# Cross-Site Scripting (XSS) Attacks And Mitigation: A Survey

**Article** *in* Computer Networks · November 2019

DOI: 10.1016/j.comnet.2019.106960

German Rodriguez
Universidad de las Fuerzas Armadas-ESPE

Jenny Torres
Escuela Politécnica Nacional

Eduardo Benavides
Universidad de las Fuerzas Armadas-ESPE

# Cross-Site Scripting (XSS) Attacks And Mitigation: A Survey

Germán Rodríguez
Faculty of Systems Engineering
Escuela Politécnica Nacional
Quito, Ecuador
german.rodriguez@epn.edu.ec

Jenny Torres
Faculty of Systems Engineering
Escuela Politécnica Nacional
Quito, Ecuador
jenny.torres@epn.edu.ec

Pamela Flores
Faculty of Systems Engineering
Escuela Politécnica Nacional
Quito, Ecuador
pamela.flores@epn.edu.ec

***Abstract*—The results of the Cisco 2018 Annual Security Report show that all web applications analyzed have at least one vulnerability. It also show that web attacks are becoming more frequent, specific and sophisticated. According to this report, 40 % of all attack attempts lead to a method known as Cross-Site Scripting (XSS), which was the most used technique. This special type of attack occurs when an attacker uses a web application to send or execute malicious code on a user's computer. In this context, we have analyzed a total of 53 documents to collect information on the tools and methods that the community has used to detect and mitigate these attacks. Our results show that the trend is increasing in the analysis of content and patterns, and there is a low tendency in the use of artificial intelligence techniques to detect or mitigate these attacks. As a complement, we propose our future line of research based on the gaps present in the existing solutions proposed by previous research projects.**

***Keywords*—*XSS, OWASP.***

## I. Introduction

The application layer has support a large number of attacks, with a large-scale increase, it has become a common threat to web security. Techniques such as malicious codes that are vulnerable are used with the aim of penetrating and paralyzing a website. They have also been used from low-level attacks to high-level data breaches exposing the infrastructure of web applications [1]. This represents the greatest security concern for a large number of applications, especially those that are implemented in high availability operations or priority services, such as medical care, banking, e-commerce, etc. [2].

According to the annual global security report 2018 [3], derived from the analysis of billions of security events recorded around the world, all tested applications show at least one vulnerability. The average is 11 failures per application. This report shows that web attacks are becoming more specific, more frequent and much more sophisticated. Many incidents of rape show signs of careful and prior planning by cyber criminals who investigate their victims more thoroughly.

One of these web attacks is known as Cross-site Scripting (XSS), with a 40 % attack attempts, followed by SQL injection (SQLi) with 24 %, an attack called cross-section with a 7 %, the inclusion of local files (LFI) with a 4 % and in the last position is the denial of services distributed (DDoS) with 3 %. XSS attacks occur when a web application is used to send or execute malicious code, usually in the form of a script, from the browser on the victim's computer. With this execution you could filter the personal information, or, steal the user [4] cookies to hijacking the identity in a fraudulent session, so it offers the attackers the possibility of stealing sensitive data or even being able to take control of certain devices.

This year, there has been a record number of vulnerabilities in web applications that include this category (XSS), but also new categories such as insecure deserializacion [5]. According to data from Imperva [6] the vulnerabilities of cross-site scripting or XSS represent the highest number of web application vulnerabilities in 2017. In fact, their number has doubled compared to 2016. And according to Imperva's predictions, they will follow being the most frequent offensives in 2018.

Currently, implementing server-side solutions to protect web applications is no longer profitable, according to [7], because developers do not always have code assurance experience. Therefore, the providers of browsers such as Firefox, Chrome or IExplorer have tried to develop filters to act on the client side and thus defend against these attacks. Some papers such as [8] propose the incorporation of adequate prevention measures during the software development cycle, thus avoiding potential damages. We have also found proposals to apply static analysis, dynamic analysis or a combination of these. However, according to [9] dynamic analysis approaches incur overhead, on the other hand, existing approaches to static analysis lack precision in the identification of XSS vulnerabilities.

In this context, we have set as a goal to find the methods and tools that have been used or proposed to detect and mitigate this type of attack. We have searched the different scientific bookstores and have oriented our research to the search for tools that have been proposed or used. From this search we have filtered the works that have used some artificial intelligence method. Our contribution shows as a result the current trend in the use of traditional methods vs methods that use artificial intelligence.

The rest of the document has been structured as follows: section II summarizes a background on cross-site scripting attacks, section III classifies the attack types derived from XSS and describes some examples of exploitation, section IV talks about the theft of cookies through XSS attacks, section V analyzes all the methods and tools found in the literature, section VI presents a discussion of the information found and finally in section VII discusses the conclusions of the proposal and future work.

## II. Background of Cross-Site Scripting Attacks

Web applications are insecure by default because their developers generally do not establish secure development protocols, this contributes to the theft of personal and crucial user information [10]. This failure is considered a vulnerability because it allows an attacker to send malicious code (usually in JavaScript format) to another user. If the website is not programmed correctly a hacker could take advantage of this flaw in the application to start and execute the malicious code in the systems, with the possibility of scaling through the network of an entire organization. An attacker writes a script and injects it into the domain, with the goal of obtaining information. The most popular attack vector is currently a web browser due to the growth of access to the Internet. In conclusion, most web applications have vulnerabilities in their source code, increasing the possibility of exploiting those flaws and exploiting them.

Some consequences of a successful malicious attack are the manipulation of social network sessions of any victim, the theft of cookies to impersonate the identity of a legitimate user, or simply, the control of a browser with or without the consent of the victims. According to the latest security statistics report in web applications [11] in this type of attacks the victim is the user and not the applications. XSS is in second place as one of the most serious vulnerabilities, with approximately 38% critically. However, what is worrying is that this type of attacks has a very low remediation rate and / or solution.

On the other hand, according to the OWASP Top 10 Web Application Security Risks - 2017 cite Online5, XSS attacks dropped to the 7th place compared to the top 10 of 2013 (third place). In summary, it has a rating based on the following parameters:

- Exploitable
- Weakness Prevalence
- Weakness Detectability
- Technical Impacts
- Business Impacts

In the case of exploitability is presented because there are automated programs that can detect and exploit the 3 types of attacks XSS (persistent, non-persistent and DOM), in addition there are free available many frameworks to exploit this type of vulnerability. In relation to prevalence and detectability, XSS is the second most frequent problem in OWASP's Top Ten reports, and is found in about two-thirds of all applications. Programs can find XSS vulnerabilities automatically, particularly in mature technologies such as PHP, J2EE/JSP and ASP/.NET.

According to the Common Weakness Enumeration CWE / SANS TOP 25 Most Dangerous Software Errors [12] the top 25 software errors are listed in three categories:

- Unsafe interaction between components (6 errores)
- Risky Resource Management (8 errores)
- Porous defenses (11 errores)

According to the CWE, XSS is in the category insecure interaction between components, identified as CWE-79. In the same way, it has a rating according to the following parameters: Weakness Prevalence, Remediation Cost, Attack Frequency, Consequences, Ease of Detection and Attacker Awareness (Table 1).

TABLE I. Summary

| Parameters | Qualification |
|---|---|
| Weakness Prevalence | High |
| Remediation Cost | Low |
| Attack Frequency | Often |
| Consequences | Code execution, Security bypass |
| Ease of Detection | Easy |
| Attacker Awareness | High |

Cross-site scripting (XSS) creation occurs frequently when [13]:

- From a web application, untrusted data can be entered into a web application.
- A web page containing this untrusted data can be generated dynamically by a web application
- While this page is generated, the application does not prevent the data from containing any type of content (JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX) that can be executed by any web browser.
- The victim can visit the website that was generated through the web browser infected with the malicious script, injected using the data that was not trusted.
- This script is executed in a web page that was sent by a web server, so the victim will execute the malicious script in the same context of the domain of the web server.
- This alters the same policies of the browser that the sequences of the commands of a domain can not be executed in a different domain.

These attacks do not necessarily exploit the holes in a specific browser. It affects all web servers and browsers currently on the market. According to [14] you could list the impact of cross-site scripting as follows:

- The content of a web application could be altered and could be used to inject various ads, influence the reputation of commercial websites or deceive the user.
- Steal session cookies from open sessions and extract information while these sessions stay online.
- Steal or impersonate the identity of legitimate users, by stealing confidential personal information.
- The user's HTTP sessions may be compromised or hijacked if the attacker misuses this stolen information.

## III. Types of attacks and exploitation examples

According to the literature, there are 3 types of attacks [15]: persistent XSS, non-persistent XSS and Document Object Model (DOM) XSS. Similarly, possible gaps or weaknesses are found in software, hardware, and even people (developers) who are part of a computer environment. XSS takes advantage of the lack of mechanisms to filter and validate input fields (text boxes for example), present in web forms, this allows sending complete scripts. These scripts are stored in text files as instructions that are interpreted line by line in real time for execution.

### A. *Types of XSS attacks*

*1) XSS non-persistent, reflected or indirect:* In this type, the attacker places a script to steal the victim's cookies, in order to personify himself as if it were his session. With the cookie received, the attacker could execute actions using the permissions of the victim without using any type of password. This attack is common in search engines, usually, the code is injected through forms, URLs, cookies, flash programs or even videos. This attack exploits vulnerabilities in Web applications that use (or reflect) information provided by the user to generate an exit page.

In this way the code is redirected by means of a third mechanism. For example, through spoofing (e-mail). With this an attacker can convince a user to click on a link in the message to execute any JavaScript code. The consequence is the redirection of the user's traffic to a web application of the attacker. If said Web application presents an XSS vulnerability, its execution will be carried out within the trusted environment of the Web site that hosts the application.

*2) Persistent or direct XSS:* The attacker injects malicious HTML code directly into the web page or site that allows it (vulnerable site). In this attack requires programming tags (script like JavaScript). These codes are made permanent on the web for all users after running in a first attack. As a result, whenever someone enters a section where there is an injected code, this will be executed in their browser and they will comply with the actions programmed in their script. This variant is more dangerous because it is based on the injection of malicious code in the content that is stored in the servers of the external web applications. That is, the data sent by the attacker is stored permanently on the server and later displayed to users who visit the website. Among the consequences are: allow the execution of code to obtain or elevate elevated privileges. The default users have their administrator account activated. For this, it is always recommended to disable the execution of JavaScript of the browsers, however a deeper analysis is required due to the need of interaction with the websites.

*3) DOM XSS:* Known as Type 0 or DOM-Based XSS, it is considered a more complicated and little known or common attack. The difference is that the malicious code is injected via the URL but it is not loaded as part of the web in its source code. Its detection is more difficult since the malicious load does not reach the server. It is considered a local XSS because the damage is caused by the scripts that are on the client side. Basically when an infected page is opened, the malicious code exploits some vulnerability to install itself in a file of the web browser and it is executed without any previous verification. Unlike the direct and indirect attacks on this server is not involved. However, like the reflected XSS, this attack requires the user to click on a link, so a script on the web page selects the URL variable and executes the code it contains. This method is more effective in stealing session cookies.

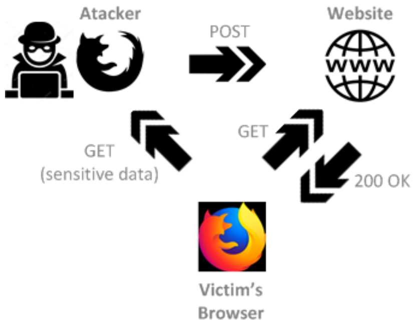
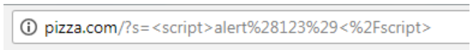


Fig. 1. Scenario for a reflected or indirect XSS attack



pizza.com/?s=<script>alert%28123%29<%2Fscript>

Fig. 2. Sample XSS code reflected injected by URL

### B. *Differences between attacks*

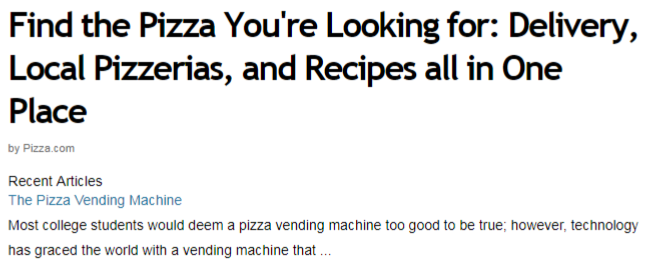
To understand DOM XSS, it is required to understand the difference between indirect and direct XSS compared to DOM. They are basically differentiated by the place where the attack is executed, on the one hand, indirect and direct XSS is performed on the server side while in DOM the server does not intervene, ie DOM is a problem of lateral injection of the client (browser ). Since the code originates in the server, it is the responsibility of the developer of the application to protect it from these attacks, regardless of the type of XSS failure.

We must always remember that these attacks are executed in web browsers. Another difference is in the place where the attack is injected into the application. With the first two the attack is injected during the processing of the requests that originate through the entries that are added using HTML. With DOM, the attack is injected directly into the application during run time on the client.

### C. *Common scenarios for running XSS attacks*

*1) Scenario for the reflected XSS attack:* The simplest scenario is shown in Fig 1, to execute this attack only a web page is required and enter code through its search engine. For example, if the following malicious script is injected into the application, as shown in Fig 2, with the aim of showing an alert in the web browser, the script that goes after the domain pizza.com it is the easiest to execute (in this case because sites vulnerable to XSS attacks were consulted).

Fig. 3 shows a preview of the page whose domain is pizza.com, for an example attack, in the code shown in Fig.2, the bottom of the web page is changed to black as shown in Fig. 4. For a second test, the Google Chrome browser blocks the execution attempts and there are no more results, this can be seen in Fig. 5 whose message indicates that an unusual code has been detected on the page and has been detected. locked.



Find the Pizza You're Looking for: Delivery, Local Pizzerias, and Recipes all in One Place

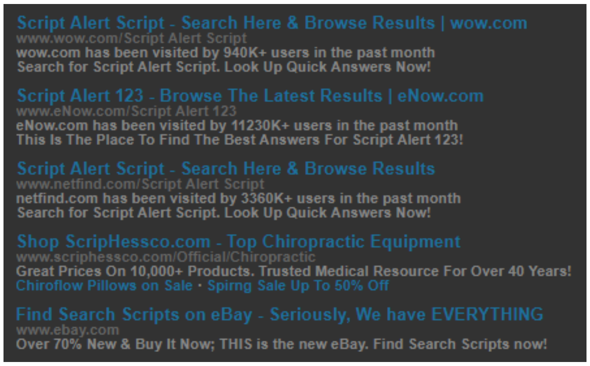
by Pizza.com

Recent Articles

The Pizza Vending Machine

Most college students would deem a pizza vending machine too good to be true; however, technology has graced the world with a vending machine that ...

Fig. 3. Pizza.com domain before the XSS attack



Script Alert Script - Search Here & Browse Results | wow.com
www.wow.com/Script Alert Script
wow.com has been visited by 940K+ users in the past month
Search for Script Alert Script. Look Up Quick Answers Now!

Script Alert 123 - Browse The Latest Results | eNow.com
www.eNow.com/Script Alert 123
eNow.com has been visited by 11230K+ users in the past month
This Is The Place To Find The Best Answers For Script Alert 123!

Script Alert Script - Search Here & Browse Results
www.netfind.com/Script Alert Script
netfind.com has been visited by 3360K+ users in the past month
Search for Script Alert Script. Look Up Quick Answers Now!

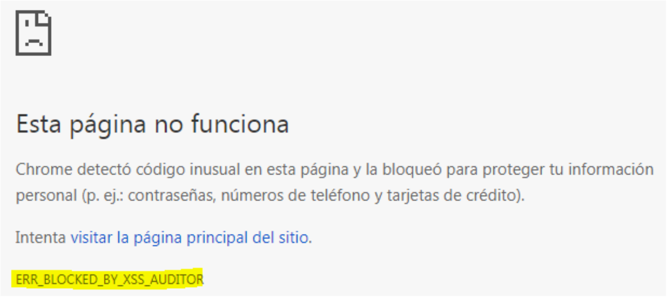
Shop ScripHessco.com - Top Chiropractic Equipment
www.scriphessco.com/Official/Chiropractic
Great Prices On 10,000+ Products. Trusted Medical Resource For Over 40 Years!
Chiroflow Pillows on Sale · Spirng Sale Up To 50% Off

Find Search Scripts on eBay - Seriously, We have EVERYTHING
www.ebay.com
Over 70% New & Buy It Now; THIS is the new eBay. Find Search Scripts now!

Fig. 4. Pizza.com domain after the XSS attack

*2) Scenario for the stored XSS attack:* As shown in our previous study [16] to execute an XSS attack it is required to use a little social engineering so that the user, through a convincing link, access a contaminated page with a JavaScript type file (* .js ) Fig. 6, which infects the victim and establishes a link with a main controller. This attack uses ignorance of the user when accessing a reliable page. In this test Beef software was used, which stores a vulnerability in a web page that turns machines into zombie teams. This link is established even after closing the victim's browser. Being a framework to make security tests to web applications could run a large number of attacks such as exploit extensions of programs such as Adobe, Flash, steal cookies, display alert messages, make the user believe that his Facebook session he closed and thus steal his credentials, etc.

*3) Scenario for the DOM XSS attack:* The DOM-based XSS takes advantage of the JavaScript type scripts that are vulnerable and run directly in the user's browser, as shown in Fig. 7. For example, a vulnerable script can be used to start an XSS attack.



Esta página no funciona

Chrome detectó código inusual en esta página y la bloqueó para proteger tu información personal (p. ej.: contraseñas, números de teléfono y tarjetas de crédito).

Intenta visitar la página principal del sitio.

ERR_BLOCKED_BY_XSS_AUDITOR

Fig. 5. Chrome browser response to XSS attack

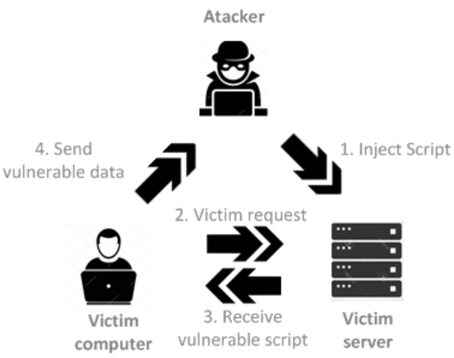


Fig. 6. Scenario for executing persistent or direct XSS attacks

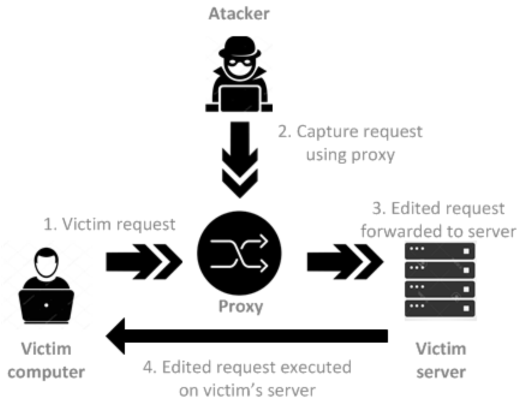

Atacker
2. Capture request using proxy
1. Victim request
3. Edited request forwarded to server
Proxy
Victim computer
Victim server
4. Edited request executed on victim's server

Fig. 7. Scenario for DOM XSS attack

There are two commonly used methods for this attack:

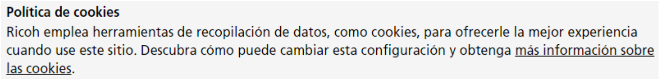
- Click on a URL link sent in an email
- Click on a URL link while visiting a website

In both cases, the URL will be linked to the trusted site, but it will contain additional data that is used to trigger the XSS attack. It is important to mention that SSL connectivity does not protect against this problem.

### *D. Most common consequences of attacks*

It details different consequences associated with XSS attacks.

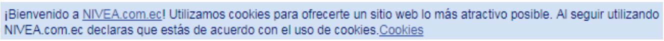
- **Disclose information stored in users' cookies**: A malicious user could create a script on the client side, which, once executed, performs some activity, such as sending all cookies from the site to a specific email address. The consequence of this script is that it will be uploaded and executed for each user who visits the website.
- **The consequence of an XSS attack is the same regardless of whether it is stored or reflected**: The difference is in how the payload affects the server.
- **Handling** : Some vulnerabilities can be exploited to manipulate or steal cookies, create requests to confuse



**Política de cookies**

Ricoh emplea herramientas de recopilación de datos, como cookies, para ofrecerle la mejor experiencia cuando use este sitio. Descubra cómo puede cambiar esta configuración y obtenga más información sobre las cookies.

Fig. 8. Alert on the policy of use of cookies in browsers



¡Bienvenido a NIVEA.com.ec! Utilizamos cookies para ofrecerte un sitio web lo más atractivo posible. Al seguir utilizando NIVEA.com.ec declaras que estás de acuerdo con el uso de cookies.Cookies

Fig. 9. Welcome message and notice of the use of cookies

or pose as a valid user of the system, compromise confidential information or execute code in the user's systems.

- **Other harmful attacks include**: Disclosure of user files, installation of Trojan programs, redirect the user to another page, execute controls ”Active X” (under Microsoft Internet Explorer) from sites that a user perceives as reliable and modify the presentation of content.

### *E. Policies for the use of cookies*

Alert messages like:

- This website uses cookies to ensure you have the best experience on our website (Fig. 8).
- We use our own and third party cookies to improve our services by analyzing their navigation habits (Fig. 9).

Currently they appear in most visited web pages as a pop up or a notification banner. However, what is worrying is the way in which this message is accepted with the following warning: *If you continue browsing, we consider that you accept its use*.

It means that although the user does not click on the OK button, only the action of navigating the site will install the cookies that were warned in the informative messages. This new mode of navigation makes web pages install cookies compulsorily on the user's computer visit. As mentioned in the previous sections, a successful XSS attack could steal the user's cookies and show an active session to pass as a legitimate user. In the next section we talk about what cookies are and how they could be a target of these attacks.

## IV. Cookie Theft through XSS attacks

These attacks are popular for stealing cookies from a browser's database. Thus, an attacker executes an arbitrary script and personal information is extracted from the victim's computer. The execution of the attack is to use some weakness or vulnerability.

### *A. What are cookies?*

Cookies, by default, are the mechanism to maintain session authentication between a user and web applications. However, they have inherent security weaknesses that allow attacks against the integrity of these sessions. It is always recommended to use the HTTPS protocol to protect cookies, but the costs of implementation, support and performance are a challenge, especially for applications that have a high degree of distribution. On the other hand, cookies can be exposed in many ways even when the HTTPS protocol is enabled [17]. Most cookies are stored within users' computers in the form of text files or small databases, such as in SQLite format (Mozilla Firefox). Its objective is to store information related to navigation preferences, sessions, credentials, etc. They store data such as the type of browser or the type of device that was used to access the website. In summary, cookies are like a memory for websites.

The preferences of the visit of a web page are established through the cookies, the visualization is personalized if a user visits this page a second time, in this way the experience is improved. The most outstanding is the advertising or ads that are inserted in all websites, or also called advertising or third-party sites. Our data is exchanged between the visited page and an advertising page, for example Facebook. The difficulty is in the analysis of the content of cookies, since we can not determine if they are harmless or if they send information to a third party, so the end user should trust the reputation of the website he visits.

They have a simple text format, and usually cookies are not some kind of virus, or do not contain automatic execution codes. They can not auto-replicate or spread across the network to run or reproduce again. However, they can be used as a spyware method, in addition, there are many anti-spyware products that could avoid this problem and block or delete some cookies in order to destroy them if the user so requires. Also options or security complements are included in most browsers to grant a certain level of use and access to cookies. The advantage is that you can control the validity time and the elimination of cookies, but not the information you can share with another website. In fact, the protection of privacy should be valued as a right of every Internet user. This opens a new vision to be protected against the threats presented by the use of cookies.

### *B. Most common types of cookies*

Below is a general description of the most common types of cookies:

*1) **Persistent Cookies**:* This type of cookies registers an expiration date, and they are destroyed by the user's equipment when they reach this expiration date. However, there is no record with a date that limits this expiration time. These cookies track the user's behavior to understand the tastes of the people. This activity is known as Web Analytics [18].

*2) **Secure Cookies:**:* are transmitted through the HTTPS protocol, and are found in government pages, banks, hospitals, services or transactions, commerce, email, etc. They travel encrypted offering a level of security in the communication between the website and the user's browser.

*3) **HttpOnly Cookies:**:* This type of cookie should be the most used, however it is not, because it prevents any type of malicious script, for example JavaScript, from accessing the content of the cookie, which means protection against XSS type attacks. they have been explaining. This would prevent a functional attack from sending the cookie information to a third party (third party cookie).

*4) **First party cookie:**:* known as first-person cookies because they offer some privacy, they do not send data to other sites. Only the host where the cookie was created can access the information on it. The simplest way to check that a cookie of this type is to compare the host domain (within the parameters of the cookie) with the browser bar, if they are the same then it is effectively a first person cookie. However, this does not guarantee that our information may be sold to third parties through an attack that recovers or hijacks our cookies.

*5) **Third party cookie:**:* or third-party cookies, unlike the previous ones, these are related to third-party domains that collect our data. They are present on websites in the form of links, tags or scripts that give new features such as ”like” buttons or connection with social networks. Its disadvantage is that many popular websites filter our identity with users or connected applications through spyware that receives unencrypted traffic [19].

They are used for advertising, so advertisers could have information about the brand or model of our devices, for example. This tracking helps to create a profile of the behavior of the users and so the programs orient their announcements according to the interests that are discovered. The problem is in the invasion of our privacy, since it is an intrusive method to send information. This has encouraged the development of new laws such as the EU law The Cookie Law [20] on cookies.

*6) **Session Cookies**:* These cookies are temporary, they are stored in the memory of the browsers and they are destroyed when they are closed, this is their biggest advantage. Their disadvantage is that they store information such as the login credentials of a session (for example the e-mail) and could be stolen to obtain this sensitive data. There is a special type of stateless session cookies that allow web applications to modify their behavior based on the user’s preferences and associated access rights, avoiding maintaining the status of the server for each session [21].

*7) **Zombie cookies**:* These cookies are dangerous because they are recreated after being deleted, the browser has no power over these because they are rebuilt independently of the browser. They are stored in the devices and not in the browsers, you can access them regardless of the type of browser used. They are a threat to the user’s privacy and security, which is why attackers look for them or create them for illegitimate and malicious purposes.

### *C. Statistics to remedy cookie theft*

Currently, most web applications use cookies to maintain the status of the session with the user, that is, the cookies are sent after the user has authenticated (session cookie). For a later connection no additional authentication will be needed because the validated cookies will only be verified to allow the new request. This authentication functionality makes cookies a potential target for attackers, because they are created by websites and contain small amounts of data that can be sent between a sender and a receiver. According to their browsing habits, cookies can identify users by storing their activity history of a website, this in order to offer more specific content according to their preferences.

For example, a user visits a web page for the first time, and a cookie is saved on his computer. If the user visits the same page later, the server of the website requests the same cookie to update it with new configurations of the site, this is how the user’s visit becomes so personalized.

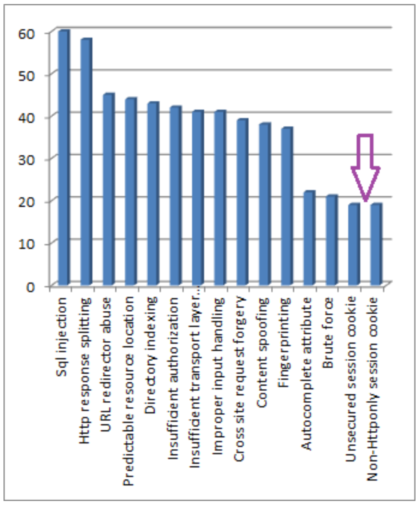


Fig. 10. Current remediation rate according to the type of vulnerabilities

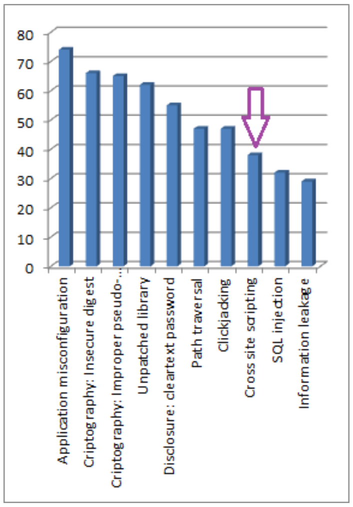


Fig. 11. Remediation rate according to the class of vulnerabilities for XSS

As seen in Fig. 10, the remediation rate according to the class of vulnerabilities for insecure session cookies and for session cookies of non-secure type, have less than 20% remediation [22].

In addition, in Fig. 11 developers are oriented to always fix or remedy the most accessible or easiest problems. For example, incorrect settings of applications have a 74% remediation, as well as unpatched libraries (62%). However, the most complex and difficult solutions are still those of type XSS (38%) and SQL injection (32%) [22].

## V. Analysis of Methods and Tools

Some of the schemes proposed for automated removal and other related vulnerability analysis techniques are discussed briefly in the following subsection.

In [23] they propose a tool called QualysGuard that is oriented to minimize the vulnerabilities and the causes of the damages that certain web applications have. This tool acts as a web scanner. Its function is to find that the controls of a login form are entered in the appropriate form, for example if an attacker makes a modification to the URL of the form, this tool will detect it, scan this URL and detect what type of vulnerability belongs It is a tool that satisfies the needs of the server side and the client. Its main limitation is that it is not 100% functional unless the client does a manual scan of the URLs that he needs to visit.

In [24] a more automatic approach has been proposed to detect this vulnerability that arises due to the incorrect coding of data that is not reliable. Your technique can determine XSS vulnerabilities on day 0 that other static analysis tools have not found. Its biggest advantage is that it is oriented to the developers since they could detect and correct the problems without depending on the security experts, which represents a saving of resources. However, its disadvantage is that it is oriented only to one type of XSS attack.

In [25], the performance and detection capabilities of the latest security scans of Black Box web applications against stored SQLI and stored XSS have been analyzed. Previous research has shown that Black-Box scanners have a relatively poor performance in detecting these two vulnerabilities. Through the development of a customized test bench, in order to compare the capabilities of the black box scanners. The challenge they present is in the choice of attack vectors that are suitable for the detection and exploitation of XSS stored in black boxes.

There are also dynamic detection methods that are based on the simulation of the behavior of browsers. In [26] they have proposed to find hidden XSS injection points, with greater accuracy, by expanding the detection coverage. This proposal acts as a more portable system, which facilitates the development of systems. This system is oriented to interpret the JavaScript code and recover the Ajax content to find the injection points hidden in the pages, through a browser that acts as a web crawler without headers. Its disadvantage is that it uses a framework to launch attacks on pages with pre established vulnerabilities.

It could be considered as another dynamic method to the study presented in [27] who have proposed an optimal repertoire of attack vectors. By generating these vectors, attacks could be executed automatically, dynamically detecting XSS vulnerabilities in applications. To do this, they use machine learning algorithms to improve efficiency in detecting XSS vulnerabilities. Its advantage is the generation of XSS attack vectors automatically, and its disadvantage is that it has only been tested on 24 websites, that is, it is limited to a number of pages.

In [28] a combination of evolutionary fuzzing with inference models to detect XSS injection vulnerabilities through the generation of test inputs has been proposed. According to their model, these entries are generated by genetic algorithms through a formal learned model, so there is an automatic generation of inputs to activate instances of the vulnerability vulnerability. This is how they have proposed an intelligent or intelligent fuzzing approach to exhibit deeply integrated injection vulnerabilities. This helps in the automated search of XSS type 1 to generate test cases..

Another technique to detect and prevent XSS vulnerabilities is presented in [29]. In a first phase, the web application has been translated into a language by means of a conical tool and thus the input and output variables that are used are identified, thereby generating test cases and determining the input / output dependencies of the application. These dependencies may indicate vulnerabilities in the application. In a second phase, monitors have been included to implement the code automatically, thus verifying exploitation at execution time. This solution is able to identify the entries that take place in the form of malicious chains by union of benign chains or by conditional copies.

These vulnerabilities can be expanded if the new HTML5 standard is used, because there is greater interactivity with the web page. In the work presented in [30], 14 XSS attack vectors that relate to HTML5 have been identified, using a systematic analysis of new labels and attributes. With this data, a repository of XSS test vectors has been built to implement a dynamic tool to detect XSS vulnerabilities focusing on Web mail systems. Applying this tool to some popular Web mail systems have managed to find exploitable vulnerabilities of XSS.

In [31] present a scanning tool, compared to [30] is distributed and tracks modern web applications on a large scale, however, it only detects and validates XSS vulnerabilities of the DOM type. According to the authors, their proposal does not produce false positives, in addition to being scalable. In this way, a database of attack vectors is provided where people can search their website and correct vulnerabilities, and thereby offer a more secure Internet. This tool is functional because it serves to analyze data on a large scale, this thanks to the use of a technique of hooking in the steps of tracking and scanning.

Another of the proposed tools is called DomXssMicro [32] which is a micro benchmark based on a template that has been extracted from representative vulnerabilities. This proposal consists of six orthogonal components called source, propagation, transformation, sink, trigger and context). Using this tool they test a specific property of XSS based on DOM with a total of 175 test cases. However, it is an empirical study to evaluate detection tools based on DOM-type XSS, including open source and commercial ones. Its disadvantage is that currently the tool is not complete.

By means of text mining in [33] a proposal is presented to detect code files vulnerable to XSS in web applications. To do this, they apply a process of using tokens adapted to extract the characteristics of the text, which in this case would be the source code of web applications. In this process, each code file is transformed into a set of unique text features with associated frequencies and thereby builds vulnerability prediction models.

Another proposal that uses pattern analysis to detect and mitigate XSS vulnerabilities is presented in [34]. In addition, they also use static pollution analysis. According to their study, most existing approaches have limitations in terms of false-positive and negative-type results. Their proposal acts as a prototype that evaluates a set of public data, however they only focus on web applications developed in PHP because they represent the highest percentage of the programming languages of web applications on the server side. Your prototype is programmed using the C # language.

Similar to the proposal in [34] there are approaches that aim to minimize false-positive and false-negative results. Thus, a detailed analysis using defensive programming [35] is based on the sensitive context vs. the HTML context to accurately detect XSS vulnerabilities. With this method, accurate detection has been achieved from the source code of applications that use PHP and can also provide suggestions for improving vulnerable source code.

In [36] a design has been proposed through proxy-level development to detect XSS attacks based on the KullbackLeibler Divergence (KLD) measure. This proposal has been oriented to web applications with open source PHP code that have presented XSS vulnerabilities. This approach is based on the distance between the probability distribution of the legitimate JavaScript code and the observed JavaScript code present in a response page. The deviation between the two types of JavaScript results in a high KLD value.

On the other hand, the use of standards and recommendations such as OWASP are also proposed for developments with J2EE [37]. For this purpose, design specifications such as DAO and Facade and WS-Security framework (MVC) have been used as architectural standards. These specifications have served to validate the prototype at the level of design, coding and security. These bases have allowed to demonstrate that the use of standards, techniques and recommendations are necessary to avoid XSS vulnerabilities, being a static analysis, it supports in the identification of security breaches and aspects of quality that many times are not considered by the developers.

Therefore, a [4] method has been proposed to prohibit the misuse of stolen cookies so that it is ineffective in stealing them through an XXS attack. This proposed method uses a one-time password and challenge-response authentication to identify whether a person is a valid owner of the cookie or not. In this document, we propose a secure cookie protocol that avoids the abuse of cookies stolen by XSS.

In [38] they have focused on XSS attacks by using tools to test the software and regular expressions to prove that websites are vulnerable, giving the developers the guidelines to deal with the flaws. Its objective is to offer the design of an application so that users can use it without suffering attacks. Based on their study, regular expressions can protect multiple domains and avoid XSS attacks, this at the development level, can protect the encoding to validate the input strings. The syntax analysis of the input string is important and must be correlated with the syntax of the regular expression provided by the developer of the application.

Another way to detect XSS attacks is presented in [39] using an intrusion detection system (IDS). This method captures all the potentially dangerous executable content on the client side and mixes it with the original content, so the page could be reprocessed at a future visit to compare if the content is repeated or there is a difference in the hashes indicating a possible XSS attack. According to the author, it indicates that this technique is useful for web forums and other sites where the user controls the content.

Another method to detect XSS attacks is presented by filtering patterns as in [40], thus filtering the user's entries to protect any web application. According to its authors, it is necessary to filter the entrances and exits to achieve an adequate level of protection, however, its solution is not scalable, and it does not offer the guarantees to function in the future, this is because the attacks become more sophisticated. As time progresses.

In [41] an approach has been proposed that uses an attack pattern model to generate and execute test cases. There is talk of the implementation of attack tests using XSS patterns to generate case studies. This is defined by programming XSS scripts emulating state machines with Modeling Unified Modeling Language (UML). This is considered as an integration of test methodologies in the software development cycle.

The detection task becomes more difficult with new attack patterns. In[42] they have focused on a detection mechanism for XSS attacks using rule-based filters using extensions in browsers. In addition, a model for estimating patterns has been presented in order to evaluate whether the intercepted requests have malicious attempts or not.

Another detection mechanism is presented in [43] through the use of improved classifiers and n-grams. This approach is oriented to the detection of XSS attacks in Social Networks Online (OSN). In summary, first a group of characteristics is identified in the web pages, secondly, an improved n-gram model is presented from these characteristics to classify the web pages, and finally, classifiers are combined with their improved model for the detection of XSS. A method has also been proposed with which to simulate and obtain more precise experimental data.

An approach to improving what is proposed in [43] uses automatic learning to detect XSS attacks on OSN [44]. This is because OSN has become the favorite target of the attackers. Their advantage is that they use an initial test database. The algorithms they use in their proposal were ADTree and AdaBoost.

In this document [45] we have investigated the recognition and detection of XSS attacks using regular expression pattern matching and a pre processing method. According to the authors, the Snort software is a security system effective enough to detect and recognize the payload of XSS attacks. This work has shown how to investigate the pattern or behavior of XSS attacks.

It has also been proposed a similar method to [46] where the weakness of the entries or their absence is validated. This is a static analysis method whose objective is to find XSS vulnerabilities. As the authors indicate, the correct input validation is difficult in large part because there are many ways to invoke the JavaScript interpreter, its method statically checks vulnerabilities and confronts it with the formalization of policies based on the World Wide Web Consortium (W3C) , the source code of the Firefox browser and tutorials for closed-code browsers. Its source of analysis is the flow of contaminated information combined with chain analysis (they take into account the semantics of entry validation routines).

The integrated mechanisms for web browsers called XSS

Jilters are considered as a means of rudimentary protection. The authors of [47] have taken advantage of poorly written PHP code to bypass jilter attacks specifically from the WebKit engine called XSS Auditor. Through two attacks, a first called PHP Array injection, and a second one that is a variant of the first. These two attacks take advantage of the incorrect administration of variables and matrices in the PHP code to bypass the XSS Auditor. In counterpart, the defense for the identified attacks is executed by means of code writing rules suitable for the developers and thus create secure web applications. Its objective is oriented to the mistakes made by PHP-based web application developers.

On the other hand, in [48] a generic and modular vulnerability scanner called ETSSDetector has been developed. This tool automatically analyzes web applications to find XSS vulnerabilities. Identifies and analyzes all the data entry points of the application and generates specific code injection tests for each one. This ensures that the correct entry of the input fields with valid information guarantees the effectiveness of the tests by increasing the XSS detection rate.

Another proposal is the feature extraction algorithms [10]. They are oriented to the source code of the internet applications. With these features several machine learning models are built to predict context sensitive XSS security vulnerabilities. This proposal has been implemented as a prototype to extract the characteristics of the PHP code. This is the basis for classifying a vulnerable source code file of a benign source code.

It could modify the characteristics of the client side as well as on the server, as shown in [49]. This method is called Buffer Based Cache Check. This technique detects and prevents XSS attacks. With this method the server stores a cache memory that contains a trusted validated instance of the last page that was presented. This way you can check inconsistencies against a new requested page. This could reduce the rendering time of browsers, as well as the number of queries within browsers. This method is oriented to mobile browsers. With this method you would only search for untrusted content because there is already a previous analysis of reliable content.

This study proposes a code audit approach to recover the defense model implemented in program source code and suggest guidelines to verify the adequacy of the recovered model against XSS [9] attacks. This approach extracts all the defenses implemented to guarantee every potentially vulnerable HTML output. It introduces a flowchart of contaminated information, as a model to audit the adequacy of the XSS defense codes.

Otra herramienta llamada XBuster [50] actua como una extension del navegador Mozilla Firefox, es decir es una defensa del lado del cliente. Esta herramienta divide y alamcena por separado cada parametro de solicitud HTTP (los convierte en contextos HTML y JavaScripttt) . Su mayor ventaja es que proteje contra todos los vectores de ataque XSS, incluida la inyeccion parcial de guiones, la inyeccion de atributos y la inyeccion de HTML. Ademas proteje contra el clickjacking.

This tool called XSS-me also acts as a security extension for the Mozilla Firefox browser. It is a proposal based on the codification of unfiltered reflections to detect vulnerable web applications that can be exploited using sophisticated attacks [51]. It is an integrated implementation that blocks the execution of malicious scripts specifically reflected XSS vulnerabilities. It is considered effective because it is integrated into the browser instead of its application as an extension. According to the authors, all browsers should include a XSS filter on the client side to help alleviate XSS vulnerabilities without patches.

A honeypot to learn about the pattern and behavior of the attacker is presented in [52]. It is a low interaction honeypot that emulates vulnerabilities that can be exploited through XSS. Its biggest advantage is that in addition to registering the activity of the attacker, he tries to expose his identity. According to the authors, your proposal could catch more useful information about the HTTP request than the famous web-based honeypot, Glastopf. In addition, with this honeypot you can detect the social networks accounts of the attackers by using the LikeJacking technique, however, this is not possible if the attackers used proxies or anonymizers. Its disadvantage is that some attackers would use techniques to hide their identity, so they could not be tracked.

This method uses automatic learning techniques such as Support Vector Machine (SVM) and Extreme Learning Machine (ELM) [53]. Its objective is to predict the type of malicious attack using the approaches of automatic learning, analyzing the prediction time. It focuses on the analysis of the content of the web pages, the URL, the redirection chain and also focuses on obfuscating the elements in a page or the scripts that are executed. Exploits search is also done to cover a wider class of malicious pages. Data is collected from user browsers, used to verify vulnerabilities. This approach has been implemented using the tools MATLAB and .NET, the same that is interactive and is called SpiderNet and is capable of detecting malicious web pages that evade the most advanced systems.

By evaluating the most used web browsers it has been shown that there is no adequate defense against XSS [7] attacks. This has been evaluated with browsers known as Internet Explorer 11, Google Chrome 32 and Mozilla Firefox. According to the authors, none of the three was able to defend against XSS attacks reflected. The experimental results show that this client-side solution can protect against a higher percentage of vulnerabilities than other browsers. It is testified that it is more propitious if this complement is integrated into the browser instead, it is applied as an extension.

A laboratory to simulate attacks is another proposal developed in [54]. In addition to attacks, the defense is designed through scripts. The goal is education and to motivate the understanding of the meaning of XSS vulnerabilities, how they occur, why they occur, how to exploit them and how to solve them.

This tool called PURITY [8] is aimed at testing web applications through test cases against certain sites. Its objective is to detect if the sites are vulnerable to these two threats (SQL and XSS injections). Through the emulation of a malicious activity and following typical sequences of these attacks to lead to a vulnerable state, with the advantage that is executed automatically and also manually. It differs from other tools because it is based on planning.

CSRFGuard [55] is a tool that runs on the Java EE platform to defend cross-application forgery (CSRF) attacks, but has some shortcomings: scripts must be inserted manually, dynamically created requests can not be handled from effective way and the defense can go through XSS attacks. To solve it, they added, dynamically, a token to avoid so that the defense would go through an XSS attack.

There are hybrid tools like [56]. It acts as a framework to detect input manipulation vulnerabilities (DIMV). This tool verifies the adequacy of the defenses in situations of ticket manipulation. For this they use the prediction of this vulnerability in a transparent way, that is, the cases that can not be proven are predicted through extracted signatures.

Another hybrid tool focuses on XSS attacks of the DOM type, which are serious vulnerabilities but little studied and appear in the extensions of web browsers [57]. This tool presents two phases of analsis. A static analysis to filter text and an abstract tree syntax analysis. In the second phase they use scripts as proof of concept to generate documents, this as a dynamic symbolic execution called DOM shadow. Through large-scale real-world experimentation, 58 XSS vulnerabilities originated by DOM previously unknown to the popular Greasemonkey browser extension have been discovered.

According to [58] the current preferred countermeasure for XSS attacks is the content security policies or CSP. It is a relatively simple method that aims to improve the level of communication security between people and devices through the Internet. This technique is aimed at protecting secure web services, such as those that provide vital information, web applications and Internet of Things (IoT) networks. This is fulfilled by the strict definition of the communication parts and the assets used in the web services. Simple and effective reports are a native part of the CSP design, which means that administrators can receive notifications about the execution of attacks almost instantly.

In [1] a linear automatic approach called XSS Chaser has been proposed that prevents web applications from XSS attacks. It is based on chain analysis to generate vulnerable patterns and thus prevent XSS attacks. These patterns are generated using forward and backward interpretation.

The intrusion detection system proposed in [59] is a container-based approach through a mapping model of query requests to recognize and prevent XSS attacks. This approach is used to identify two different customer requests. The impact measurement is calculated using the HTTP load and auto bench tool. On the other hand, performance measurement is done through various parameters, such as average page time, pages per second, memory and processing time.

Another method proposes attacks on systems in the form of web requests [60], through the use of tags such as ¡script¿, ¡iframe¿, etc. whose objective is to attack the client's web browser in the form of XSS or an SQL query. Clients access the application through a web server offering a web service to each one and separately. The client will send a web request using the user interface of the web application, based on the web request, a query will be generated in the database and the data will be recovered to the client in particular.

Two types of attacks called coding of N and binary alphabets have been analyzed in [61]. In addition, a method of dynamic access control is presented to prevent them by means of the existing technologies of detection and prevention of XSS attacks.

Another framework called ZEND [14] focuses on the problems surrounding XSS attacks. It acts as a simple and effective security model to protect websites. This model is based on a sequence of levels and is created by the combination of many tools. The implementation of the proposed model combines the Zend Framework web application and the HTML Purifier library. Zend Framework (ZF) is an open source framework for developing web applications and services with PHP. HTML Purifier is a standards compliant HTML filter library written in PHP and relatively simple to use. It removes malicious codes that result in XSS attacks.

There is a verification method for the defense against XSS attacks. Errors are found in websites of the e-commerce type. The behavior model of the website is stored in the form of an XML file. In [62] an automatic modeling algorithm for the HTML code of these websites is presented. A simple HTML code is modeled through the algorithm of automatic modeling for proposed HTML code, and the result is the behavior model of the website, stored in the XML file.

In [2] they also focus on the security vulnerabilities resulting from the generic entry validation issues that cause XSS attacks. The proposed detection method identifies a malicious execution sequence based on the initialized list of legitimate execution sequences and malicious malicious or literal strings generated during a training phase. The initialized lists are stored in four different web application execution profiles corresponding to four different attack scenarios. The detection module looks for the sequence of execution time.

The work proposed in [63] suggests a scheme for a system that can detect XSS attacks using an intrusion detection system (IDS). Signatures are used to detect these attacks. To test the usefulness and effectiveness of the proposed work, a proof-of-concept prototype was implemented using SNORT IDS. It has been proposed to use rules to identify the XSS attack by monitoring incoming and outgoing packets that coincide or not with the defined rules.

Do not forget the high rates of false negatives and false positives, for example for the proposal of [64] on fuzzing black-box and static analysis. For the dynamic analysis, operation and complexity costs are required, however the results are more efficient. This proposal is based on a dynamic detection framework (TT-XSS) for the DOM-XSS attack type through the analysis on the client side. Here they rewrite all JavaScripttt functions and DOM APIs to contaminate the rendering process of browsers.

In [17] the use of unique cookies is proposed as a more robust alternative for the authentication of sessions. This prevents attacks such as session hijacking by signing each user request with a session secret stored securely in the browser. Its implementation acts as a complement to the popular WordPress platform and as an extension for Firefox for both PCs and mobile browsers.

In [65] the first evaluation is carried out based on a set of cookies that have been compiled from 70 popular websites obtained from the Alexa [11] ranking. The data obtained designed a semiautomatic procedure that is based on a new authentication token notion to capture multiple web authentication schemes. Then, by means of a detection method based on supervised learning, it has been used to train a binary classifier.

Currently there are also checklists for web page developers to check that the cookie-based authentication mechanism is implemented securely [66]. To this end, a tool called Newton has been developed that helps programmers to identify authentication cookies for specific parts of the website, so it can be verified, through this list, that these cookies have been implemented in a secure manner.

Another evaluation of collected cookies is presented in [67]. It is based on the proposal of a hypothesis for 2,464 cookies collected from 215 popular websites of the Alexa ranking. It has been proposed the design of a semi-automatic procedure based on authentication tokens to capture multiple web authentication schemes. They have also proposed a detection method based on supervised learning, where the hypothesis is used to form a set of binary classifiers.

Another robust framework is proposed in [68] called XSS-SAFE by Cross-Site Scripting SecureWeb Application FramEwork. It is an automated framework on the server side that detects and mitigates XSS attacks. Its operation is through the injection of JavaScript disinfection codes in the same source code, thereby mitigating the attack vectors.

In the same way, another framework is proposed to detect and alleviate the propagation of XSS [69] worms from multimedia web applications based on online social networks (OSN) for cloud environments. It is based on two modes of operation, the first one disinfects the JavaScript variables that are extracted and that are not reliable, they have called it a training mode, it is trained by storing these codes in a repository. The second mode is known as detection, which compares the disinfected HTTP responses that are generated on the OSN web server with the disinfected responses that were stored in the repository. If there are variations, it means that XSS worms were injected from the OSN servers.

### A. Other methods of defense

***Patches for web servers:*** The advantage of the servers that are configured to raise web services is that they provide mechanisms for the prevention of XSS attacks through modules at the level of browsers. In other words, they offer solutions at the level of the developer user. However, the problem is that they only focus on client-side attacks instead of server-level scripting. Another defect in the patch is that they are designed to work on separate systems.

***XSS (Cross Site Scripting) Prevention Cheat Sheet***: There is a rule definition  cite Online11 that can be used to avoid XSS attacks in web applications. It is not necessary to implement them all. Many organizations can verify that the implementation of only some rules are sufficient to cover their needs.

Rule 0: The first rule is to deny all.
Rule 1: To place untrusted data directly in the HTML body.
Rule 2: To put untrusted data into typical attribute values such as width, name, value, etc.
Rule 3: It refers to the dynamically generated JavaScript code. The only safe place to put untrusted data in this code is within a ”data value” cited.
Rule 4: To place untrusted data in a style sheet or style label. CSS is powerful and can be used for numerous attacks.
Rule 5: To put untrusted data on the value of the HTTP GET parameter.
Rule 6: If the application handles marked, an entry that is not trusted and that is supposed to contain HTML, can be very difficult to validate.

***Content Security Policy (CSP)***: The Content Security Policy is known as a security standard introduced to prevent attacks XSS [70], clickjacking attacks and other types of injection attacks that are the result of the execution of malicious content within the content of the page Web. It is a recommendation of the W3C [71] working group for Web Application Security backed by modern web browsers. With this method, developers must declare the approved origins of the content that browsers will upload to their website, for example, JavaScript, CSS, HTML frames, fonts, images, embeddable objects such as Java, ActiveX, audio and video files, and other HTML5 features.

***Input text validation***: This method is a more common type to defend against XSS attacks. This processing, of text entries that are not reliable, is done through the programming of modules to filter and analyze text.

***Libraries or Frameworks***: These are used to make XSS vulnerabilities easier to avoid. Some examples include Microsoft’s Anti-XSS [72] library, a free and open collection of all the security methods a developer needs to build a secure Web application called OWASP ESAPI [73] and Apache Wicket [74]. The idea is to apply the correct coding of all alphanumeric characters for when data of any type of input is received.

***XSSer***: Or called also Cross Site ”Scripter” is an automated framework for detecting, exploiting and reporting XSS vulnerabilities in web applications. It contains several options to try to bypass certain filters and several special techniques of code injection. It is a kind of open source project XSS developed by OWASP; is a testing tool that can automatically perform the detection process and launch the feats of XSS injections on any website.

## VI. Discussion

According to the classification proposed in [75] there are two analysis approaches:

- Static Analysis
- Dynamic Analysis

In the same way in[76] a combination between the static and dynamic approach called *hybrid* has been proposed. On the other hand, we found 3 forms of prevention analysis of this type of attacks:

- On the client’s side

- On the server side
- Hybrid (client-server combination)

Most of the proposals aimed at the DOM Cross Site Scripting detection method (DOM-XSS) have been divided into three types: black box fuzzing, static analysis and dynamic analysis [64].

Other defense methods found in [9] are based on the analysis of code extraction:

- Through input validation
- Through escape characters
- Through filters
- Through the set of characters that is specified

To complement this classification, the analysis of tables II, III, IV and V has been presented. The information has been structured according to the interest of our research, which in this case would be to analyze all the tools and methods proposed or used to mitigate attacks XSS. In Fig. 12 we propose a mental map that is summarized below:

TABLE II. SUMMARY OF TOOLS AND METHODS FOUND TO MITIGATE XSS ATTACKS

| **Tool/Method** | **Description** | **Reference** |
|---|---|---|
| Laboratories | Activity Emulation, Simulation, Honeypots | [50][53][54] |
| IDS | Snort, Containers | [43][60][65] |
| Content Analysis | Executable Content, Rule Filters, Cache Test, Text Filter, Content Security Policy, String and URL Analysis, XML, Input Validation, Input Strings, Absence and Weakness, VB Script | [12][37][40] [47][57][58] [61][63][22] [36][44][64] [59] |
| Web Toolkit | PHP, Browsers Complements, Browsers without headers | [45][48][49] [52][24] |
| Pattern Analysis | Using C Software, Filtering Patterns, Modeling Attacks | [32][38][39] |
| JAVA | JAVA EE, Javamail Development Toolkit | [28][35] |
| Web Scanner | Distributed, Normal, Black Box | [21][23][29] [46] |
| Defensive Programming | – | [33] |
| Concolic Test | – | [27] |
| Micro Benchmark | – | [30] |
| Proxy | Coding of Alphabets | [34][62] |
| Cookies Analysis | Session authentication, Checklists | [4][15][68] |
| Using Tokens | Session | [55] |

In the same way, in Fig. 13 we propose a mental map detailing the methods and tools that are based on artificial intelligence to stop or detect XSS attacks. Table III summarizes the information found:

From the analysis of the proposals found, we propose the following enumeration of methods used for the detection and blocking of XSS attacks using artificial intelligence:

- Evaluate and Marking Cookies [69][67]

TABLE III. SUMMARY OF TOOLS AND METHODS FOUND TO MITIGATE XSS ATTACKS USING ARTIFICIAL INTELLIGENCE

| **Tool/Method** | **Description** | **Reference** |
|---|---|---|
| Ground Truth Cookies | Binary Classifier | [69] |
| Attack Prediction | Support Vector Machine, Extreme Vector Machine | [51] |
| Code Audit | Pattern Matching | [5] |
| GIT Repository | SVM, NB, Bagging, JRIP, J48 | [7] |
| Classifiers and n-gramas | AdTree, AdBoost | [41][42] |
| Text Mining | Random Tree, J48, JRIP, NB, Bagging | [31] |
| Authentication Token | Supervised Learning, Binary classifier | [67] |
| Mining of Attributes | Data Mining | [56] |
| Repository of Attacks Vetors | Automate Attacks | [25] |
| Fuzzing Evolutionary | Genetics Algorythms | [26] |

- Predict and Automate Attacks [51][25]
- Code Audit [5]
- Predict and Detect vulnerabilities [7][31]
- Web or Software classifier [41][42][56]
- Generate test cases [26]

In summary, a large number of proposals have been found and structured so that they can be grouped considering a common goal, or are oriented to the same point of study. As shown in Fig. 14, there is a greater tendency in the proposals of traditional tools and methods, among which are:

- Content Analysis
- Input Validation
- Web Scanner
- Web toolkits
- Pattern Analysis

Similarly, an analysis is presented in Tables IV - VII of the strengths and weaknesses of all the research works, and as a result a lower tendency in the use of artificial intelligence to detect or mitigate these XSS attacks was obtained. In Fig. 14 a smaller number of proposals is shown, among which stand out:

- Evaluate and Marking Cookies
- Predict and Automate Attacks
- Code Audit
- Predict and Detect Attacks
- Web or Software Classifier
- Generate test cases

With these two analyzes the objective of our proposal has been structured, with which we will analyze the preferences of the users by mining their cookies to avoid XSS attacks.

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | E/D/H | IA |
|---|---|---|---|---|---|---|---|---|---|
| [23] | Qualys Guard | Escaner Web | Detect intrusions, anomalies based on learning and attacks against applications and the web server | Web Application and Servers | H | It is based on a system that qualifies URLs as safe or not, allowing their execution or not in the system | It is not 100% functional since the client must do a manual scan of the URLs that he needs to visit | E/D | – |
| [24] | – | Entry validation | Automatically extract the encoding functions used in a web application to disinfect untrusted entries | Web Application | C | SIt guides developers to detect and correct problems without relying on security experts | It only targets 0 Day XSS attacks | E | – |
| [25] | Acunetix WVS, Rational AppScan Enterprise, ZAP | Black box web scanner | Develop a custom test bench to compare the capabilities of 3 black box scanners | XSS Stored | C | Allows the use of scan profiles to specify to which category of vulnerabilities the attack should be directed | Relatively poor performance in detecting only stored XSS | – | – |
| [26] | Ghost.py | Browser without header | Interpret JavaScript and load Ajax content simulating browser behavior to obtain hidden injection points | Web Application | C | More accessible and portable system for secondary development | Use a framework to attack pages with pre-established vulnerabilities | D | – |
| [27] | – | Optimized repertoire of attack vectors | Automate attacks and dynamically detect XSS vulnerabilities in applications | Web Application | C | Generate XSS attack vectors automatically | It has only been tested on 24 websites | D | Machine Learning |
| [28] | Kameleon Fuzz | Combination of evolutionary fuzzing with inference models | Generate entries through genetic algorithms through a formal learned model | XSS type 1 | C | Generates test cases through the automated search of XSS type 1 | It has not been experienced in real-world applications | E | Genetic algorithms |
| [29] | – | Technique based on concolic tests | Translate web applications into another language using a concolic tool to identify the input and output variables that are used | JSP Web Application | C | It is able to identify the entries that occur in the form of malicious strings | It is only oriented to pages of type JSP | – | – |
| [30] | – | JavaMail Development Kit | Build a repository of XSS test vectors to implement a dynamic tool | Webmail systems | C | 14 XSS attack vectors related to HTML5 have been identified | The filtering mechanism does not pay enough attention to the new tag called (embed) in HTML5 | D | – |
| [31] | Web Input Vector Extractor Teaser (WIVET) | Distributed scanning tool | Provide a database of attack vectors to search and correct vulnerabilities within websites | Modern Web Applications, DOM (XSS, Open Redirection, Clobbering) | S | It is functional to analyze data on a large scale | Only 1000 of the best websites in the Alexa ranking were analyzed | – | – |
| [32] | Dom XSS Micro | Micro Benchmark | Extract representative vulnerabilities to create templates | XSS DOM | S | Acts as a basis for further development and discussion in the community | Currently the tool is not complete | D | – |
| [77] | Repositorio GIT | Text mining | Detect code files vulnerable to XSS | Source code of PHP web applications | S | It does not depend on the programming language version and works with object-oriented code | Source code tokens are not enough as a feature to develop machine learning models | D | NB, Bagging, Random Tree, J48, JRip. |
| [34] | XSSDM | Analysis of patterns by C | Act as a prototype to evaluate a set of public data | PHP | C | Minimizes false positive and false negative type results | It has only focused on web applications developed in PHP | E | – |

TABLE IV. ANALYSIS AND DISCUSSION OF THE TOOLS AND METHODS FOUND. (1/5)

TABLE V. ANALYSIS AND DISCUSSION OF THE TOOLS AND METHODS FOUND (2/5)

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantage | Limiting | E/D/H | IA |
|---|---|---|---|---|---|---|---|---|---|
| [35] | RIPS, PIXY | Detailed analysis through defensive programming | Precisely detect XSS vulnerabilities based on sensitive context vs. HTML context | PHP | – | Minimize false positive and false negative type results | Only one empirical evaluation has been presented | E | – |
| [36] | Fiddler | Development at the proxy level based on Kullback-Leibler Divergence (KLD) | Analyze the distance between the probability distribution of the legitimate JavaScript code and the JavaScript code present in a response page | PHP | C | Can detect most known XSS attack signatures and display a very low false positive warning | They only apply the approach in three vulnerable PHP applications | – | – |
| [37] | JAVA, SOA, DAO , MVC, Facade y DAO | Validate the prototype at the design, coding and security level | Use design specifications such as DAO, Facade and WS-Security framework (MVC) as architectural pattern | J2EE | – | Demonstrate that the use of standards, techniques and recommendations are necessary to avoid XSS vulnerabilities | The SOAP web service description document (WSDL) is extensive | E | – |
| [4] | – | Analysis of cookies | One-time password and authentication to identify if a person is a valid owner of the cookie | Avoid theft of cookies | C | It is a secure protocol that avoids the abuse of cookies stolen by XSS | It does not work if the attack is made before the cookies expire, it also has latency | – | – |
| [38] | – | Validate the input strings | Check that the websites are vulnerable, giving the guidelines to the developers | Software and regular expressions | C | Regular expressions protect multiple domains and prevent XSS attacks | Only 50 popular domains have been analyzed | – | – |
| [39] | Pearl based IDS | Capture the executable content | Mix the executable code with the content and obtain a Hash for a later visit | Web Application | C | Useful for web forums and other sites where the user controls the content | The tests are carried out with web pages created by the authors | E | – |
| [40] | – | Filtering patterns | Filter user entries to protect any web application | Web Application | C | An adequate level of protection has been achieved | It is not scalable, it does not offer the guarantees to function in the future, this is because the attacks become more sophisticated as time progresses | E | – |
| [41] | – | Attack pattern model | Implement attack tests using XSS patterns to generate case studies | UML | C | Test methodologies are integrated into the software development cycle | Only use 2 web applications for type 1 and type 2 attacks | E | – |
| [42] | XSSERC | Filter based on rules | Estimate patterns to evaluate whether intercepted requests have malicious attempts or not | Extensions in browsers, CORJACKING, HTML5 | C | Intercept web requests and evaluate them to estimate patterns | They only test the performance of the system for the 50 most popular popular websites in the world | E | – |
| [43] | DMOZ, XSSed DB | Classifiers and improved n-grams | Sort web pages by identifying their characteristics with an improved n-gram model | Online Social Networks (OSN) | C | They can simulate and obtain more precise experimental data | The ways of combining the improved n-gram model and the classifiers affected the detection result | E | AdTree |
| [44] | DMOZ, XSSed database, WEKA | Classifiers and improved n-grams | Improve what is proposed in [43] | Online social Networks (OSN). | C | An initial test database has been used | – | E | AdTree, AdaBoost |

TABLE VI. Analysis and discussion of the tools and methods found (3/5)

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | E/D/H | IA |
|---|---|---|---|---|---|---|---|---|---|
| [45] | Snort | Coincidence of regular expression patterns | Show how to investigate the pattern or behavior of XSS attacks | Persistent and non-persistent XSS | H | It is enough as a first defense barrier | Many false positives, use of resources, does not perform prevention or deterrence | E | – |
| [46] | – | Validation of the absence and weakness of entries | Statically check the vulnerabilities and face the formalization of policies based on the World Wide Web Consortium (W3C) and the source code of the Firefox browser | Javascript interpreter | C | Its source is the flow of contaminated information combined with string analysis (semantics of input validation routines) | Only oriented to Firefox, large number of lines programmed in O'neil | Practical Static | – |
| [47] | WebKit XSS Auditor | PHP Array Injection, PHP Array-like Injection | Take advantage of the incorrect administration of variables and matrices in the PHP code to bypass the XSS Auditor | Web applications based on PHP | C | Focus on the mistakes that PHP-based web application developers make | They only present concrete examples of poorly written PHP code | E | – |
| [48] | ETSS Detector | Web Scanner | Automatically analyze web applications to find XSS vulnerabilities | Web Application | C/S | Ensures the correct entry of the entry fields | Can only detect persistent and non-persistent XSS | – | – |
| [10] | GIT Repository | Feature extraction algorithms | Build several machine learning models to predict context sensitive XSS security vulnerabilities | PHP | C | Sort and separate a vulnerable source file from a benign source code | Unable to extract vulnerability prediction characteristics | E | SVM, NB, Bagging, J48, and JRip |
| [49] | – | Buffer-based cache test | Store a cache that contains a validated instance of trust from the last page that was submitted | Mobile Browsers | C/S | It has reduced the rendering time of browsers, as well as the number of queries within browsers | Oriented only for mobile browsers | E | – |
| [9] | XSSDE, SOOT | Program analyser, defence feature miner (DF miner) and TIFG | Propose a code audit approach to recover the defense model implemented in source code | HTML | S | A variant of flowcharts has been proposed | It is not oriented to DOM XSS | – | Pattern Matching |
| [50] | Xbuster | Firefox extension | Split and store each HTTP request parameter separately (convert them into HTML and JavaScript contexts) | Firefox browser | C | Protects against all attack vectors XSS, protects against clickjacking | There is a delicate balance between the false positive rate and the false negative rate | – | – |
| [51] | XSS-ME | Firefox extension | Block the execution of malicious scripts specifically XSS vulnerabilities reflected | XSS reflected | C | Condensing the false negative rate | Only oriented to mozilla firefox | E | – |
| [52] | JSON dictionary, JavaScript | Honeypot con LikeJacking | Emulate vulnerabilities through a low interaction honeypot to be exploited using XSS | HTTP requests | – | It not only records the attacker's activity but also tries to expose his identity | Some attackers use techniques to hide their identity, so they can not be tracked | – | – |
| [53] | SpiderNet | Machine learning techniques, MATLAB y .NET, | Predicting the type of malicious attack used in machine learning approaches, the prediction time has been considered | DOM XSS | C | Structured dataset of a set of pages in real time | – | – | Support Vector Machine (SVM) and Extreme Learning Machine (ELM) |

TABLE VII. Analysis and discussion of the tools and methods found (4/5)

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | E/D/H | IA |
|---|---|---|---|---|---|---|---|---|---|
| [7] | XSS-me (Firefox), XSS Auditor (Chrome) | Browser add-ons | Evaluate the most commonly used web browsers to demonstrate that there is no adequate defense against XSS attacks | Reflected XSS, Internet Explorer 11, Google Chrome 32 and Mozilla Firefox | C | It is more appropriate if the plug-in is integrated within the browser, that is, it is applied as an extension | Only the comparison has been made with 3 browsers (Chrome, Firefox and Explorer) | – | – |
| [54] | – | Laboratory to simulate attacks | Educate and motivate the understanding of the meaning of XSS vulnerabilities | End users | – | Students can better understand XSS vulnerabilities, how they occur, how to exploit them and how to solve them | The proposal is extremely simple and ignores many technical details as it only includes essential source codes for XSS attacks | – | – |
| [8] | PURITY | Malicious activity emulation | Detect if sites are vulnerable to SQL and XSS injection threats | Web Application | C | It is executed automatically and also manually. It differs from other tools because it is based on planning | Allows little manual intervention, the program chooses the partially automated mode. Only one case study has been evaluated | – | – |
| [55] | CSRFGuard | Session Tokens | Filter and intercept customer requests, verify if a token exists or not and match the reserved token of the current session | Java EE | E | Dynamic addition of Token through event triggering can prevent the use of XSS to bypass CSRFGuard. | Scripts must be inserted manually. CSRFGuard can be skipped if attackers make use of XSS vulnerabilities | D | – |
| [56] | DIMVC | Mining of code attributes to predict vulnerabilities | Classify SW as ”safe” or ”unsafe” | Software | H | The cases that can not be proven are predicted through extracted signatures | Mining is applied only to the relevant code that is affected by the data entry (input validation) | H | Data mining |
| [57] | Grease Monkey | Filter text and analyze syntax, use scripts as proof of concept to generate documents | A text filter written in Python. A high performance analyzer written in JavaScript. The DOM symbolic execution based on Jalangi3. An automaton written by Java that is used to encode regular expressions. A navigation simulation library. | XSS DOM, extensions of web browsers | H | The waiting time is relatively low | Replacement of strings that are not compatible, solved manually, the proposal could omit some vulnerable scripts due to the lack of compatibility with DOM manipulation operations | H | – |
| [58] | – | Content security policies or CSP | Improve the level of communication security between people and devices through the Internet | Secure web services, IoT | H | It is recommended for environments in which the parties that communicate (people and devices) | No XSS attacks have been spread on IoT devices | – | – |
| [1] | XSS Chaser | Analysis of chains to generate vulnerable patterns. Non Deterministic Turing Machine | Create vulnerable patterns to avoid XSS. These patterns are generated using text string analysis | Web Application | C | XSS Chaser finds attack patterns in less time than existing algorithms | Analyze only with samples of attack patterns | E | – |

TABLE VIII. ANALYSIS AND DISCUSSION OF THE TOOLS AND METHODS FOUND (5/5)

| Ref. | Tool | Action | Objective | Oriented | C/S/H | Advantages | Limiting | E/D/H | IA |
|---|---|---|---|---|---|---|---|---|---|
| [59] | IDS | Containers through a mapping model of query requests | Each client has its own container that limits the damage caused by any attacker to that container only | Web Application | S | Use containers to limit XSS attacks | The speed of the network connection affects the final result, the user must save the URLs before executing the tool in a text file | E/D | – |
| [60] | .Net | Attacks on systems in the form of web requests | Offer a web service for each client, a query is generated in the database and the data will be recovered to the client in particular. | Web Application | S | Minimize the detection rate of false positives and strengthen the disinfection of the inputs | It's just a proposed study | E/D | – |
| [61] | – | Coding of N and binary alphabets | Dynamic access control method to detect and prevent XSS attacks. | Web Application | C | An improved web proxy has been proposed to avoid XSS attacks based on advanced coding | Not optimal for multidomain XSS attacks | D | – |
| [14] | Zend Framework, HTML Purifier | Combination of two models | This model is based on a sequence of levels and is created by combining many tools | Web Application | C | Provide protection to the URL by hiding the original names of web pages and their design language | – | E/D | – |
| [2] | VB Script | Malicious execution sequences Identification | Identify sequences based on official lists of legitimate execution sequences and malicious chains generated during a training phase. | Input validation | C/S | The proposal works under the content-based approach. It is capable of detecting all categories of XSS attacks | The WAEP automatic update using the log report analysis is a challenging task | E | – |
| [62] | – | Automatic modeling algorithm | Store the behavior model of the website in the form of an XML file | Web application of e-commerce type | C | The behavior of an operation is judged if it complies with the requirements of the legal behavior of the website, in order to avoid XSS attacks from the point of operation | It is only oriented to e-comerce pages | E/D | – |
| [63] | Snort | Signatures to detect XSS attacks | Use rules to identify XSS attacks by monitoring incoming and outgoing packets that match or not match defined rules | Analysis of packages | C/S | The experiments have been carried out in a real network environment | Many resources are required for large-scale package analysis | E | – |
| [17] | Wordpress, BuddyPress, OTC browser, Fennec | Session Authentication | Use unique cookies as a more robust alternative to prevent attacks such as session hijacking | Web Application (PC and mobile) | C | The proposal is an experimental evaluation to characterize and compare the performance of authentication cookies | Some cookies use symmetric encryption to protect session information of sensitive users. | E | – |
| [65] | Weka | Authentication Tokens | Evaluate a set of cookies collected from 70 popular websites obtained from the Alexa ranking | Web Application | C | The proposal allows storing the user name and password information using tokens | Only protect size 2 authentication tokens | – | Supervised learning, binary classifier |
| [66] | Newton | Checklists | Help programmers to identify authentication cookies, in order to verify, through these lists, that these cookies have been implemented safely | Developers | S | It is oriented to the developer user | Many of the sites that were taken during deployment were not previously tested in the lab, excessive server load | – | – |
| [67] | Ground truth of cookies | Collect cookies from a group of websites and mark each cookie with a binary identifier | Evaluate the effectiveness of the techniques of identification and detection of cookies and check if they offer a good degree of protection | Web authentication schemes | C | The proposal of supervised learning is very precise, achieving a good balance between security and usability | The proposal is limited since the client's authentication is based on complex uses, often difficult to predict. | – | Binary Classifiers |

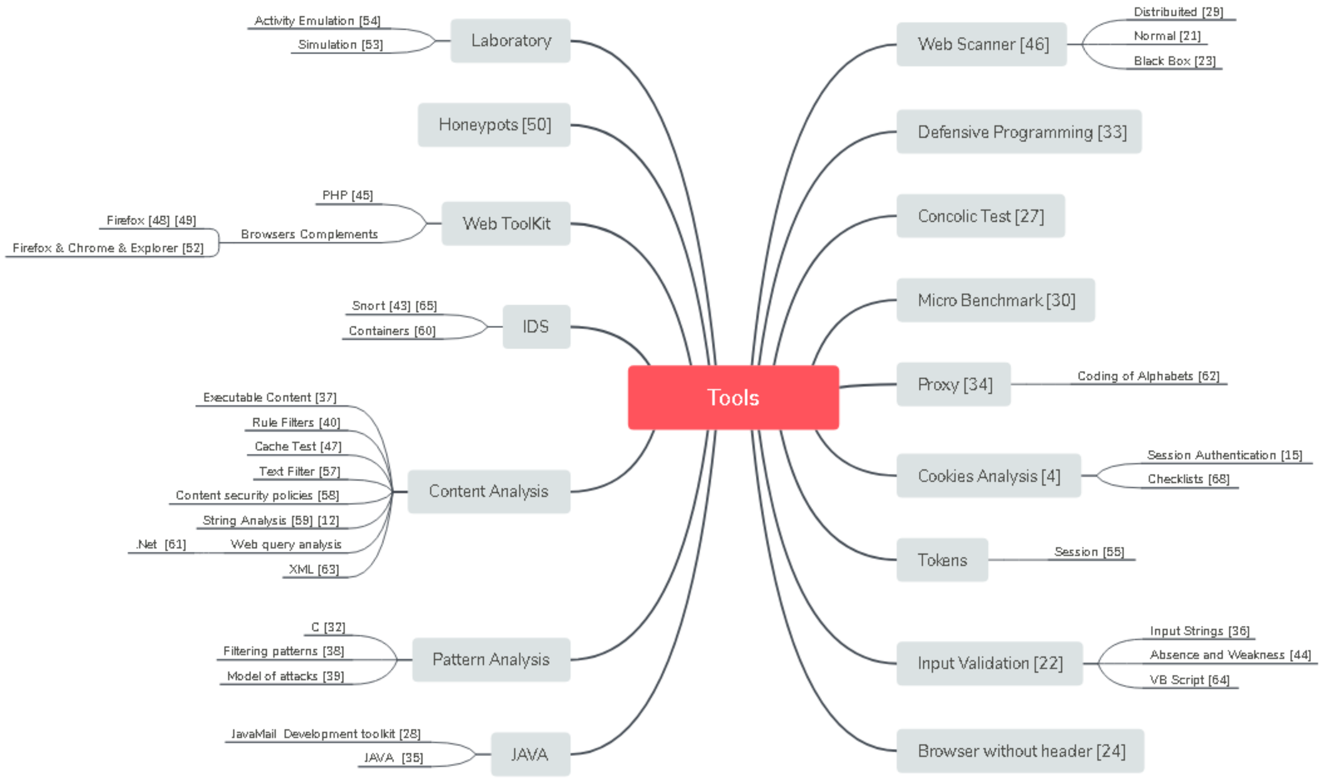


Fig. 12. Summary of tools proposed and used to mitigate XSS attacks

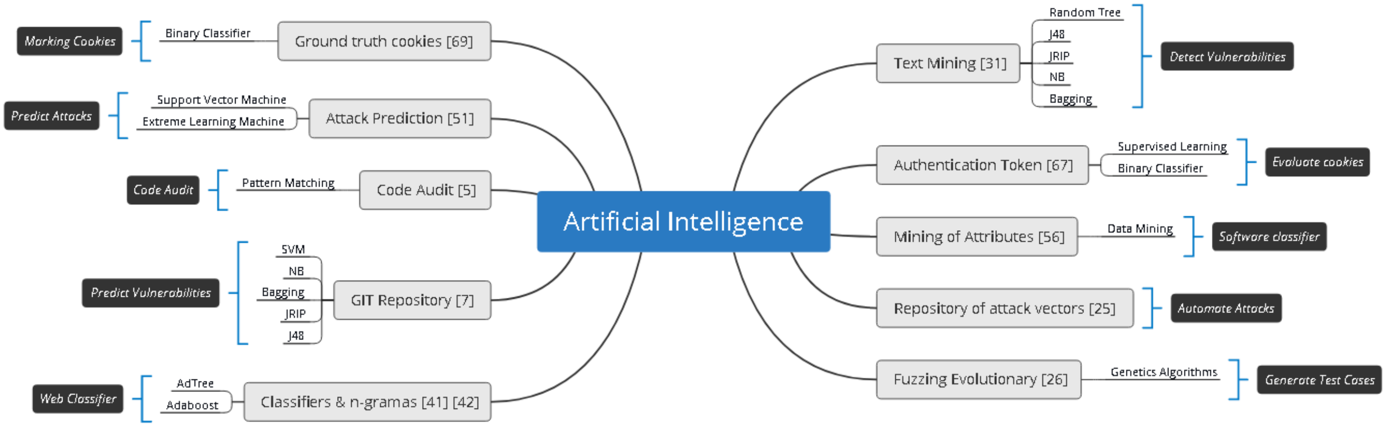


Fig. 13. Proposals that use some method of Artificial Intelligence to mitigate XSS attacks

## VII. Conclusions and Future Work

After the analysis of all the approaches we found a growth in the trend of content analysis (9 documents) and patterns (3 documents) as methods to detect and mitigate XSS type attacks, in addition we observed a low tendency in the use of artificial intelligence (11 documents).

This shows a clear result that most proposals lean towards analyzing the content of web pages to find patterns that allow identifying if their programming contains XSS type scripts. As a result of this, the executable content, the text filtering rules, the string analysis, the web query analysis and the web cache analysis, are some of the proposals that we have found among the most common to mitigate attacks of type XSS.

So we have also found few proposals aimed at analyzing the cookies that are created when a user visits a web page, as shown in our analysis only 1 document makes cookies to assign an identifier to be processed to avoid these attacks In this context, our research proposal will focus on the analysis of user behavior by studying their cookies, to search and obtain patterns in all the cookies that are stored in their equipment

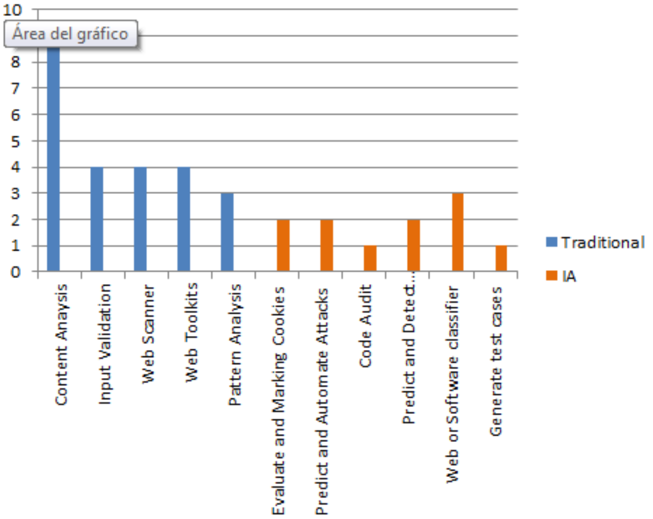


Fig. 14. Trends in the proposal of methods or tools to mitigate XSS attacks

and/or devices and thus generate a framework that allows to evaluate the behavior and needs of the users and offer a self-training system to prevent attacks of type XSS and other complementary.

## References

[1] D. A. Suju and G. M. Gandhi, "An automaton based approach for forestalling cross site scripting attacks in web application," in *2015 Seventh International Conference on Advanced Computing (ICoAC)*, Dec. 2015, pp. 1–6.

[2] D. Das, U. Sharma, and D. K. Bhattacharyya, "Detection of cross-site scripting attack under multiple scenarios," *The Computer Journal*, vol. 58, no. 4, pp. 808–822, Apr. 2015.

[3] IT-Digital. (Apr. 2018). El 100% de las aplicaciones web contienen vulnerabilidades, [Online]. Available: http://discoverthenew.ituser.es/security-and-risk-management/2018/04/el-100-de-las-aplicaciones-web-contienen-vulnerabilidades.

[4] H. Takahashi, K. Yasunaga, M. Mambo, K. Kim, and H. Y. Youm, "Preventing abuse of cookies stolen by xss," in *2013 Eighth Asia Joint Conference on Information Security*, Jul. 2013, pp. 85–89.

[5] Imperva. (Apr. 2018). The state of web application vulnerabilities in 2017, [Online]. Available: https://www.imperva.com/blog/2017/12/the-state-of-web-application-vulnerabilities-in-2017/.

[6] PandaSecurity. (Apr. 2018). Equifax no fue un caso aislado: El peligro de las web apps, [Online]. Available: https://www.pandasecurity.com/spain/mediacenter/seguridad/equifax-y-peligro-de-las-web-apps/.

[7] J. Shanmugam and M. Ponnavaikko, "Xss application worms: New internet infestation and optimized protective measures," in *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, vol. 3, Jul. 2007, pp. 1164–1169.

[8] J. Bozic and F. Wotawa, "Purity: A planning-based security testing tool," in *2015 IEEE International Conference on Software Quality, Reliability and Security - Companion*, Aug. 2015, pp. 46–55.

[9] L. K. Shar and H. B. K. Tan, "Auditing the xss defence features implemented in web application programs," *IET Software*, vol. 6, no. 4, pp. 377–390, Aug. 2012.

[10] M. K. Gupta, M. C. Govil, and G. Singh, "Predicting cross-site scripting (xss) security vulnerabilities in web applications," in *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Jul. 2015, pp. 162–167.

[11] Verizon. (Apr. 2018). 2017 data breach investigations report, [Online]. Available: http://www.ictsecuritymagazine.com/wp-content/uploads/2017-Data-Breach-Investigations-Report.pdf.

[12] SANS. (Apr. 2018). Cwe/sans top 25 most dangerous software errors, [Online]. Available: https://www.sans.org/top25-software-errors.

[13] CWE. (Apr. 2018). Cwe-79: Improper neutralization of input during web page generation ('cross-site scripting'), [Online]. Available: http://cwe.mitre.org/top25/index.html#CWE-79.

[14] Y. F. G. M. Elhakeem and B. I. A. Barry, "Developing a security model to protect websites from cross-site scripting attacks using zend framework application," in *2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING (ICCEEE)*, Aug. 2013, pp. 624–629.

[15] E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic, "Noxes: A client-side solution for mitigating cross-site scripting attacks," in *Proceedings of the 2006 ACM symposium on Applied computing*, ACM, 2006, pp. 330–337.

[16] G. E. Rodríguez, D. E. Benavides, J. Torres, P. Flores, and W. Fuertes, "Cookie scout: An analytic model for prevention of cross-site scripting (xss) using a cookie classifier," in *Proceedings of the International Conference on Information Technology & Systems (ICITS 2018)*, Á. Rocha and T. Guarda, Eds., Cham: Springer International Publishing, 2018, pp. 497–507.

[17] I. Dacosta, S. Chakradeo, M. Ahamad, and P. Traynor, "One-time cookies: Preventing session hijacking attacks with stateless authentication tokens," *ACM Trans. Internet Technol.*, vol. 12, no. 1, 1:1–1:24, Jul. 2012, ISSN: 1533-5399.

[18] GOOGLE. (Apr. 2018). Desarrolle sus habilidades analíticas, [Online]. Available: https://www.google.es/analytics/learn/.

[19] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten, "Cookies that give you away: The surveillance implications of web tracking," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15, Florence, Italy, 2015, pp. 289–299, ISBN: 978-1-4503-3469-3.

[20] Optanon. (Apr. 2018). The cookie law explained, [Online]. Available: https://www.cookielaw.org/the-cookie-law/.

[21] S. J. Murdoch, "Hardened stateless session cookies," in *Security Protocols XVI*, B. Christianson, J. A. Malcolm, V. Matyas, and M. Roe, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 93–101.

[22] WhiteHatSecurity. (Apr. 2018). 2017 application security statistics report, [Online]. Available: https://www.whitehatsec.com/resources-category/premium-content/web-application-stats-report-2017/.

[23] T. S. Mehta and S. Jamwal, "Model to prevent websites from xss vulnerabilities," *IJCSIT) International Journal of Computer Science and Information Technologies*, vol. 6, no. 2, pp. 1059–1067, 2015.

[24] M. Mohammadi, B. Chu, H. R. Lipford, and E. Murphy-Hill, "Automatic web security unit testing: Xss vulnerability detection," in *2016 IEEE/ACM 11th International Workshop in Automation of Software Test (AST)*, May 2016, pp. 78–84.

[25] M. Parvez, P. Zavarsky, and N. Khoury, "Analysis of effectiveness of black-box web application scanners in detection of stored sql injection and stored xss vulnerabilities," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec. 2015, pp. 186–191.

[26] Y. Liu, W. Zhao, D. Wang, and L. Fu, "A xss vulnerability detection approach based on simulating browser behavior," in *2015 2nd International Conference on Information Science and Security (ICISS)*, Dec. 2015, pp. 1–4.

[27] X. Guo, S. Jin, and Y. Zhang, "Xss vulnerability detection using optimized attack vector repertory," in *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Sep. 2015, pp. 29–36.

[28] F. Duchene, R. Groz, S. Rawat, and J. L. Richier, "Xss vulnerability detection using model inference assisted evolutionary fuzzing," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, Apr. 2012, pp. 815–817.

[29] M. E. Ruse and S. Basu, "Detecting cross-site scripting vulnerability using concolic testing," in *2013 10th International Conference on Information Technology: New Generations*, Apr. 2013, pp. 633–638.

[30] G. Dong, Y. Zhang, X. Wang, P. Wang, and L. Liu, "Detecting cross site scripting vulnerabilities introduced by html5," in *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, May 2014, pp. 319–323.

[31] T. K. Nguyen and S. O. Hwang, "Large-scale detection of dom-based xss based on publisher and subscriber model," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2016, pp. 975–980.

[32] J. Pan and X. Mao, "Domxssmicro: A micro benchmark for evaluating dom-based cross-site scripting detection," in *2016 IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 208–215.

[33] P. Chaudhary, B. B. Gupta, and S. Yamaguchi, “Xss detection with automatic view isolation on online social network,” in *2016 IEEE 5th Global Conference on Consumer Electronics*, Oct. 2016, pp. 1–5.

[34] M. K. Gupta, M. C. Govil, G. Singh, and P. Sharma, “Xssdm: Towards detection and mitigation of cross-site scripting vulnerabilities in web applications,” in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Aug. 2015, pp. 2010–2015.

[35] M. K. Gupta, M. C. Govil, and G. Singh, “A context-sensitive approach for precise detection of cross-site scripting vulnerabilities,” in *2014 10th International Conference on Innovations in Information Technology (IIT)*, Nov. 2014, pp. 7–12.

[36] H. Shahriar, S. North, W.-C. Chen, and E. Mawangi, “Design and development of anti-xss proxy,” in *8th International Conference for Internet Technology and Secured Transactions (ICITST-2013)*, Dec. 2013, pp. 484–489.

[37] D. Guamán, F. Guamán, D. Jaramillo, and M. Sucunuta, “Implementation of techniques and owasp security recommendations to avoid sql and xss attacks using j2ee and ws-security,” in *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, Jun. 2017, pp. 1–7.

[38] S. al Azmi and A. R. Khan, “A comprehensive research on xss scripting attacks on different domains and their verticals,” in *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 01, Dec. 2015, pp. 677–680.

[39] C. M. Frenz and J. P. Yoon, “Xssmon: A perl based ids for the detection of potential xss attacks,” in *2012 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, May 2012, pp. 1–4.

[40] I. Yusof and A. S. K. Pathan, “Preventing persistent cross-site scripting (xss) attack by applying pattern filtering approach,” in *The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, Nov. 2014, pp. 1–6.

[41] J. Bozic and F. Wotawa, “Xss pattern for attack modeling in testing,” in *2013 8th International Workshop on Automation of Software Test (AST)*, May 2013, pp. 71–74.

[42] C. H. Wang and Y. S. Zhou, “A new cross-site scripting detection mechanism integrated with html5 and cors properties by using browser extensions,” in *2016 International Computer Symposium (ICS)*, Dec. 2016, pp. 264–269.

[43] R. Wang, X. Jia, Q. Li, and D. Zhang, “Improved n-gram approach for cross-site scripting detection in online social network,” in *2015 Science and Information Conference (SAI)*, Jul. 2015, pp. 1206–1212.

[44] R. Wang, X. Jia, Q. Li, and S. Zhang, “Machine learning based cross-site scripting detection in online social network,” in *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS)*, Aug. 2014, pp. 823–826.

[45] M. R. Zalbina, T. W. Septian, D. Stiawan, M. Y. Idris, A. Heryanto, and R. Budiarto, “Payload recognition and detection of cross site scripting attack,” in *2017 2nd International Conference on Anti-Cyber Crimes (ICACC)*, Mar. 2017, pp. 172–176.

[46] G. Wassermann and Z. Su, “Static detection of cross-site scripting vulnerabilities,” in *2008 ACM/IEEE 30th International Conference on Software Engineering*, May 2008, pp. 171–180.

[47] A. Stasinopoulos, C. Ntantogian, and C. Xenakis, “Bypassing xss auditor: Taking advantage of badly written php code,” in *2014 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Dec. 2014, pp. 000 290–000 295.

[48] T. S. Rocha and E. Souto, “Etssdetector: A tool to automatically detect cross-site scripting vulnerabilities,” in *2014 IEEE 13th International Symposium on Network Computing and Applications*, Aug. 2014, pp. 306–309.

[49] B. Panja, T. Gennarelli, and P. Meharia, “Handling cross site scripting attacks using cache check to reduce webpage rendering time with elimination of sanitization and filtering in light weight mobile web browser,” in *2015 First Conference on Mobile and Secure Services (MOBISECSERV)*, Feb. 2015, pp. 1–7.

[50] K. S. Rao, N. Jain, N. Limaje, A. Gupta, M. Jain, and B. Menezes, “Two for the price of one: A combined browser defense against xss and clickjacking,” in *2016 International Conference on Computing, Networking and Communications (ICNC)*, Feb. 2016, pp. 1–6.

[51] B. Mewara, S. Bairwa, J. Gajrani, and V. Jain, “Enhanced browser defense for reflected cross-site scripting,” in *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization*, Oct. 2014, pp. 1–6.

[52] B. Rexha, A. Halili, K. Rrmoku, and D. Imeraj, “Impact of secure programming on web application vulnerabilities,” in *2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*, Nov. 2015, pp. 61–66.

[53] R. Johari and P. Sharma, “A survey on web application vulnerabilities (sqlia, xss) exploitation and security engine for sql injection,” in *2012 International Conference on Communication Systems and Network Technologies*, May 2012, pp. 453–458.

[54] H. Zeng, “Research on developing an attack and defense lab environment for cross site scripting education in higher vocational colleges,” in *2013 International Conference on Computational and Information Sciences*, Jun. 2013, pp. 1971–1974.

[55] J. You and F. Guo, “Improved csrfguard for csrf attacks defense on java ee platform,” in *2014 9th International Conference on Computer Science Education*, Aug. 2014, pp. 1115–1120.

[56] S. Ding, H. B. K. Tan, L. K. Shar, and B. M. Padmanabhuni, “Towards a hybrid framework for detecting input manipulation vulnerabilities,” in *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, vol. 1, Dec. 2013, pp. 363–370.

[57] J. Pan and X. Mao, “Detecting dom-sourced cross-site scripting in browser extensions,” in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2017, pp. 24–34.

[58] I. Dolnák, “Content security policy (csp) as countermeasure to cross site scripting (xss) attacks,” in *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, Oct. 2017, pp. 1–4.

[59] R. M. Pandurang and D. C. Karia, “Impact analysis of preventing cross site scripting and sql injection attacks on web application,” in *2015 IEEE Bombay Section Symposium (IBSS)*, Sep. 2015, pp. 1–5.

[60] P. A. Sonewar and N. A. Mhetre, “A novel approach for detection of sql injection and cross site scripting attacks,” in *2015 International Conference on Pervasive Computing (ICPC)*, Jan. 2015, pp. 1–4.

[61] D. Lan, W. ShuTing, Y. Xing, and Z. Wei, “Analysis and prevention for cross-site scripting attack based on encoding,” in *2013 IEEE 4th International Conference on Electronics Information and Emergency Communication*, Nov. 2013, pp. 102–105.

[62] Y. Sun and D. He, “Model checking for the defense against cross-site scripting attacks,” in *2012 International Conference on Computer Science and Service System*, Aug. 2012, pp. 2161–2164.

[63] K. Gupta, R. R. Singh, and M. Dixit, “Cross site scripting (xss) attack detection using intrustion detection system,” in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, Jun. 2017, pp. 199–203.

[64] R. Wang, G. Xu, X. Zeng, X. Li, and Z. Feng, “Tt-xss: A novel taint tracking based dynamic detection framework for dom cross-site scripting,” *Journal of Parallel and Distributed Computing*, vol. 118, pp. 100–106,

[65] S. Calzavara, G. Tolomei, M. Bugliesi, and S. Orlando, “Quite a mess in my cookie jar!: Leveraging machine learning to protect web authentication,” in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW ’14, Seoul, Korea, 2014, pp. 189–200, ISBN: 978-1-4503-2744-2.

[66] Y. Mundada, N. Feamster, and B. Krishnamurthy, “Half-baked cookies: Hardening cookie-based authentication for the modern web,” in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS ’16, Xi’an, China, 2016, pp. 675–685, ISBN: 978-1-4503-4233-9.

[67] S. Calzavara, G. Tolomei, A. Casini, M. Bugliesi, and S. Orlando, “A supervised learning approach to protect client authentication on the web,” *ACM Trans. Web*, vol. 9, no. 3, 15:1–15:30, Jun. 2015, ISSN: 1559-1131.

[68] S. Gupta and B. B. Gupta, “Xss-safe: A server-side approach to detect and mitigate cross-site scripting (xss) attacks in javascript code,” *Arabian Journal for Science and Engineering*, vol. 41, no. 3, pp. 897–920, Mar. 2016.

[69] ——, “Xss-secure as a service for the platforms of online social network-based multimedia web applications in cloud,” *Multimedia Tools and Applications*, vol. 77, no. 4, pp. 4829–4861, Feb. 2018.

[70] L. Weichselbaum, M. Spagnuolo, S. Lekies, and A. Janc, “Csp is dead, long live csp! on the insecurity of whitelists and the future of content security policy,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2016, pp. 1376–1387.

[71] W3C. (May 2018). Content security policy 1.0, [Online]. Available: https://www.w3.org/TR/CSP1/.

[72] OWASP. (May 2018). .net antixss library, [Online]. Available: https://www.owasp.org/index.php/.NET_AntiXSS_Library.

[73] ——, (May 2018). Proyecto owasp api de seguridad empresarial (esapi), [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API/es.

[74] APACHE. (May 2018). Apache wicket, [Online]. Available: https://wicket.apache.org/.

[75] M. K. Gupta, M. C. Govil, and G. Singh, "Static analysis approaches to detect sql injection and cross site scripting vulnerabilities in web applications: A survey," in *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, May 2014, pp. 1–5.

[76] V. Nithya, S. L. Pandian, and C. Malarvizhi, "A survey on detection and prevention of cross-site scripting attack," *International Journal of Security and Its Applications*, vol. 9, no. 3, pp. 139–52, 2015.

[77] M. K. Gupta, M. C. Govil, and G. Singh, "Text-mining based predictive model to detect xss vulnerable files in web applications," in *2015 Annual IEEE India Conference (INDICON)*, Dec. 2015, pp. 1–6.