International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 04 | Apr 2019 www.irjet.net p-ISSN: 2395-0072

A Survey on Various Cross-Site Scripting Attacks and Few Prevention Approaches along with a Conducive Approach

Akshay Deodare[1], Jyoti Raghatwan[2]

[1,2]*Department of Computer Engineering, RMD Sinhgad School of Engineering, Maharashtra, India*

Abstract - *In this paper, a very destructive threat Cross – Site Scripting is discussed where the attack script is executed by the browser. Commercial reasons have made a necessity of building dynamic web applications. Web application vulnerabilities are mostly found due to lack of input or output sanitization moreover these are often exploited to manipulate source code or gain unauthorized access. Social engineering is intentionally executed to gain the evil benefits of a reflected cross-site scripting script. Design of the third party web application can be changed by storing the script permanently on the applications server. Also few preventive methods proposed earlier are portrayed in this paper along with a favourable method to detect base64 based evil script. Every method presented has certain limitations and to overcome these various approaches should be designed. . The base64 encoded malicious script can bypass firewall if it isn't secured well. The algorithmic flowchart construed in this paper decodes the base64 format and then checks for XSS attack vector.*

Keywords- **XSS, Cross-Site Scripting, base64, security.**

1. INTRODUCTION

It has been a high time since the era of HTML only web pages are gone, there was no exchange of data between server and client. Dynamic webpages have made internet content more interactive [1]. They execute codes on user's server and manages server data. As these web pages store and access the data in a database it is necessary to protect the web application from security threats. With the rise in dynamic webpages, security threats have also increased [2]. If the web application is not enough protected it can be a victim of evil attacks such as SQL injection, Cross-Site Scripting, XML injection, Host header attack, Denial of service and many other dangerous scripts.

1.1 Cross Site Scripting (XSS)

XSS is an injection attack which allows a hacker to execute scripts in a user's browser. The user becomes a victim when he visits the webpage that implements the evil script. The web application behaves as a deliverer of the attack script to the browser. The scripting language mostly used is JavaScript because it is core or necessity of maximum browsing experiences [3]. Basically, XSS can be used to deface your web application.

Various methods such as Escaping and Restriction are used to prevent and mitigate the impact of XSS attacks. Firewalls are also put up to protect the sensitive data that provides web services [2].

2. SURVEY OF EXISTING ATTACKS AND THEIR PREVENTION

2.2 Cross-site scripting example

<script>window.location='http://www.website/?cookie='+document.cookie<script>

This example shows the cross-site scripting implementation to steal the cookies of the user [3]. An attacker sends a script injected link to the victim which redirects him to a dissimilar website. The evil script gets executed onto the browser and the sensitive data that is cookies are fetched by the attacker [5].

2.2 Cross Site Scripting Types

2.2.1 DOM -based

The script used to seek sensitive data uses the document object [5]. The script appears in DOM rather than in HTML format. Here it is not possible to find the payload in response. Netsparker a widely known automated vulnerability finding software company demonstrates DOM based XSS in the following way
Consider URL *example.com/test.*html contains the code.

<script>document.write("Current URL : " + document.baseURI);</script>

If an http request with xss script like this *http://example.com/test.html#<script>alert(1)</script*> is made , JavaScript code gets executed as the page writes everything typed in url with document.write() function. Anything written after # is never sent to server. We cannot find *<script>alert(1)<script>* in the source code because all this was happening in DOM.

2.2.2 Reflected XSS

This method usually requires social engineering to trick the victim. Social engineering means tricking the user by manipulating user to achieve a successful conclusion of the attack [6]. The XSS occurs or impacts when the user opens a maliciously crafted link. The browser-executable code is not stored in the application.

An attacker creates and tests the website if it is vulnerable to XSS. If found vulnerable he tricks the victim using social engineering to load the infected URL (similar to the tested website) on the browser. Eventually, the evil code is executed using victims browser [2].

This may include session stealing or installing key loggers or changing the webpage content.

2.2.3 Stored or persistent XSS

Certain web applications allow users to store the content on the server such as comments, FAQs, etc. These type of web applications can be prey of Stored Cross Site Scripting. The web application gathers input that might be a hacking script through certain forms and then stores them on the server. The input stored is not properly validated thus the malicious data appears on the website on the victim's browser.

Stored XSS doesn't need a specific victim to carry out the attack. The successful conclusion occurs when anybody visits the XSS affected site. An attacker requires a host input form in order to execute the malicious script

Scenario:

- An attacker tests the vulnerability of the web application.
- On successful testing, he plants the evil script.
- Victim visits the vulnerable webpage.
- User's Browser executes the malicious code.

The malicious script is stored permanently on the server and is executed every time the user opens the infected web application.

2.2.4 Induced XSS

If a web application is vulnerable to HTTP Response Splitting Vulnerability, Induced XSS can be executed. The attacker usually manipulates the HTTP header of the server's response. Induced XSS's are not very common but can still be mentioned in the classification [4].

2.3 Preventive Approaches

In Reflected XSS a maliciously scripted link is sent in order to trick the user. Florian Kerschbaum proposed a lightweight and efficient way to prevent XSS in 2007. It ensures that input given to the web application from the user's browser is not shaped by an attacker by forwarding a link. The website must contain 2 types of pages namely entry and regular pages. No input can be accepted by entry pages unless a gateway filters the inputs or discards them. The system allows access to the regular pages (servlets or scripts, applications) only through entry pages. Gateway divides the input into query URLs or POST request data while accessing the entry page and gives access to regular pages only if a cookie is set and the referrer string originates from the same site [7].

Imran Yusof and Al-Sakib Khan Pathan gave an approach for preventing Persistent XSS. The method works on input filtering. Input data must be filtered such that the browser doesn't execute the evil scripts. User input must be sanitized before it gets stored in the database [8]. Event handlers, Data URIs are filtered using regular expressions. Insecure keywords such as document.cookie, document.write, etc. are filtered by either encoding them or blocking them. Dangerous characters used in XSS payloads can be escaped using &# followed by the character code [8].

Secure PHP functions can be used to prevent XSS attack using 2 stages as proposed by Twana Assad TAHA and Murat KARABATAK in 2018. First, the input validation is carried out using regular expressions and then another regular expression for checking if any malicious script is present in the input field. This malicious code will be removed instantly on detection. Certain PHP functions such as htmlEntities() and htmlspecialchar() are used to convert characters HTML entities. As proposed by the above two authors the two methods are AllowList regular and DenyList regular expression. AllowList allows only trusted and expected user inputs i.e. it performs validation (contact number must not contain characters other than + and numbers) whereas the DenyList regular expression checks the suitability of data and abandons or substitutes the suspicious characters such as HTML and script tags [3].

3. STUDY RESULTS

XSS ranks 7th according to Owasp top 10 vulnerabilities [10]. The threat of reflected, stored and DOM-based XSS is almost the same. The only difference arises is due to the way of attacking. At some point, the webpage is unable to control the malicious JavaScript and thus become the prey of XSS.

Stored XSS stores the impact permanently on the server. Impact of Reflected XSS is seen when the specific user

interacts with the malicious script. DOM-based XSS destruction occurs if a user interacts with a malicious script that is created on the basis of Document Object Model.

Table-1:XXS TYPES AND CHARACTERISTICS

XSS type	Characteristics
Stored	Stored permanently on a server
Reflected	Requires specific user interaction
DOM	The script includes Document Object
Induced	Manipulates the HTTP header of the server's response

XSS vulnerabilities cannot be totally prevented. An assumption of the characters and strings is made to validate the input fields. A highly complex attack vector must be first developed in order to prevent the subcategories of that evil script and that script itself. PHP provides an input filtering layer using certain functions. An application should never output data received as input directly to the browser without checking it for malicious code. These suggested methods for user input validation are mostly used to defend XSS [3].

1. Replacement: replace the evil script in the input with true characters
2. Removal: remove the dangerous inputs
3. Escaping: dangerous characters are escaped by adding &# then the character code [8].
4. Firewalls are usually established to protect the server side. Moreover web applications can be tested using software such as Acunetix, Netsparker, etc

Table -2: COMPARISON OF ABOVE MENTIONED METHODS

Method	Limitations
Building a gateway[7]	• The browser must send a reliable reference string. • Bookmarks can be set only to the entry pages. • Can detect and avoid reflected XSS only
Pattern Filtering Approach[8]	• An assumption of the characters and strings is made to validate the input fields. • The approach is built only for Stored XSS. • Manual filtering of characters is required.
Use of PHP functions and regular expressions to detect and prevent XSS[3]	• If the XSS script is encoded in base64 format, it might not get detected.

4. CONDUCIVE METHOD TO PREVENT XSS

A method can be developed to detect the XSS script that is encoded in base64 format. One way that might detect a base64 encrypted script is, to set up a decode mechanism in the web application source code and build a regular expression that matches the decoded format in order to escape it in an efficient way.

4.1 Algorithm

1. Decode the base64 script.
2. Set a pattern of regular expression to detect and protect the input entry that may encompass the evil script.
3. End

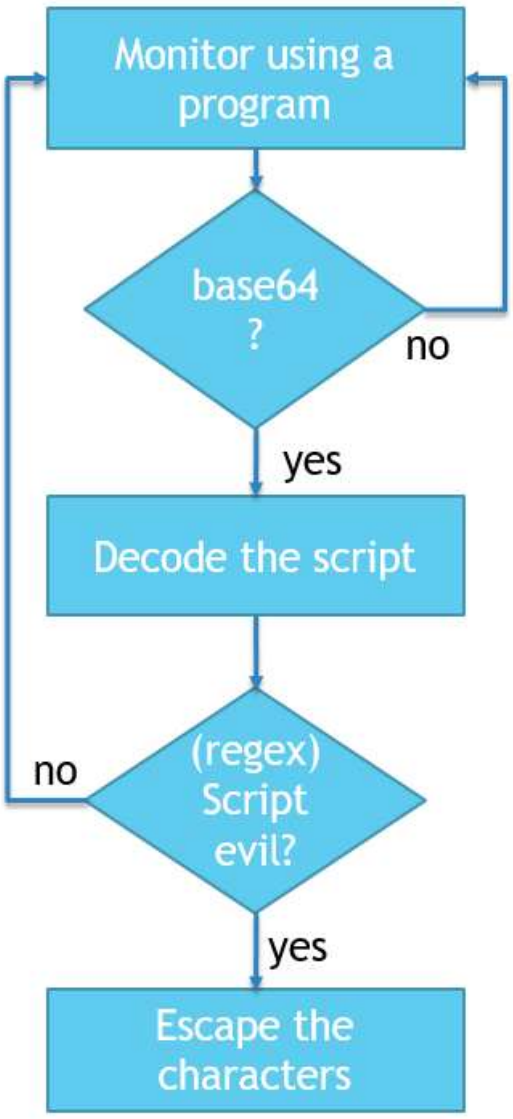

Figure-1 DETECTION OF BASE64 ENCODED XSS

5. CONCLUSION

This paper presents a detailed survey of a few XSS prevention methods, also specifies the ways in which various cross-site scripting attacks can be exploited. The limitations of different methods vary. A method that might be conducive to detect base64 based XSS script has been proposed. In the course of time, these proposed methods (by respective authors) may become less useful due to rapid sophistication of evil attacks. Web application developers must know the latest threats that may affect the web applications. New threats are discovered by hackers in very less time so the developers must also carry out the research continuously to make the digital world secure.

REFERENCES

[1] N. Niu E. Stroulia M. El-Ramly: Understanding Web usage for dynamic Web-site adaptation: a case study. 2002 Proceedings. Fourth International Workshop on Web Site Evolution.

[2] K. Pranathi, S. Kranthi, Dr.A.Srisaila, P. Madhavilatha: Attacks on web Application Caused by Cross Site Scripting: 2018 2nd international conference on electronics, Communication and Aerospace Technology.

[3] Twana Assad TAHA, Murat Karabatak: A proposed approach for preventing Cross Site Scripting: 2018 6th International Symposium on Digital Forensic and Security (ISDFS)

[4] V.K Malviya, S.Saurav: On security issues in web applications through cross-site scripting: 2013 20th Asia Pacific Software Engineering Conference (AtiPSEC), Bangkok, 2013, pp.583-588

[5] Mohit Dayal, Nanhay Singh, Ram Shringar Raw: A comprehensive Inspecon of Cross Site Scripting Attack. International Conference on Computing, Communication, and Automation (ICCCA2016)

[6] Francois Mouton; Mercia M. Malan; Louise Leenen; H.S. Venter: Social engineering attack framework. 2014 Information Security for South Africa

[7] Florian Kerschbaum 2007. Simple Cross-Site Attack Prevention: 2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007.

[8] Imran Yusof, Al-Sakib Pathan: Preventing Persistent Cross-Site Scripting (XSS) Attack By Applying Pattern Filtering Approach.

[9] https://www.netsparker.com/blog/web-security/dom-based-cross-site-scripting-vulnerability/

[10] https://www.veracode.com/directory/owasp-top-10