IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 7, January 2023

Hacking the Web: "A Deep Dive into Cross-Site Scripting (XSS)"

Jayashree Patade, Abdul Qadir, Shoeb Khan
Shri G.P.M. Degree College of Science and Commerce, Andheri, Mumbai, Maharashtra

Abstract: *Advances in technology and the digitization of organizational functions and services have moved the world into a new era of computing capabilities and sophistication. The proliferation and usabilityof such complex technological services raises several security issues. One of the most critical issuesis cross-site scripting (XSS) attacks. This paper focused on concise and accurate detection and comprehensive analysis of XSS injection attacks, detection and prevention. I conducted a thorough study and reviewed several research papers and publications with a specific focus on researchers' defensive techniques to prevent XSS attacks and divided them into five categories: machine learning techniques, server-side techniques, client-side techniques, proxy-based techniques, and combined approaches. Most of the existing state-of-the-art XSS defense approaches carefully analyzed in thispaper offer protection against traditional XSS attacks such as stored and bounced XSS. There is currently no reliable solution that provides adequate protection against a newly discovered XSS attack known as DOM-based and mutation-based XSS attacks. After reading all the proposed modelsand identifying their flaws, I recommend a combination of static, dynamic, and code auditing in conjunction with secure coding and ongoing user awareness campaigns about new XSS attacks.*

Keywords: XSS Attacks, Defensive Techniques, Vulnerabilities, Web Application Security.

I. INTRODUCTION

XSS (Cross-Site Scripting) is a programming error that occurs when user input is not properly sanitized. An attacker exploits this vulnerability to inject unfiltered scripting code into a web application, resulting in account takeover, session theft, or cooking and redirection to the attacker's website when the parser processes the script. An XSS attack can be launched on any vulnerable webpage written in any programming language. A thorough analysis of cross-site scripting vulnerabilities was presented in detail. We talked about what XSS is, the many forms of XSS attacks,how an attacker can exploit this weakness, the results of an XSS attack, and the protection strategiesthe research community has put in place to combat XSS attacks. On the other hand, we examined these defense strategies and identified flaws in how they defended against specific XSS attacks. However, despite the efforts of researchers, XSS attacks can still disrupt web applications to a greaterextent regardless of the fact that various tactics and approaches have been implemented to prevent the vulnerability. Due to the virtually unchanged behavior of the browser, it is difficult to detect XSSattacks and distinguish between malicious JavaScript and legitimate online content. Several sections of the article are neatly organized according to relevant topics: Definition and classification of XSS,as well as the injection methods used by XSS and the damage it causes to web applications, are covered in Segment 2. Segment 3 describes the research data composition and compares CWE names using development vulnerability data analysis software. Segment 4 presents related work. Segment 5 discusses XSS prevention and defense mechanism along with researchers' defensive techniques for XSS attacks (advantages and disadvantages). Segment 6 describes the challenges associated withdetecting and defending against XSS attacks, along with the precise precautions that should be implemented in response to a given episode. The current issue is broken down into individual parts,and then a perspective for the future is presented.

II. BACKGROUND OF THE CROSS-SITE SCRIPTING ATTACK

2.1 Categories of XSS Attacks

A cross-site scripting attack typically occurs when an attacker attacks a website by injecting malicious JavaScript code into client-side input parameters. Figure 1 shows a comprehensive view of the four XSS attack scenarios described in this document. The XSS vulnerability exploits the fact that web applications run scripts in user browsers. If the user

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 7, January 2023

manipulates or changes the dynamically generated script, he exposes the online application to danger. Although four categories of XSS attacks are mentioned in this document, as shown in Figure 1, most current web application developers and researchers are familiar with only three of them because they are more common in the research community. Organizations such as the Open Web Application Security Project (OWASP) have recognized these three types of XSS attacks as the most common XSS attack vectorson the web.

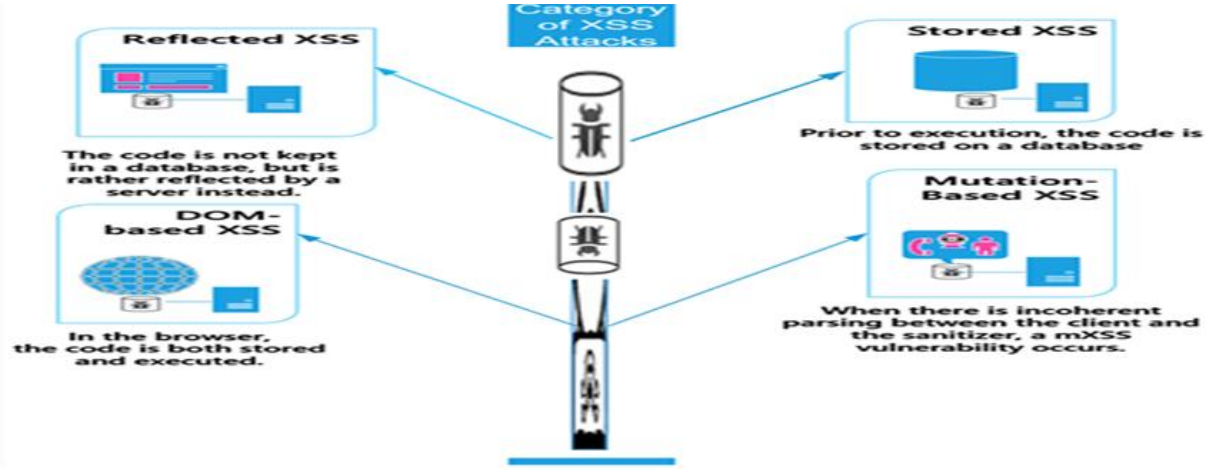

Figure 1. A brief overview of the four categories of cross-site scripting vulnerabilities.

2.2 Stored Cross-Site Scripting (XSS) Attack

This form of XSS vulnerability is sometimes referred to as persistent XSS. This is due to the fact that the malicious script is still present on the server even after the attack is complete. During this type of attack, an attacker injects code that has been maliciously written to the server in a way that cannot be removed. As shown in Figure 3, the scenario I used to illustrate a stored XSS attack inserted a script tag directly into the Document Object Model (DOM) and then hypothetically executed a malicious script using JavaScript. While this is the most popular way to exploit XSS, it is also the most common approach that has been neutralized by advanced security professionals and security-conscious software developers. A user uploads a malicious XSS script to the database, which is requested and viewed by other users, resulting in the script being executed on their systems,as described in Figure 3.

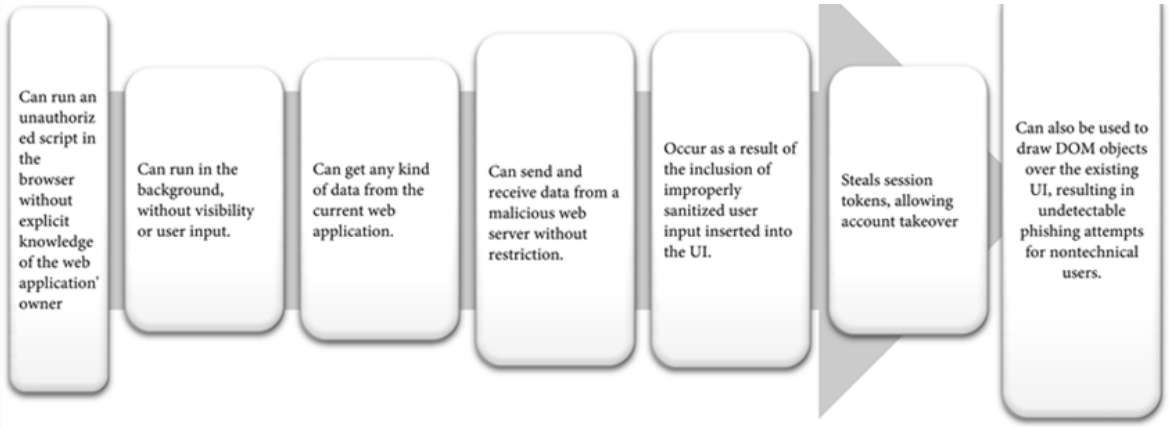

Figure 2. Injection methods of a typical cross-site scripting attacks.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 7, January 2023

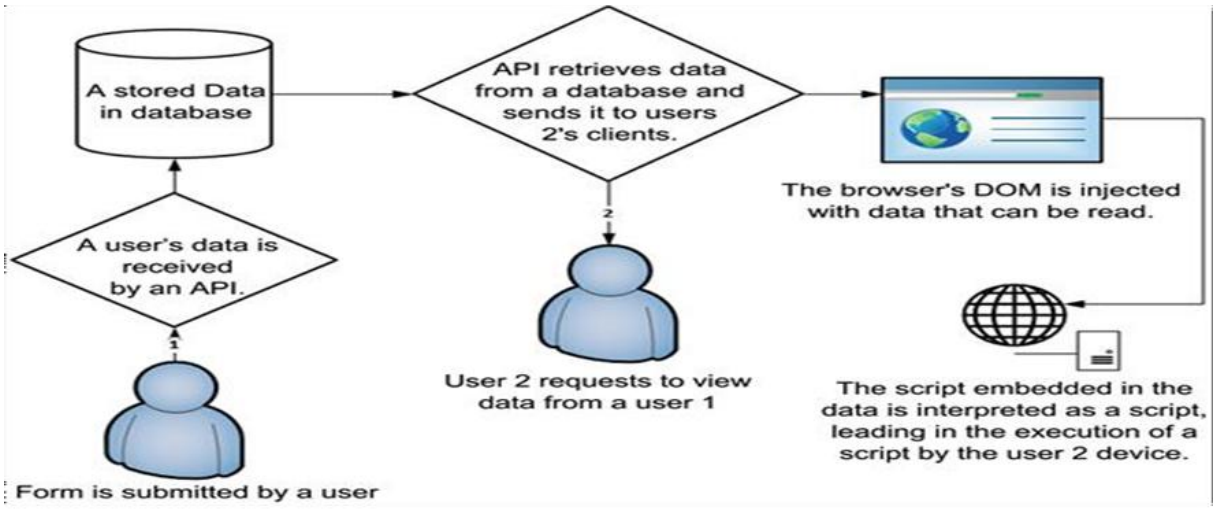

Figure 3. Stored XSS attack scenario.

2.3 Reflected Cross-Site Scripting (XSS) Attack

Reflected XSS attacks, also known as non-permanent attacks, create a URL that allows an attacker to inject arbitrary script into a targeted web application. Most publications and academic sources introduce reflected XSS before addressing the concept of preserved XSS. For inexperienced programmers, reflected XSS attacks are often more difficult to detect and exploit than cached XSS attacks.

Cached XSS attacks are relatively easy to understand from a developer's perspective. Clients provideresources to servers. This is usually done via the HTTP protocol. After the server receives the requested resource from the client, it enters it into the database. When another client later accesses the resource, the malicious script is inadvertently executed in the client's Internet browser, as shownin Figure 3.

Reflected XSS attacks, on the other hand, work similarly to cached XSS attacks, but do not require a database or server. As shown in Figure 4, the client code is directly affected in the browser, so theserver is not involved in the reflected XSS attack. Web applications can be vulnerable to this type ofattack depending on the actions users take (see Figure 4). Scripts that are not stored on the user's computer.

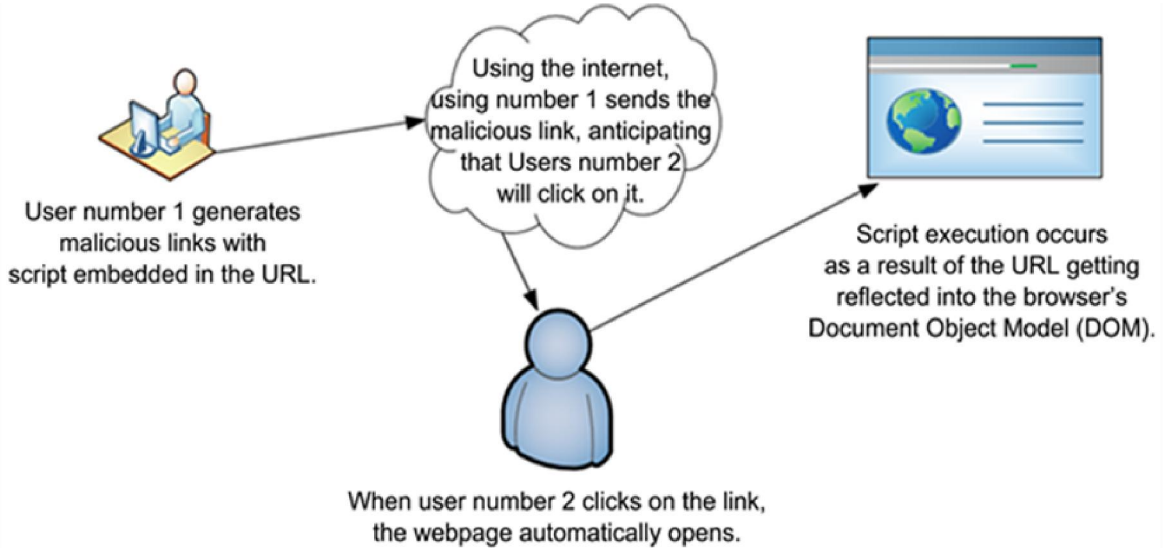

Figure 4. Reflected cross-site scripting scenario.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 7, January 2023

2.4 Document Object Model-Based Cross-Site Scripting (XSS) Attack

A DOM-based XSS attack is clearly a client-side attack. The type of XSS attack based on the DOMmodel is shown in Figure 5 as the third important classification of XSS attacks

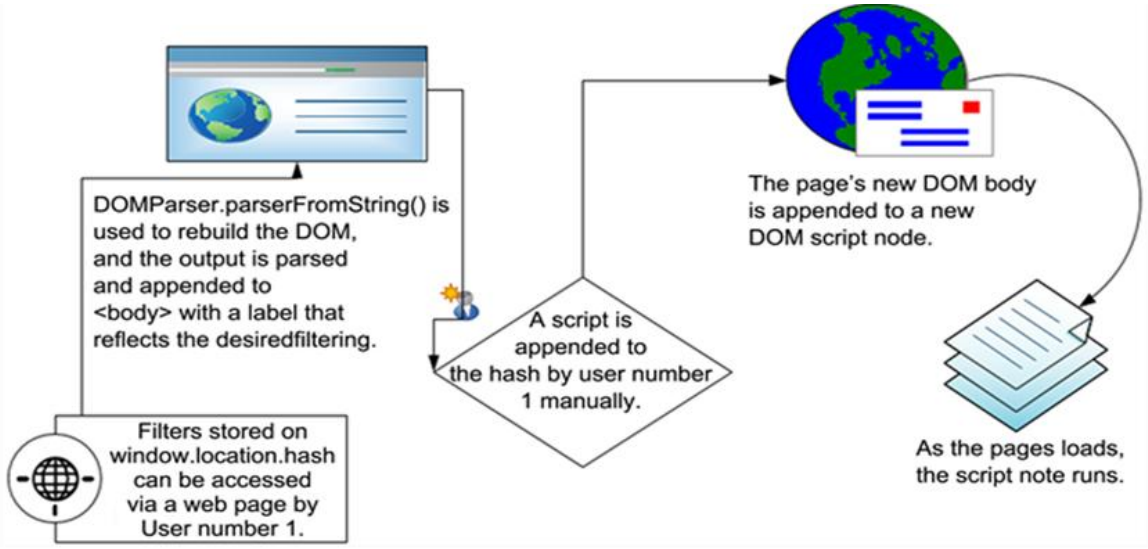

Figure 5. Dom-based cross-scripting attack scenario.

The implementation of the DOM in different browsers can make some browsers vulnerable while others are not. Compared to typical reflected or cached XSS attacks (Figure 3 and Figure 4), these XSS attacks require extensive knowledge of the browser's DOM and JavaScript to be discovered and exploited. DOM-based XSS attacks are fundamentally different from other types of XSS in that theydo not require any communication with the server. By convention, the source is usually a DOM object that can store text, and the sink is generally a DOM API that can run a script that has been stored as text. For DOM XSS to work, both a "source" and a "sink" must be present in the browser's DOM, since no server is involved. In most cases, the sink is a DOM API that can run a script storedin the resource as text. It is almost impossible to detect DOM XSS using static analysis tools or otherpopular scanners because it never touches the server

2.5 Mutation-Based Cross-Site Scripting (mXSS) Attack

Dr. In his publication, Mario Heiderich revealed six (6) new subclasses of mXSS attacks In an mXSSattack, the DOM can be completely avoided using InnerHTML, which allows automatic changes tothe HTML content. mXSS is sometimes referred to as mutated XSS or mutation-based XSS.

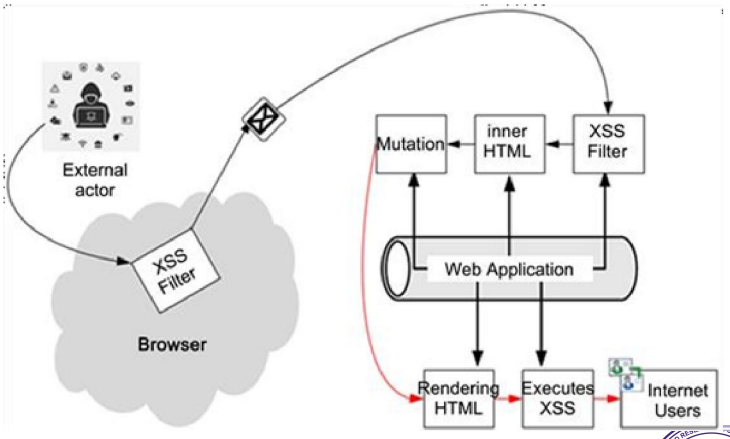

Figure 6. Mutation-based cross-site scripting (mXSS) attack scenario.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 7, January 2023

This is because it is difficult to predict and involves recursion. When the HTML script is loaded into the browser's document object model, the data is mutated, causing an error. However, the content loaded into the browser's DOM is mutated to verify that it is error-free and does not contain any incorrect tags. This is achieved using the element. An internal HTML attribute. The fundamental disadvantage of this form of XSS attack is its ability to bypass server-side defenses and client-side filters. Figure 6 shows a potential mutation-based XSS attack scenario.

When an external actor inserts something that appears to be safe, as shown in Figure 6, the browser overwrites and modifies it when processing the HTML, resulting in a mutated XSS attack. This makes it incredibly difficult to find and debug errors in application logic. Despite its novelty and widespread misinterpretation, mXSS attacks have been used to bypass the most sophisticated XSS filters available. mXSS was used to bypass solutions such as DOMPurify, OWASP AntiSamy, and Google Caja, and a large number of popular web applications (especially email clients) were found to be vulnerable. At its core, mXSS works by using filter-safe payloads that mutate to insecure payloads after filtering. All major browsers are vulnerable to mXSS attacks. Developers must understand how browsers handle optimizations and conditionals when rendering DOM nodes.

2.6 Composition of XSS Comparative Research Data Sources

This research uses a subset of the global dataset containing the CVE and CWE security vulnerability database. However, I only focused on the software development component containing information containing CVE details for XSS vulnerability assessment, as shown in Figure 7. The data consists of CVE-ID, CWE-ID, Explanation, Severity, and the CVSS and CWE names under which the vulnerability falls.

However, the abbreviations and acronyms used in this survey are carefully explained in Section 3.1.

As Figure 7 shows, this was the dataset category used from a programming perspective. The results of this survey were thoroughly analyzed to determine annual trends in XSS vulnerabilities.

```
In [34]: year_list=[]
         for i in range(0,len(goodSD["CVE-ID"])):
             a=goodSD["CVE-ID"][i].split("-")[1]
             year_list.append(a)

         goodSD.insert(loc=2, column='Year', value=year_list)
         goodSD.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 59833 entries, 0 to 59832
         Data columns (total 9 columns):
          #   Column       Non-Null Count  Dtype
         ---  ------       --------------  -----
          0   List         59833 non-null  int64
          1   CVE-ID       59833 non-null  object
          2   Year         59833 non-null  object
          3   CVSS-V3      59833 non-null  object
          4   CVSS-V2      59833 non-null  object
          5   SEVERITY     59833 non-null  object
          6   Explanation  59833 non-null  object
          7   CWE-ID       59833 non-null  object
          8   CWE-NAME     59833 non-null  object
         dtypes: int64(1), object(8)
         memory usage: 4.1+ MB

In [36]: goodSD.describe(include=["object", "bool"])
Out[36]:
```

	CVE-ID	Year	CVSS-V3	CVSS-V2	SEVERITY	Explanation	CWE-ID	CWE-NAME
count	59833	59833	59833	59833.0	59833	59833	59833	59833
unique	59060	24	75	64.0	5	55840	150	152
top	CVE-2018-1000224	2020	None	4.3	MEDIUM	Multiple vulnerabilities in the web-based mana...	CWE-79	Improper Neutralization of Input During Web Pa...
freq	5	9557	13825	16503.0	28384	122	18566	18564

Figure 7. A brief overview of the dataset used for analyzing XSS vulnerability.

Abbreviations and Acronyms

XSS = Cross-Site Scripting;

DOM = Document Object Model;

mXSS = Mutation-Based Cross-Site Scripting; NVD [23] [24] = National Vulnerability Database;

CVE [25] = Common Vulnerabilities and Exposures; CWE [26] = Common Weakness Enumeration; CVSS = Common Vulnerability Scoring System.

Comparative of the Top 20 Software Development Vulnerabilities

The pie charts below illustrate the number of the top 20 Software Development Vulnerabilities based on CWE Name from 2014 to 2022. Over the last nine years, the most frequent report of a cross-site scripting (XSS) vulnerability has been alarmingly received, as shown in Figure 8. I used python Jupiter Notebook to analyze the data.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 7, January 2023

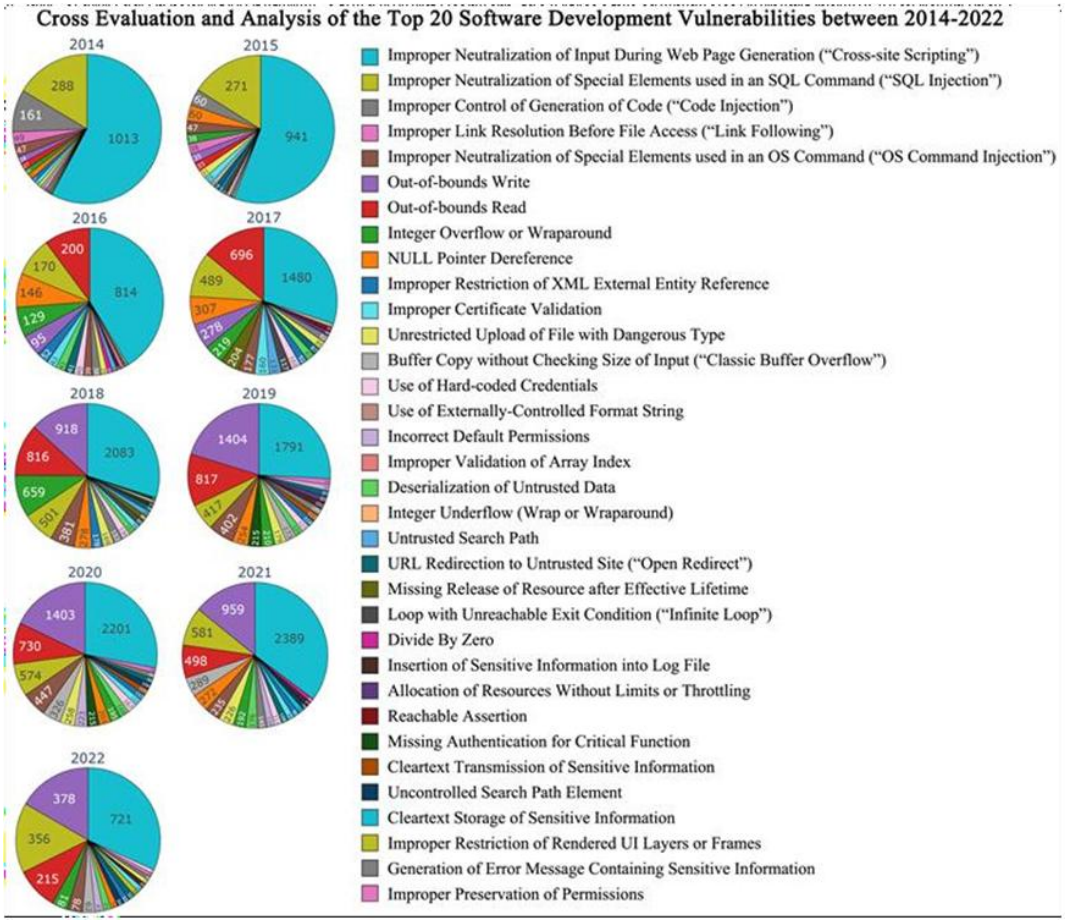

Figure 8. Comparative analysis of XSS vulnerability's yearly trends

IIII. XSS PREVENTION AND DEFENSE MECHANISM

The XSS prevention and defence mechanism are explicitly explained in the following sections:

3.1 Preventive Measures and Standard Procedures for Cross-Site Scripting Attack

This section highlights most of the standard solutions that can be used to significantly reduce the impact of XSS attacks. It highlights the description of XSS mitigation rules that developers can implement to prevent XSS attacks. Obviously, these techniques are not magic; they are ineffective without sufficient user awareness. Figure 8 illustrates that only two vulnerabilities, XSS and SQL injection, dominate the web application security attack field. Developers can now protect themselves against XSS attacks using numerous measures. User-entered data that is not trusted is protected using a combination of filtering, leakage, and sanitization procedures. The following Table 1 and Table 2describe each technique: There are two types of escaping: input escaping and output escaping. Practical input escaping requires proper context detection of embedded untrusted data. In contrast, output escaping is performed on the written data of the response web page. It also takes into accountthe context of the data, which is necessary to mitigate stored XSS attacks

Technique	Explanation
Filtering	This implies that any unsafe user input must befiltered to remove dangerous phrases like the

Table 1. General methods for preventing XSS attacks.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 7, January 2023

Character	Encoded Format
/	/ or & #47
‘	‘ or & #39
“	& quot; or & #34
>	> or & #62
<	< or & #60
&	& amp; or & #38
#	& #35
)	& #41
(	& #40

Table 2. HTML entity encoding

The spread of XSS vulnerabilities is attracting the interest of security researchers and developers. The variety of XSS attacks that each solution is designed to defend against has inspired the development of a wide range of countermeasures. Based on the metrics of their implementation model, I grouped these solutions or techniques into five categories: client-side techniques, server- side techniques, machine learning techniques, and proxy-based techniques. In the following subsections, I have highlighted the most significant and effective methods proposed by researchers as advantages and observed the limitations of these approaches as disadvantages. See the appendix for more information on the researchers' techniques

IV. DISCUSSION AND CONCLUSION

Discussion: The deep dive into Cross-Site Scripting (XSS) has shed light on the significance of thisweb vulnerability, its potential consequences, and the measures needed to mitigate it. During our deep dive into Cross-Site Scripting (XSS), we also explored the evolving nature of this threat and the need for continuous adaptation and improvement

Conclusion

In conclusion, this discussion underscores the multifaceted nature of XSS and the necessity for a collaborative approach to mitigate its risks. XSS remains a formidable challenge in web security that demands vigilance, cooperation, and adaptability.

Final Thoughts

This deep dive into XSS serves as a call to action for web developers, ethical hackers, legal experts, and users to work collectively to make the web a safer place. By embracing a proactive stance, staying informed about emerging threats, and fostering a culture of security awareness, we can better protect ourselves and our digital environments. We appreciate the insights and expertise shared by our panelists and the engagement of our audience in this discussion. As we navigate the ever- changing web security landscape, let's remember that security is a shared responsibility, and together, we can mitigate the risks of XSS and other vulnerabilities, making the web a more secure place for all.

REFERENCES

[1] Kirsten, S. (2016) Cross Site Scripting (XSS) Software Attack. https://owasp.org/www-community/attacks/xss/
[2] Agrawal, D.P. and Wang, H. (2018) Computer and Cyber Security. Auerbach Publications, New York. https://doi.org/10.1201/9780429424878
[3] Jiang, F., Fu, Y., Gupta, B.B., Liang, Y., Rho, S., Lou, F., et al. (2020) Deep Learning Based Multi-Channel Intelligent Attack Detection for Data Security. IEEE Transactions on Sustainable Computing, 5, 204-212. https://doi.org/10.1109/TSUSC.2018.2793284
[4] Baş Seyyar, M., Çatak, F.Ö. and Gül, E. (2018) Detection of Attack-Targeted Scans from the Apache HTTP Server Access Logs. Applied Computing and Informatics, 14, 28-36. https://doi.org/10.1016/j.aci.2017.04.002

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 7, January 2023

[5] Chen, H.-C., Nshimiyimana, A., Damarjati, C. and Chang, P.-H. (2021) Detection and Prevention of Cross-Site Scripting Attack with Combined Approaches. 2021 International Conference on Electronics, Information, and Communication (ICEIC), Jeju, 31 January-3 February 2021, 1-4. https://doi.org/10.1109/ICEIC51217.2021.9369796

[6] Gan, J.-M., Ling, H.-Y. and Leau, Y.-B. (2021) A Review on Detection of Cross-Site Scripting Attacks (XSS) in Web Security. International Conference on Advances in Cyber Security, Penang, 8-9 December 2020, 685-709. https://link.springer.com/chapter/10.1007/978-981-33-6835-4_45

[7] Wibowo, R.M. and Sulaksono, A. (2021) Web Vulnerability Through Cross Site Scripting (XSS) Detection with OWASP Security Shepherd. Indonesian Journal of Information Systems, 3, 149-59. https://doi.org/10.24002/ijis.v3i2.4192

[8] Dora, J.R. and Nemoga, K. (2021) Ontology for Cross-Site-Scripting (XSS) Attack in Cybersecurity. Journal of Cybersecurity and Privacy, 2021, 319-339. https://doi.org/10.3390/jcp1020018

[9] Nirmal, K., Janet, B. and Kumar, R. (2018) Web Application Vulnerabilities—The Hacker's Treasure. 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, 11-12 July 2018, 58-62. https://doi.org/10.1109/ICIRCA.2018.8597221

[10] Cui, Y., Cui, J. and Hu, J. (2020) A Survey on XSS Attack Detection and Prevention in Web Applications. Proceedings of the 2020 12th International Conference on Machine Learning and Computing, Shenzhen, 15-17 February 2020, 443-449. https://doi.org/10.1145/3383972.3384027

[11] Khazal, I. and Hussain, M. (2021) Server Side Method to Detect and Prevent Stored XSS Attack. Iraqi Journal for Electrical and Electronic Engineering, 17, 58-65. https://doi.org/10.37917/ijeee.17.2.8