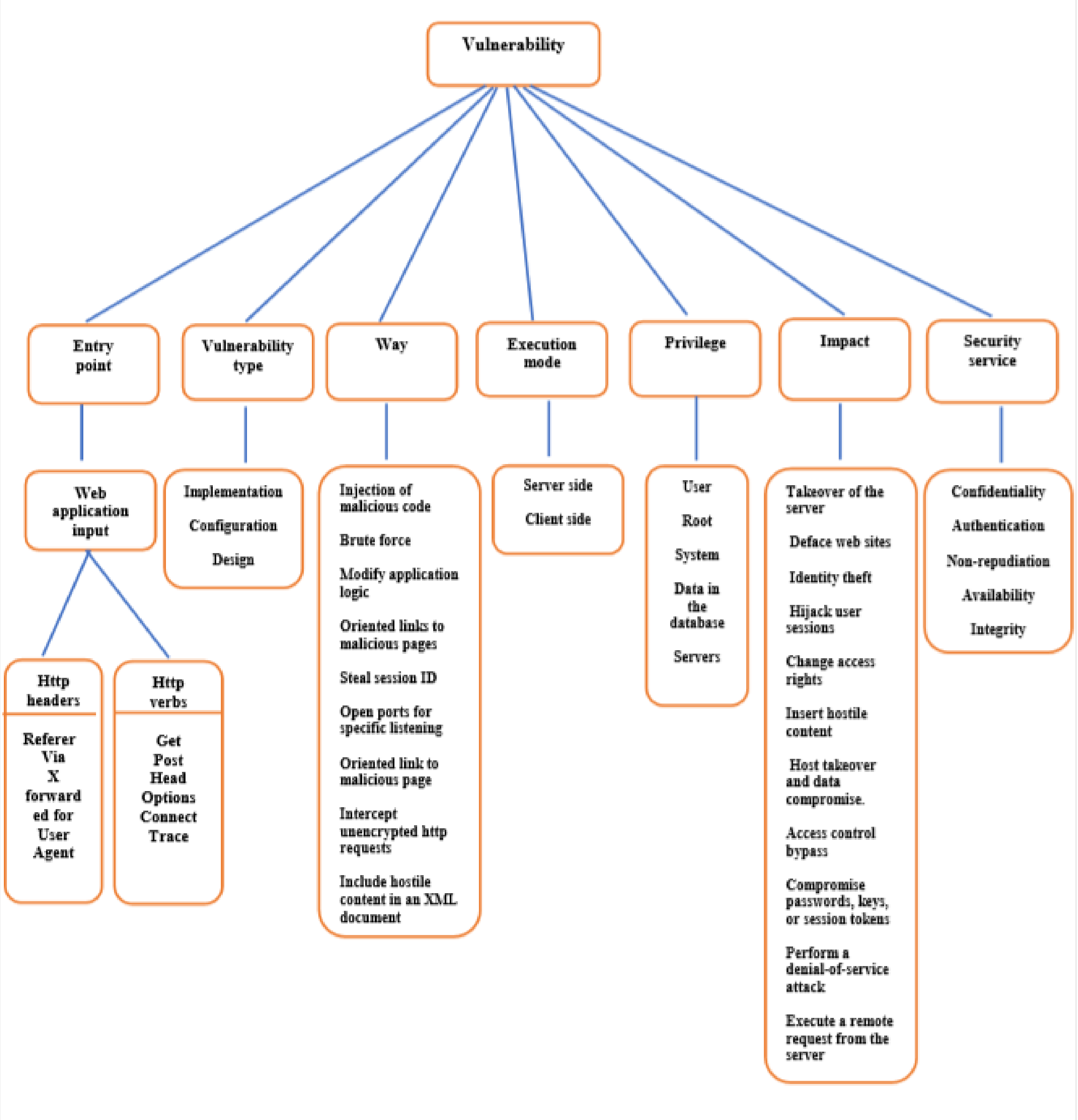

Fig. 4. A New Vulnerability Modelling

TABLE 3: Modelling results

Vulnerability name	Vulnerability type	Entry point	Way	Execution mode	Privilege	Impact	Security service
A1	Implementation	- Post-method http verb. - Get –method http verb	- Injection of malicious code	Server side	User Root System	- Loss or corruption of data - Loss of rights, or denial of access - Takeover of the server.	Confidentiality Availability Integrity
A2	Implementation	- Referrer http Header	- Brute force	Client side	User Root	- Compromise passwords, keys, or session tokens.	Authentication Confidentiality
A3	Configuration	- Get –method http verb	- Exploit the data in clear - Break the weak encryption keys	Server side	Registered data in the database	- Identity theft	Integrity Confidentiality
A4	Configuration	- XML Element Content	- Include hostile content in an XML document	Server side	Servers System	- Perform a denial-of-service attack - Execute a remote request from the server	Confidentiality Availability Integrity
A5	Implementation	- Get –method http verb - Put http Header	-Creating, accessing, updating or deleting every record.	Server side	User Root	- Unauthorized access to data or functionality - Change access rights	Confidentiality Integrity
A6	Configuration	- Get –method http verb	- Default password - Open ports for specific listening	Server side	Servers	- Unauthorized access to system data or functionality - Complete compromise of the system.	Availability Authentication
A7	Implementation	- Post-method http verb - Get –method http verb	- Injection of malicious code	Client side	User	- Hijack user sessions - Deface web sites - Insertion of hostile content - Redirection of users to malicious site	Integrity Confidentiality
A8	Configuration	- Post-method http verb - Get –method http verb	- Modify application logic	Server side	User Root System	- Remote code execution	Availability Integrity
A9	Configuration	- Post-method http verb	- Use of manual analysis or scanners to identify a vulnerable component	Server side	Servers	- Host takeover and data compromise	Availability Integrity Confidentiality
A10	Implementation	- Post-method http verb - Get –method http verb	- Attack systems	Server side	Servers System	- Extract or destroy data	Availability

© Copyright: All rights reserved.