ISSN 2395-1621

Framework to Detect and Avoid XSS and Hiding Sensitive Information Using Shadowing

[#1]Ms. Aishwarya Bolegave, [#2]Ms. Devyani Bhosale, [#3]Ms. Mamta Burhade, [#4]Mr. Sandeep Rai, [#5]Prof. Premlata G.

[1]aishbolegave1995@gmail.com
[2]devyanibhosale19@gmail.com
[3]mamtaburhade007@gmail.com
[4]sandeep.rai248@gmail.com

[#1234]Department of Information Technology
[#5]Prof. Department of Information Technology

ICOER, SPPU, Pune, India.

ABSTRACT

Web application is made up of different components like static images or dynamic links. Protected web component can keep private data safe from opportunistic attackers by hiding static data in DOM and isolating sensitive elements within web components. Inclusion of third party code in the website is dangerous, this expose private data to attackers. Protected web component helps to keep private data safe from attacks by using document object model to hide static data and sensitive interactive elements can be isolate within web components. Third party code runs within security context of web application this gives the full access to the application. Lack of code isolation is a serious threat. frames can support content isolation which can mitigate the attack.

Keywords: **Shadow DOM, XSS, Document isolation, Protected web components, Forgery.**

ARTICLE INFO

Article History

Received: 6th May 2016

Received in revised form :

6th May 2016

Accepted: 10th May 2016

Published online :

12th May 2016

I. INTRODUCTION

Web includes static images and document links provided by numerous service providers this gives included code full entree to the web application. Lack of code isolation is serious issue. There is risk in including remote scripts in web application which may affect site's integrity and safety of user's data. By hiding private and sensitive elements attacks on client-side content can mitigate.

JavaScript sand boxing can use to provide code isolation but it does not provide data isolation in document object model. Custom HTML tags can use in the page custom elements supports hidden DOM called shadow DOM. Functional isolation of DOM does not offer code isolation feature. There is need of providing desired level of isolation without compromising the flexibility of web application.

There is need of mechanism which supports isolation of user's data and isolation of sensitive element like input elements of login forms. Including remote scripts creates vector for targeting specific web application and also attack vector for attackers who executes low profile attacks on huge number of web applications, which yield huge sensitive information by scraping the web page's user specific content, recording input in form fields and extracting security tokens and session identifiers, etc.

The main examples of third party code inclusion are onscreen keyboard scrapping malware, malwares through advertisement. Attacker can gain client-side context through many attacks like cross site scripting, untrusted remote script, advertisement, etc. Attack may be looking for input elements of password, scraping email messages, bank statements, etc. Even if developers wisely select only reliable third parties for remote script inclusion, a certain risk exists.

There are many web applications which show user's private and sensitive information which is not isolated from the page. There is need of active isolation mechanism for application content which stops scrutiny and collection of

information by attacker. Security token can embed as a hidden form element to protect from cross site request forgery attack this increase the security level and remove the chances of attacks. The main aim is to protect client side input elements which contain the sensitive information like passwords onscreen keyboard element, attacker can easily collect this information, isolating this sensitive information can guarantee that user's input can't stole by attacker. Isolation mechanism must cover event handlers associated with isolated input elements. The insertion of untrusted third party code into a Web application is a common and possibly dangerous practice. To support the use cases for hiding sensitive information in the DOM, investigation of popular online password managers, shows DOM holds all of the user's passwords to every website.

To support the use case for protecting sensitive input elements, measurement of exposure of login forms to third-party script providers is done. There are online password managers which stores authentication credentials this highly sensitive and private information is stored in encrypted container which can decrypt at client side by providing accurate master key. Decrypted data should handle with care which may leak or may get stolen. There are many password managers which contain remote scripts from third party on the page that masses passwords in the document object model. These remote scripts have complete access to the user's credentials. Every web page has login form which is the main target of attacker to extract the user's credentials. On many sites login page suited with the remote script and remote script has complete access to the login form. The number of attacks on login forms through untrusted remote script is day by day increasing, which is a serious threat.

There are many technologies exists for the content separation from the scripts, such as document isolation, DOM separation, JavaScript sandboxing. But they have their own threats.

II. RELATED WORK

Web applications are vulnerable to many attacks, in this system shadow DOM and document isolation is provided. In document-based isolation, the frames or varying degrees depending on origins. Data is placed in the document with different origin. Isolation offers DOM-based and script-based isolation. Document isolation offers strong security for data but has rigid structure which is less attractive. For fetching the frames from unlike origin additional round trip is required which increase the page load time. JavaScript sandboxing prevent scripts from misbehaving. In this remote script is isolated in the sandbox and developers has control over its capabilities. Sandboxing technique is effective in isolating remote scripts but don't provide isolation to the parts of DOM, arises the question of security. DOM separation allows creation of certain custom HTML elements. Shadow DOM is technology that permits custom elements to hide their internal DOM structure from the outside world. Shadow DOM is motivating technology that hides content in the DOM but don't prevent later access and not provide script based isolation. For protection mechanism web components are most worthwhile starting point. They provide flexibility to cope with requirement of modern web application. Possess the capability to host separate DOM tree using shadow DOM. In protected web components static sensitive data hides in the DOM tree without being reachable by attackers. Protected web components should host interactive elements vulnerabilities of script-based compromises, such as function-overriding or prototype-poisoning attacks. Shadow DOM trees can forever hidden using ECMA Script 5 getters and different techniques can be used to isolate script code within a hidden tree. In hiding static data main task is to embed private data in DOM without exposing to attackers. Attackers can use DOM manipulation technique to extract sensitive data. Shadow DOM offers creation of separate DOM trees which are attach to traditional HTML element using shadow Root properties and during rendering process can composed into single DOM tree. Shadow DOMs and main document are functionally isolated, limits propagation of CSS selectors between the main document and the sub-trees, in both directions. Internal shadow DOM of browser is not accessible through shadow Root property but shadow DOM trees created by developer remain accessible by JavaScript. There are some conflicts with script defined shadow DOM properties of hiding static data in DOM. Developers can make their script defined DOM inaccessible to JavaScript by redefining the getter of shadow Root property.

To hide data in shadow DOM existing references are cleared and only access points are redefined, getter return null instead of reference to the shadow DOM. After redefining the getter and by rubbing all possible references to the shadow DOM, it's not possible to access the data stored in the shadow DOM. Inaccessible shadow DOM tree with sensitive data is instantiated before stuffing untrusted code guarantees that private and sensitive data will never exposed to attacker. Isolation of interactive scripts is also necessary to protect input elements of form from attackers. Input elements of form are basically depends on JavaScript handler's validation and interactive input processing.

The shadow DOM provides functional separation but does not instantiate a separate JavaScript context, JavaScript code in the shadow DOM is susceptible to many attacks like function overriding and prototype poisoning. In case of protected web components DOM's script code should be successfully isolated to avoid JavaScript variables and functions from leaking into namespace and to avoid use of contaminated functions. Obtaining script isolation in shadow DOM needs two steps, first is any code in shadow DOM should be encapsulated in distinct namespace by use of closure in JavaScript and second is preventing the use of contaminated function by storing and using known good version of the functions. Protected Web components provide a robust mechanism to isolate data and sensitive elements within the DOM tree, without losing the flexibility to place this data anywhere within the page, like iframes do. These properties guarantee that protected Web components are well suited to protect the sensitive and private data. By inserting

sensitive data in a secure Web component, using the shadow DOM we successfully avoid an attacker from taking the data in an automated way. Security tokens are embedded in forms and interactive elements, interactive elements can be protected by placing inside protected web components. Sensitive input elements is the main target of attackers, these elements can also be placed in protected web components, preventing stolen by and querying by the attacker. If input elements based on script-based handlers for validation, auto completion, the handler code must present in the protected web component. Protected web components can use to protect online password managers and login forms. Login forms and online password manager can embed in protected web components, avoiding the malicious script from thieving user's credentials through input events. Protected web components offer security benefits against realistic attackers but have narrow effect and burdens on the developer and can mitigate the attack, does not provide airtight security solution.

III. PROPOSED WORK

This paper targets to develop efficient framework which implements Web application with attack vector repository algorithm to find the attacks. Hence we are proposing an efficient framework to avoid & detecting XSS attacks which will work on DB-scan algorithm to cluster the attacks in particular category. Mainly XSS attacks are detected by using AVBS algorithm. The system overcomes the problem and limitation of the existing system.

The main objective of this system is to protect the user's private and sensitive data and to create a secured web application by using shadow DOM and document isolation. The proposed system helps to generate more secured web application than the existing web applications. It uses the information about different attacks to cluster the attacks in particular category and take action according to the attacks. The system uses hash mapping instead of array and linked list for getting immediate results and saving time also.

In the web based services three tier architecture is used. Because two tier architecture that is client-server architecture not provide the security because direct communication with the database that is between client and server. So hackers or attackers can easily access that database. So in this system three tier architecture is used. Previously attackers were easily able to inject the malicious script in the web application to stole private data or to hide some important contents of webapplication. In this system, social webapplication is created. The framework is created which can detect cross-site scripting attack and accordingly takes the action against the attack.The framework can provide mechanism for sending mail to the admin about the attacks on the web application which gives information about which element of webapplication is targetted by the attacker. According to this admin will decide whether this attack is harmful to webapplication or not so thoughtful and take the action of temporary or permanent blocking of attacker.

The architecture of this system is as follows:

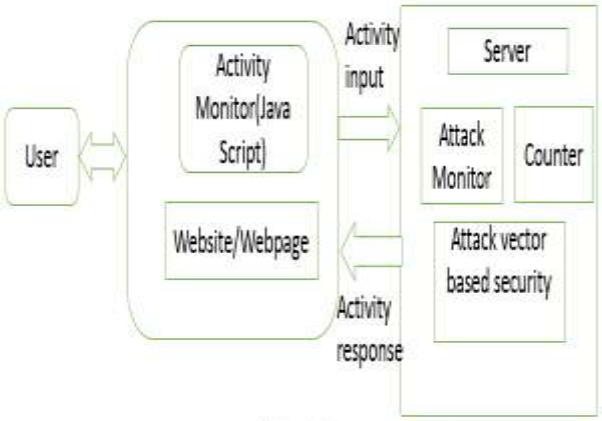


Fig. Workflow of a system

Fig 1. The architecture of this system is as follows

The system uses,

Activity Monitor: This functional block of the system always monitors the activities on web application and continuously provides the information about activities.

Attack Vector Based Security: According to activity input and input events such as on-click, on-change, on-blur, on-focus, on-delete this block will decide category of attacks and using DB-scan algorithm it will categorize the attacks.

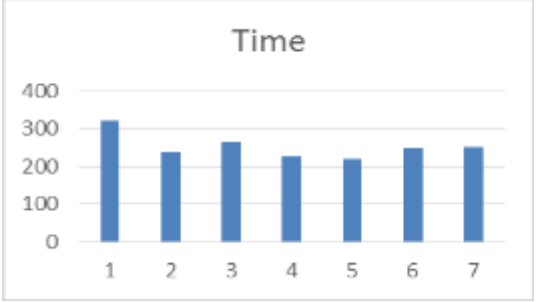
Counter: This block deals with taking the actions on attacks according to its category it will set the time to block the attacker.

DB-scan Algorithm: This algorithm is used for clustering the attacks.

IV. PERFORMANCE ANALYSIS

The following are some random values given input to AVBS mechanism and the output is plotted as the bar diagram shown next to the table:

Index	Time
1	323
2	238
3	264
4	225
5	219
6	250
7	253



Hence we can see the better recommendation results of the system than the previous systems.

V. CONCLUSION AND FUTURE WORK

This system represents the framework for preparing the protected web application. System incorporate attack vector based security algorithm for detecting and avoiding XSS and other malicious attacks. System successfully provides isolation to the user's sensitive data. Sensitive information can easily secured by hiding behind the shadows. Private data is kept inaccessible from the attackers by making its references null. System provides mechanism for continuously monitoring the web application and sending mail to the admin about the attacks. According to this admin will take action against the attack. System provides the efficient framework for securing the web applications using shadow DOM and document isolation.

In future work, the system will use artificial intelligence to cluster the new type of attack and will take action against it. In addition, the system can provide the location of the attacker or hacker. Proposed system will provide airtight security solution. The proposed framework can embed in any web application for providing the security. By use of Http Only flag for cookies many common session attacks can be prevented. The latter extension is web workers can run in the background in separate context.

REFERENCES

[1]. Detecting Cross Site Scripting Vulnerabilities Introduced by HTML5 (Authors-Guowei Dong, YanZhang,Xin Wang,Peng Wang, Liangkun Liu, 2014)

[2] Preventing Cross Site Request Forgery Attacks. (Author-NenadJovanovic, EnginKirda, and Christopher Kruegel, 2006)

[3] JSand: Complete Client-Side Sandboxing of Third-Party JavaScript without Browser Modifications (Author – Pieter Agten, Steven Van Acker, YoranBrondsema, Phu H. Phung, Lieven Desmet, Frank Piessens, 2012)

[4] You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions (Author –Nick Nikiforakis, Luca Invernizzi, AlexandrosKapravelos, Steven Van Acker, WouterJoosen,Christopher Kruegel, Frank Piessens, and Giovanni Vigna , 2012)

[5] TreeHouse: JavaScript sandboxes to helpWeb developers help themselves (Author- Lon Ingram and Michael Walfish, 2012)

