EXPLORING DOM BASED CROSS-SITE SCRIPTING

AMMAR ALDALLAL

IEEE member
Computer Enginering Department, Ahlia Univesity, Kingdom of Bahrain

Abstract- The world is familiar of web application as it is widely used throughout the world and the work of cyber crime is directly proportional of it so as the ratio of using web application is increasing the cyber crime is also increasing. Most of the users are not familiar of the methods of hacking therefore users do not know they are getting hacked by the various usage of internet. Different vulnerabilities are used by hacker when users are browsing; cross-site scripting, injection, security misconfiguration, sensitive data exposure, cross-site request forgery etc. the vulnerability cross-site scripting has three parts stored, reflected and DOM based cross-site scripting. The topic of discussion in the paper is DOM based cross-site scripting as it is used widely by the hackers. The hackers use it by sending a JavaScript or malicious code at URL when the user is browsing so it allows hacker to read the cookies and bypassed the access control of user.

Keywords- Automatic exploit generation, Cross-sites scripting, DOM-based XSS, Vulnerability detection.

I. INTRODUCTION

In 21st century the world is surrounded gies therefore people want to get connected to the world through different technologies and social websites just like facebook, Instagram, and snapchat. The web applications are developed using modern technologies are composite in nature. The web developers are implementing the applications on AJAX, JavaScript and .NET framework. While the use of web application is increasing day by day therefore the rate of cybercrime is also increasing.

Hence, it is considered as major security threat as people can access their personal data or information. The most of the breaches are done by applications thus different types of vulnerabilities are used for hacking like, Cross-site scripting, Session hijacking, Parameter manipulation, Buffer overflow, Denial of service and SQL injection. Most of the people are hacked by this type of threats as they do not understand the threat and get hacked. Moreover, around 70% of the vulnerabilities are reported through web application. It can affect business, banking and other different sectors therefore most departments used server rather than web applications. In the paper of DOM based cross-site scripting we are discussing about the protection of web applications through DOM-based cross-site scripting.

II. CROSS-SITE SCRIPTING

Many web applications are hacked through different vulnerabilities but now are discussing about DOM based cross-site scripting. In cross-site scripting (XSS) the lines of java script are injected first but if not secured against then attacker can execute malicious code on the victim machine through by utilizing insufficient support of data flowing to statements that output HTML [3]. The approach of identification and blocking the XSS vulnerabilities contains static analysis, dynamic monitoring, test generation and defensive coding. In static analysis, the exploitation of vulnerabilities occurs when there is not any concrete example and also false warnings are produced [1]. The false positives are not available in dynamic monitoring therefore it can only spot problems that are generated by input. There are two different types of test generations, Black-box test generation and White-box test generation. The white-box technique does not determine the unknown vulnerabilities and the advantages of application's internal are not used by the black-box test generation while the defensive coding contains rewriting software to make his library safe as it suffers from errors.

Cross-site scripting (XSS) threat occurs at various client-sides technologies such as HTML, JavaScript, VBscript, active x and flash when there is weakness of internet security. In XSS hacker attacks authentic web page with his malicious code therefore when user visits the web page the script is downloaded [1]. It is like a script is injected in legal ecommerce site URL therefore as the user clicks on that link the fake identical page will open and that page run a script to steal the cookies of the user using that specific e-commerce site so that hacker captures the legitimate user's session through the malicious code. The attack was not done on e-commerce site but user is tricked as the malicious code is encoded to the XSS part of URL that is not able to recognize easily. There are different types of cross-site scripting. (1) Non-persistent/Reflected cross-site scripting (2) Persistent/Stored cross-site scripting (3) DOM based cross-site scripting.

Non-Persistent/Reflected Cross-Site Scripting

The non-persistent cross-site scripting is used in HTTP parameters of query or form submission of HTML that are provided by web user. To convince the user to click on a hidden URL that contains malicious HTML/JavaScript code attackers mostly uses social engineering. Then HTML displays at user's browser and executes JavaScript that is the part of attacker malicious URL and through browser's cookies and other data can steal. The applications

need to filter input values that contain script code just because to prevent the cross-site scripting.

Persistent/Stored cross-site scripting

In persistent cross-site scripting the input of attacker is stored in a database after that insert in HTML page which is displayed to multiple users. It is not so hard to stop reflected XSS than stored XSS. It has two reasons, first it does not require the social engineering for attack although it can attack directly by supplying the malicious input that tricked the users clicking on a URL. The second is malicious script which once inserted in database it can be executed later on the browsers of users. Applications need different techniques to sanitize input that contain script code that is used in database commands. Persistent XSS occurs when the input of user is stored on targeted server like in message forum, visitor log, command field etc. the data made safe to deliver in browser without that victim is not able to retrieve the stored data. The start of HTML5 and other browser technologies, the attack is imagined that payload is stored in the victim's browser permanently and will not be sent to server at all [8].

III. DOM BASED CROSS-SITE SCRIPTING (XSS)

The document object mode (DOM) is the type of cross-site scripting which occurs due to data related with DOM present in HTML page. It is used for retrieving and operating objects in HTML document. The vulnerabilities of all types are originating by insecure user in the DOM based cross-site scripting. By using properties of DOM the entities of HTML can be retrieved and personalize such as document.referrer, document.url and document.location [7]. The logic depends on web application's program therefore at some point in time manages a string-to-code conversion and takes inputs that are controlled by hackers. The malicious script does not get to web server in DOM XSS type of attack that only execute at user side. The methods eval(), document.write or innerHTML are used when DOM based XSS occurs and client sends untrusted data to interpret as JavaScript. The functions InnerHTML and eval are widely used in DOM based XSS.

Eval

Eval function is widely used in web applications. As it is discussed above DOM based XSS uses HTML parser and JavaScript engine to intercept while in the DOM based XSS required to go through the HTML parser. The execution will never pass the HTML parser when the website invokes the java script function eval with user-provided data therefore a malicious payload is not shown when attacker injects in call to eval.

innerHTML

There is possibility in execution JavaScript via inline event handler when the execution of script tags through innerHTML is not done. Thus the innerHTML is also horizontal to XSS attacks. The filtration of content parsed through innerHTML is done in past versions. Some drawbacks in performances of innerHTML heavy applications experienced by Google as well.

The code is the main reason for these types of attacks in web applications. The hacker can insert his own code in application when the input is not sanitized properly. The subtypes of DOM-based XSS used the methods for translating the string to code are HTML context, JavaScript context and URL context.

A. HTML context

The general HTML code is inserted in the DOM through functions like document.write inner HTML and insert adjacent HTML. The functions are called to resolve the string parameter and convert the content as HTML code and then it is inserted to a DOM XSS at a particular position. The encoding functions and sanitization is used to avoid code inserted vulnerabilities. There is an opportunity for attacker to insert own html tags when the input is not sanitized correctly, that include JavaScript execution. The reaction of browser in terms of script execution is different as compared to document.write and inner, outer or adjacentHTML. The innerHTML only performs the first step of inserting script but does not execute into the DOM while document.write inserts and carry out the script immediately from DOM. The innerHTML flow is not successful in utilization but the scripts can be executed via an injection by innerHTML. When the img tag is inserted by the innerHTML, the HTTP request is created by browser to the non-existing resource.

JavaScript context

The eval function is vulnerable to DOM-based XSS that takes the strings parameter and convert it as JavaScript code and executes it. There are some other contexts besides eval and its aliases which are used to convert JavaScript code such as scipt.innerText, script.text, script.textcontent and the strings related to assignment to event handler attributes. If there is mistake in parsing and implementing of HTML browsers will forgive that but it is very strict in JavaScript code execution. If there is a syntax error in JavaScript the execution for complete function is cancelled. Therefore, it is required that exploit generator have to generate a syntactically correct exploit; hence, it will not be cancelled by parser at execution of function. Moreover, the exact location of tainted bytes is determined by the system.

URL context

The input flows in URL attribute of any DOM node i.e img.src, iframe.src, object.data or a href which are controlled by attacker therefore string to code does not occur immediately. Moreover, in these flows, there are still different types of security problems are located [5]. When the complete URL is controlled by the attacker, he can easily make full use of JavaScript or schemes of data to execute the script code but if

only some parts of URL are controlled, it can also be executed as there is very interesting coding pattern in the dataset. Web page must assign a partly string to script.src that includes a JSONP script with a call back parameter for abusing the patterns of this type.

Other Context in DOM based XSS

There is some other context which allows the execution of code that are security sensitive like, document. Cookie, the web storage API, post message or set attribute. Our systems also captured the flows in cookie and in web storage API. Although, it does not include the automatic exploit generation but it can find several persistent manually. The flows is detected of user's input and then went into cookie or web storage sinks successfully and persisting the code or data to the browser at user side. The data used in a call or eval when it is able to trigger the successful exploit, thus it exposed the web site to persistent DOM-based XSS.

IV. VULNERABILITIES DETECTION

There is system which offers to automatically distinguish and corroborate DOM-based XSS vulnerabilities. Due to challenges in assessment of user side security problems it deals to integrate our techniques directly in full browser rather than building analytical process that complement and emulate the client-side behavior [2]. The systems consist of two separate components for vulnerability detection. The modified browsing engine is used to support dynamic and byte-level suspicious flows. It can achieve complete coverage of features of all JavaScript language and full DOM API. Although, in second component there is automated vulnerability validation mechanism that influence the information of fine grained context provided by the browsing engine of taint aware [9]. The system is able to attack at payloads that matches the syntactic surroundings on injection point because accurate information of data source and syntactical context of final data sink. Moreover, it allows the unambiguous vulnerability validation through verification that JavaScript was executed.

Modified Chrome

The dynamic taint tracking approach is used to automatically detect the flow of attacker controllable input into a sink in the sense of DOM-based XSS. The edge cases are ensured that are not implemented correctly into pure testing engine just as HTML unit, it decided to apply taint-tacking into real browser for properly execution of edge cases [6]. Therefore, the open-source browser Chromium is modified in such a way that both the JavaScript engine V8 and DOM implementation in Web Kit are enhanced with taint-tracking capabilities. The byte-wise taint-tracking approach is selected for both the components that built directly into string representations. In this method, the tool not only distinguishes between fully untainted string and potentially harmful content string, but gets the information of each given character in string exclusively.

V8 JavaScript engine

The java script engine v8 is effective in memory allocation and execution of speed. When representing JavaScript objects like strings or arrays, V8 for maximum parts does not make use of a class concept or use of member variable, while the code is written in C++ language. Therefore, objects are addressed by using specific offsets related to the address of object. In addition, small header is used therefore after checking the given code we only chose to encode the desired data into the header [2]. Every object in V8 stores a pointer to the map and the class of an object is described by map. There are different types of maps in V8 for specific object thus the used part of bitmap is found and used to make new map objects for tainted strings. Additional memory is given to store the actual data for strings of dynamic length. The additional memory is allocated on creation of object based on string is pure ASCII or contain two-byte character. The address of newly made space and information that a string is tainted are mentioned to one of the abovementioned offsets in the header therefore it also need to save the taint bytes. Hence the change is required for string implementation if the additional bytes are allocated. The taint bytes are stored in the last part of allocated memory because there should be very little changes in existing code. Although, the functionality for taint data access is added and the functionality for normal access to a string's character does not require any change [10]. The V8 engine is optimized for performance therefore assembler code directly written from macros. Simple operations known as string allocation are done without using complex run time code of C++ language. For easily mixing in the existing code it decides to disable the optimization for creation or sub-string access and other string operations as well. First the patching of string implementation is done than the string propagation functions are instrumented for functions that are substring, concat, charAt etc. Hence it assures that during string conversation the byte-wise taint tracking information is also propagated.

Web kit JavaScript engine

In comparison of V8 engine common use of member's concept for its classes are made by webkit. There is addition of such member that denotes that string is tainted or not just because the detection of a tainted string is allowed. The array is used to store the character data by the string implementation of Web Kit [10]. Therefore, a second array is used to hold the taint bytes. The strings are converted that came from V8 just before writing into the DOM therefore to allow the propagation of taint information functions are patched. The tainted data might be stored temporarily in the DOM before flowing to a sink. The example is like first the setting through href of anchor then using it in a document.write. It is required to

change the string implementation and also to modify the HTML token just for correct propagation of the taint information. If the HTML content is set through JavaScript by using innerHTML so that it parsed and spilt up into its tree structure and stored as string also.

V. AUTOMATIC EXPLOIT GENERATION

The taint tracking engine gives first notification for cross-site scripting vulnerabilities, therefore it is not ufficient to ensure that vulnerability was discovered by detecting the flow single handedly. The successful exploitation is not possible for an existing flow and there are reasons for it like the built-in or filter functions are used by websites that can defuse a malicious payload while an exploit could stop from executing when other random circumstances occurs suddenly [4]. The tainted value generated from GET parameter, the web server is triggered due to tampering with the parameter and load a different page or show the message of error when the flow is not present anymore. Hence the verification is required to inform vulnerable data flows apart from non-exploitable flows. The system takes data that generate valid cross-site scripting exploits from the taint tracking.

CONCLUSION

The rate of cyber crime is increasing very rapidly so different vulnerabilities are used in which cross-site scripting is one of that. Cross-site scripting hackers can insert JavaScript lines or malicious code to web page and as user visit that page the script is downloaded and the attack is done on the XSS of URL. Different types of cross-site scripting are used like reflect/persistent cross-site scripting, stored/non-persistent cross-site scripting and DOM based cross-site scripting. The DOM based cross-site is used to operate and retrieve the objects in HTML object. Different methods are used in DOM based cross-site scripting such as eval() document. Write or inner HTML only when the client sends unreliable data just to interpret the JavaScript. There is different system to make a distinction and confirm the different DOM based vulnerabilities. While different vulnerability reorganization methods are discussed in the paper like Modified chrome and automatic exploit generation. Therefore in (modified chrome) method of vulnerability detection JavaScript engine v8 and web kit DOM implementation are used to enhance the taint-tracking capabilities while in (automated exploit generation) method of vulnerability detection a system is used which take the various information and generate cross-site scripting exploits from traint-tracking. There are various types of hacking criteria or methods are used by hackers in which cross-site Scripting is one of them and is completely described in the paper so the user can get familiar of various methods so that he becomes aware while browsing just to prevent hacking.

REFERENCE

[1] Van Rooij, A. J., Schoenmakers, T. M., Van de Eijnden, R. J., & Van de Mheen, D. (2010). Compulsive internet use: the role of online gaming and other Internet applications. Journal of Adolescent Health, 47(1), 51-57.

[2] Akhawe, D., Barth, A., Lam, P.E., Mitchell, J. and Song, D., 2010, July. Towards a formal foundation of web security. In Computer Security Foundations Symposium (CSF), 2010 23rd IEEE (pp. 290-304). IEEE.

[3] Gupta, S., & Gupta, B. B. (2015). Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. International Journal of System Assurance Engineering and Management, 1-19.

[4] Johns, M. (2011). Code-injection Vulnerabilities in Web Applications—Exemplified at Cross-site Scripting. it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik, 53(5), 256-260.

[5] Van Gundy, M., & Chen, H. (2012). Noncespaces: Using randomization to defeat cross-site scripting attacks. Computers & Security, 31(4), 612-628. Data retrieved from: http://web.cs.ucdavis.edu/~hchen/paper/cs2012.pdf on 20th April, 2017

[6] Shar, L.K. and Tan, H.B.K., 2012. Defending against cross-site scripting attacks. Computer, 45(3), pp. 55-62.

[7] Yusof, I., & Pathan, A. S. K. (2014, November). Preventing persistent Cross-Site Scripting (XSS) attack by applying pattern filtering approach. In Information and Communication Technology for The Muslim World (ICT4M), 2014 The 5th International Conference on (pp. 1-6). IEEE.

[8] Baojiang, C., Baolian, L. and Tingting, H., 2014, November. Reverse analysis method of static XSS defect detection technique based on database query language. In P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2014 Ninth International Conference on (pp. 487-491). IEEE.

[9] Stock, B., Lekies, S., Mueller, T., Spiegel, P., & Johns, M. (2014, August). Precise Client-side Protection against DOM-based Cross-Site Scripting. In USENIX Security (pp. 655-670). Data retrieved from: https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-stock.pdf on 20th April, 2017

[10] Gupta, S., & Sharma, L. (2012). Exploitation of cross-site scripting (XSS) vulnerability in real world web applications and its defense. International Journal of Computer Applications, 60(14).

★★★