A Survey on Web Application Vulnerabilities

[1]D. K. Patil, [2]Kailas. R. Patil

[1,2]Computer Engineering, VIIT, Pune Maharashtra, India, SPPU University
Email: [1]dnyaneshwar11.patil@gmail.com, [2]kailas.patil@viit.ac.in

Abstract — Web applications are becoming popular and ubiquitous in our daily lives. The web applications are used for multiple purposes such as e-commerce, financial services, emails, healthcare services and many other captious services. Multiple web browsers are available for accessing these web applications. Due to presence of vulnerabilities in web applications attackers can take benefit by stealing sensitive information of the web user or the organization. However modern web browsers do not provide any notification, alerts or indications of security holes or vulnerabilities in the web applications. This may deface these web applications. In this paper we survey the state of the art web application vulnerabilities; First we explain working of the web application then different types of vulnerabilities present in real world web applications and we focus on the Cross-site Scripting vulnerabilities (XSS) and SQL Injection attacks on the web applications, how the web application compromised by the attacker for breaking security of the web application and after that various approaches for the detection of the web application vulnerabilities. Finally we summarize the topics and discussed future scope and opportunities in this area.

Keywords— Web Application, Cross Site Scripting(XSS), SQL Injection, vulnerabilities.

I. INTRODUCTION

Web application is the software that is able to run in web browser. Such web applications can be developed with the help of programming languages (for example: HTML, CSS and JavaScript etc.) that are supported by the web browser. These programming languages rely on the web browser for rendering web applications. Due to the ubiquitous nature of the web browser web applications are becoming more popular. Another reason for popularity of the web application is its attractive graphical user interface. The main reason for becoming popular of the web application is that to maintain its updatability excepting the trouble of installing the software on strongly millions of web client computers. Web application borrows itself towards multi-tiered perspective by its occurrence. Generally 3-tiered approach is followed by the web application developers and it consists of initial tier as a web application browser intermediate tier as a application tier that include business logic and last tier is for the storage purpose. Initial tier will any type of the web browser like Mozilla Firefox, Safari, Google chrome and many other web browsers, and by using technologies like ColdFusion, JavaScript, Ruby, Struts, PHP, Perl, Python for developing the business logic it becomes intermediate tier and the last tier is nothing but the store that will handle databases of the web applications. In this phenomenon initial tier generate request and sends it to the intermediate tier that facilitates it by generating queries as well as updates for the database management system and creates a user interface. This 3 – tier architecture will not more effective for the complex web applications therefore for handling this problem web developers are preferring multi-tiered perspective.

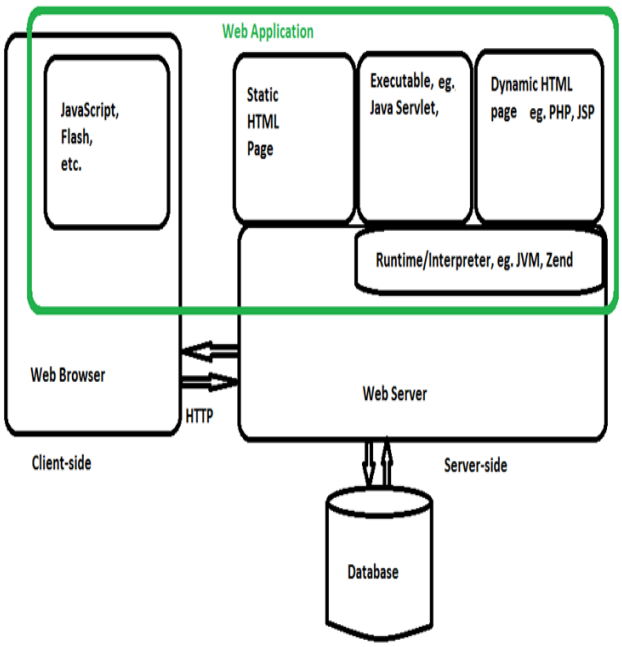

Figure 1.Web application working scenario

As shown in Figure 1 web application works in the client-side mechanism and server-side mechanism. Client-side mechanism can be used by web browser for rendering web application. It may contain JavaScript, Flash etc. and by using this Client-side mechanism user can use web browser for searching the content on web or to do his intended work. Web user may use multiple web browsers like Mozilla Firefox, Google chrome, Safari

and many more for making request to the web server [18]. Web browser works as the interface between web application user and web server. Web user enters URL into address bar of the web browser for making request to the web server or web user can use search engine for making request to the web server and using web application. In between web browser and web server once web user enters keyword into the search engine at that time web browser generates HTTP request and sends it to the web server. Here security of the generated request depends on the HTTP headers used by the web application developers as well as policies used by the web application developers. So it is necessary to focus on the Client-side mechanism to make stronger protection for the web application to save important data from cyber criminals.

Continuous growth in web application development without considering its vulnerable status is important factor for web security. Web application developers are mostly focusing on good user interface, runtime facility and user friendliness of web applications. But vulnerabilities present in web application may cause serious issues like stealing sensitive information of web users and cyber attacks as Cross-site Scripting (XSS), SQL Injection attacks, Information leakage, Cross-site Request Forgery (CSRF) and many vulnerability related attacks.

Web application is useful for e-commerce services, financial organizations, governmental websites and social media like Facebook, twitter etc. all these service providers and information users require online security for their important information but due to different types of vulnerabilities present in web application attacker can easily access this valuable information of user. According to Survey[17] conducted by Cenzic Inc. 96% of tested web applications in 2013 have at least one or more serious security vulnerabilities. The application layer is continuously targeted by attackers as soft way for attack.

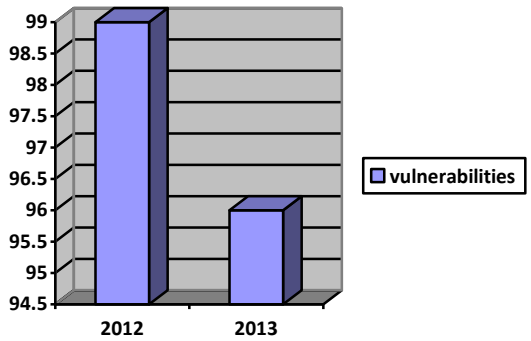

Figure 2.Percentage of tested apps with vulnerabilities

As shown in figure 2, 99% of vulnerability found in their tested application in year 2012 and 96% of vulnerability found in year 2013. A median of these vulnerabilities per web application is 13 for year 2012 and 14 for year 2013 respectively. Cross-site Scripting is the topmost vulnerability among web applications. Most of the web applications are vulnerable due to unawareness of web application developers about secure practices. Web browser works as an interface between web client and web server. Current browsers are having extensions for detecting specific vulnerability attack but none of the browser having all in one solution for the detection of all these web vulnerabilities.

In remaining sections of this paper, Section II describes different types of web application vulnerabilities, the client side mechanism for web application vulnerabilities and server side mechanism for web application vulnerabilities, Section III describes research works on web application vulnerabilities and their preventive measures, Section IV describes our observations on this entire topic of the web application vulnerabilities and Section IV describes conclusion about this topic of web application vulnerabilities.

II. WEB APPLICATION VULNERABILITIES

A. Injection Vulnerabilities

Injection vulnerabilities are the techniques used to attack applications of the type data-driven [22]. In injection vulnerabilities malicious executable statements are inserted into an entry field for the execution. Basically there are four types of the injection attack categories as given below.

1) LDAP Injection:

It is a type of attack used for exploiting web based applications which builds LDAP statements on the basis of the user given inputs. Whenever web application becomes unable to appropriately sanitize web user input, there is chance of modifying the LDAP statement using the local proxy. This may become cause for executing commands by unauthorized queries and change on the actual content of the main LDAP tree.

2) XPATH Injection:

This attack can be possible when web application uses information provided by the web application user for representing an XPath query for XML data. By sending malicious information to the web application the cyber attacker can recognize what is the structure of the XML data or can get access to the data for which he is not authorized. He can also raise his privileges for the web application and will that XML data and utilized it for the authentication purpose.

3) Format String Injection:

Format string injection can occur when the inserted data of the input string is executed as command by the web

application. By using this method attacker can read the stack, able to execute the code, or may become cause for the web application as segmentation fault.

TABLE I. PARAMETERS USED FOR ATTACK

Parameters	Output	Passed as
%%	% character (Literal)	reference
%p	External representation of a pointer	reference
%d	Value in decimal	value
%c	Character	
%u	Value in unsigned decimal	value
%x	Value in hexadecimal	value
%s	Reference of the string	reference
%n	Writes the number of characters into a pointer	reference

4) Command Injection:

Command injection attacks are having aim of executing random commands on the operating system of the host machine by the vulnerable web application. These command injection attacks can occur when a web application forwards unsecure user inputs towards the system shell. Command injection attacks are mostly occurs due to insufficient input validation.

B. Cross-site Scripting Vulnerabilities

Cross-site Scripting vulnerabilities are common vulnerabilities in most of the web applications. There are three types of Cross-site Scripting vulnerabilities. In Cross-site Scripting attacker uses malicious scripts for exploiting web applications and breaking their security environment. Cross-site Scripting attacks are possible by using web technologies like VBScript, JavaScript, HTML, ActiveX, Flash and other client-side programming language. These attacks are capable of gathering of data from account hijacking, modification of the user settings, theft poisoning or wrong advertising. Cross-site Scripting is also capable to attack on web application server which may become cause for denial of service attacking. Following are the types of Cross-site Scripting attacks:

1) Stored XSS Vulnerabilities:

Stored Cross-site Scripting vulnerability is the most powerful type of the XSS attack. When web application user provides information to the web application and that information is stored permanently on the server and later displayed on the webpage of the web application without encoding it with entity encoding of the HTML language. Stored XSS vulnerability is also known as second order vulnerability [20]. A real world example of this vulnerability is Samy Myspace Worm. Example of stored XSS vulnerability is:

Suppose there are two categories of users, executive user and general user. When executive user log-in, he will be authorized for looking list of all general users and when general user log-in he or she can update their display name.

1. This is login_page.php

```
<?php
$Host= '192.168.4.2';
$Dbname= 'abc';
$User= 'aaa';
$Password= 'xxx';
$Schema = 'test1';
$Conection_string="host=$Host dbname=$Dbname
user=$User password=$Password";

/* Connect with database asking for a new connection*/
$Connect=pg_connect($Conection_string,$PGSQL_CO
NNECT_FORCE_NEW);

/*For error checking of the connection string */
if (!$Connect) {
 echo "Failure in database connection ";
 exit;
}

$query="SELECT user_name,password from
$Schema.members where
user_name='".$_POST['user_name']."';";

$result=pg_query($Connect,$query);
$row=pg_fetch_array($result,NULL,PGSQL_ASSOC);

$user_pass = md5($_POST['pass_word']);
$user_name = $row['user_name'];

if(strcmp($user_pass,$row['password'])!=0) {
 echo "Failed login";
}
else {
 # Start the session
 session_start();
 $_SESSION['USER_NAME'] = $user_name;
 echo "<head> <meta http-equiv=\"Refresh\"
content=\"0;url=home.php\" > </head>";
}
?>
```

2. This is homepage.php

```
<?php
session_start();
if(!$_SESSION['USER_NAME']) {
 echo "Should login";
}
else {
 $Host= '192.168.4.2';
 $Dbname= 'abc';
 $User= 'aaa';
 $Password= 'xxx';
 $Schema = 'test1';
```

```
 $Conection_string="host=$Host
dbname=$Dbname user=$User password=$Password";

$Connect=pg_connect($Conection_string,$PGSQL_CO
NNECT_FORCE_NEW);
 if($_SERVER['REQUEST_METHOD'] == "POST") {
 $query="update $Schema.members set
display_name='".$_POST['disp_name']."' where
user_name='".$_SESSION['USER_NAME']."';";
 pg_query($Connect,$query);
 echo "Updated Successfully";
 }
else {

if(strcmp($_SESSION['USER_NAME'],'admin')==0) {
   echo "Welcome executive<br><hr>";
   echo "List of General user's are<br>";
   $query = "select display_name from
$Schema.members where user_name!='executive'";
   $res = pg_query($Connect,$query);

while($row=pg_fetch_array($res,NULL,PGSQL_ASSO
C)) {
   echo "$row[display_name]<br>";
   }
 }
else {
echo "<form name=\"tgs\" id=\"tgs\" method=\"post\"
action=\"homepage.php\">";
echo "Update display name:<input type=\"text\"
id=\"disp_name\" name=\"disp_name\" value=\"\">";
echo "<input type=\"submit\"   value=\"Update\">";
     }
}
    }
   ?>
```

When attacker will log-in, he will insert following script into the login_page.php

```
3.          <a href=#
onclick=\"document.location=\'http://xyz.com/xss.php?c
=\'+escape\(document.cookie\)\;\">U r attacked</a>
```

The above mentioned information will be stored permanently in the database.

When executive user log-in, he will see list of general user names with the link " U r attacked" . When executive user click the link it will send session ID of the executive user to the attacker through the cookie. The cookie information will look something like:

```
4.          xss.php?c=PHPSESSID%5Dxjz6s2d5f1r1gv
myteru7iossdferttresy6ryt8sjllf9hfd4
```

So once attacker knows this session ID of executive user he can use until that session ID expires.

2) Reflected XSS Vulnerabilities

When data provided by the web application user is used for reflection by the server according to the requested web page for generating the expected result then this mechanism can become source for reflected XSS type of the vulnerability. It can be used for denial of service attacking. For example consider the following example:

By using 'meta' tag .php page can be reloaded

```
<META HTTP-EQUIV=Refresh CONTENT="1;
URL=http://www.ursite.com/ururl.php">
```

As shown in above script in php, particular page will be refreshed after each second. So it will become as an infinite loop for refresh requests which will cause database server down due to flooding of requests. In this way denial of service attack may happen on web application.

3) DOM-based XSS Vulnerabilities

DOM-based XSS vulnerabilities can occur in page's client-side script itself. Suppose JavaScript accesses a URL request parameter and takes that information to write some HTML to its own page which is not encoded using HTML entities then DOM-based XSS vulnerability will be present there and this written data will be reinterpreted by web browsers that can include additional client-side script. For example:

We have web application as given below

```
http://yourwebsite.com/entity5/worldencapsultaed/forige
ndata/returnpage11.php
```

attacker can write code for DOM-based attack for above mentioned url as

```
http://yourwebsite.com/entity5/worldencapsultaed/forige
ndata/returnpage11.php?<script>alert('DOM-
BASED_XSS_ATTACK')</script>
```

so above code will generate DOM-based XSS attack for that particular webpage.

III. RELATED WORK

1..Authors C. Yue and H. Wang have a paper [1] in this paper they have analyzed insecure JavaScript practices and suggested alternative JavaScript practices for it[1].

They have studied and analyzed three main functions of JavaScript as given below:

1. eval()
2. document.write() method
3. innerHTML property

They examined 6805 unique websites for the measurement and analysis of JavaScript. They emphasized JavaScript because JavaScript is an interpreted programming language and mostly used for enhancement of the interactivity and functionality of web pages and it also has attractive features to interact with web browser windows as well as webpage documents. According to their analysis they found 66.4% of analyzed websites convicts unsafe practices with inclusion of JavaScript into the top level documents of their web-pages. 44.4% of their measured websites used eval() function for dynamic generation and execution of JavaScript codes in their web pages. And they also found the function document. write() of the JavaScript and property of inner HTML are very popular instead of alternative secure practices for them. But they have specific solution for avoiding web application vulnerabilities that are related to avoiding insecure JavaScript practices.

2. Authors M. Cova, C. Kruegel and G. Vigna have a paper [2] which presents the solution for detection of the attacks which are possible due to execution of the online downloaded files. They have presented a unique approach for the detection and for analyzing for harmful JavaScript code in the web applications. Their Approach integrates abnormality detection with endeavor to automatically understand harmful JavaScript code and to support for the decomposition of it. For implementation they have developed a system that uses machine learning techniques and number of features to establish the features of usual JavaScript code and their system is also capable to detect behavior of abnormal JavaScript code by imitating its behavior and equating it to launch prominence [2]. This solution presented by authors cannot protect web applications from JavaScript malwares.

3. Authours Canali D., Cova M., Vigna G., Kruegel C. have a paper [3] in this paper they have explained concept of the attacks that are happening at the time of downloading and prevention techniques for it. In drive by download mechanism attacker attaches executable files of scripts with downloadable files due to which attacker can take control of the victims system[3]. For preventing drive by download attacks they have build a filter named as 'Prophiler' which is used for detection of the harmful web pages. This filter filters web application due to which number of pages can be reduced that is essential for analysis with runtime to recognizing harmful web pages. For demonstration and efficiency of the Prophiler they crawled millions of web pages for which they analyzed malicious behavior. This is nothing but the filter, not the detector for vulnerabilities it means it does not provide protection against web vulnerabilities.

4. Authors Falk L., Prakash A. and Borders K. have presented paper [4] which proposes solution for analysis of websites for the design flaws that are visible to user. User visible security design flaws may contain flaws that can become risk for web user. Further they examine that the influence of user visible security by examining websites from 214 United States commercial institutes. They intentionally chose commercial web applications because of their high demand for security [4]. After experimentation they found lot of faults which may direct web clients to make worse security permissions. According to their survey 76 percent of their examined websites containing minimum one design fault which indicates that these design flaws have not understood widely even experts who have information about security and responsibility of security. Therefore finally they implemented solution to recover from these security design flaws which are user visible design flaws. This paper detecting only design flaws which are user visible. **5.** Authors M. Finifter and others introduced special solution for preventing capability leaks of the subsets of JavaScript in their [5]. In this paper they proposed new technique for preventing capability leaks of JavaScript by improving statically verified JavaScript subset[5]. They explained about one-third of Alexa Top 100 web applications are exploitable by an advertisement by the ADsafe which is verified. They proposed an updated mathematically verified subset of the JavaScript which uses namespaces. Their approach consists the well presentation and the capability to perform well of mathematically static identification at the time of improving security. It is only possible to prevent web application from capability leaks of JavaScript codes it means authors have considered only capability leaks problem of JavaScript.

6. Authors D. Florencio and C. Herley presented paper [6]. This paper explains study conducted for password habits of web users and gives protection to the password given by user for his system and which stored on web browser [6]. This system is having client-side approach but related to protection of passwords that are stored on web browsers.

7. Authors S. Kals, E. Kirda, C. Kruegel, and N. Jovanovic presented [7] which is having explanation for how to find potentially vulnerable websites. They mentioned many web application security holes are the result from generic validation problem of the input. They have mentioned about loss of important information of the credit card belonging to number of credit card users. According to their view CSS vulnerability and SQL Injection vulnerability are major type of vulnerabilities in web applications. By usefulness of the SecuBat they were able to detect many potential vulnerable websites and for validating the performance and accurateness of the SecuBat they picked 100 interesting websites from the potential list of victim for the purpose of further analysis as well as to confirm exploitable flaws in the recognized web pages and they also mentioned all of their victims were from well known industrial companies and of a vulnerable web sites about possible security problems. The only

limitation of this proposed solution is we have to submit websites to this scanner means it is not based on Client-side approach.

8. Authors Shuo Chen, and others have research work over the topic of security flaws in GUI logic and published their work in the paper [8]. As per their perspective for achieving security at the end point, conventional security techniques are incapable if the integrity of HCI is compromised by third party. GUI logic flaws are included in the classification of the vulnerabilities that are the outcome from logical errors in GUI implementations [8]. These GUI logic flaws can be attacked by visual spoofing attacks and these visual spoofing attacks can lure to security expertise or security conscious users for performing unintended activities. Their focus is on to formulating the problem of logic flaws of the GUI and to produce a solution for recovering logic flaws of the GUI. Basically, by the help of an thorough study of main subparts of IE browser implementation code, they have constructed the model to browser logic of GUI and they have applied informal reasoning to detecting novice spoofing mechanisms with the consideration of four for spoofing of bar by address and nine for spoofing of the bar by the status. The team of developers of IE have already confirmed mechanisms and has fixed. In this paper they have demonstrated a member set of spoofing vulnerabilities by visually originated from logic faults of GUI. Authors are totally focusing on the vulnerabilities which are only related to GUI logic means they have implemented their solution with the specific consideration of the problem.

9. Authors R. Zhao and C. Yue have worked for the topic which is related to browser saved passwords in their [9]. In their work they mentioned problems related to web users pass-words. According to their perspective web application users are facing problems with the intimidating challenges of forming, memorizing and using safe as well as strongest passwords for maintaining their important assets on respective web applications. They have also mentioned the five popular browsers are providing password managers as a inbuilt functionality but, regrettably design of all these five web browsers password managers have severe vulnerabilities inside them. They have presented a novel approach for detecting vulnerabilities in web browsers password managers and it is based on cloud computing and their proposed system is known as Cloud-based Storage-Free Browser Password Manager(CSF-BPM)[9] which is designed for maintaining integrity, confidentiality as well as availability properties. They have implemented their proposed system into the Firefox web browser with the evaluation and its accuracy and functional behavior. They have suggested that their system can be implemented in other global browser. They have implemented different approach for the protection of browser saved passwords rather than the conventional password manager systems.

IV. OBSERVATION

Recent web applications are consisting very complex structures but due to security loopholes inside these structures, they are prone to various web application vulnerabilities. Web development teams are not aware about secure web development practices and they are developing web applications without considering the security factor.

V. CONCLUSION

Solutions for web application vulnerabilities are specific for particular vulnerability and applicable to particular web applications. Solutions that are applicable to the client-side are breaking security at the server-side and vice-versa. So for providing robust security to web applications there is the need of generic solution that can detect all kinds of web application vulnerabilities in the web application.

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to the administration of Department of Computer Engineering of VIIT, Pune, India and SPPU University for providing the academic environment to pursue research activities. In particular I would like to thank Dr. K. R. Patil, Associate Professor (VIIT, Pune) for his invaluable guidance.

REFERENCES

[1]. Yue, C. and Wang, H. 2013. A measurement study of insecure JavaScript practices on the web. ACM Transactions. Web 7, 2, Article 7 (May 2013). DOI: http://Dx.Doi.Org/10.1145/2460383.2460386.

[2]. M. Cova, Kruegel, G. Vigna, 2010. Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code. In Proceedings of the International Conference on World Wide Web(WWW). 281-290.

[3]. Canali, D., Cova, M., Vigna, G., And kruegel, C. 2011. Prophiler: A fast filter for the large-scale detection of malicious web pages. In Proceedings of the International Conference on World Wide Web (WWW).197{206 , Jan. 2012.

[4]. Falk, L., Prakash, A., And Borders, K. 2008. Analyzing websites for user-visible security design flaws. In Proceedings of the Symposium on Usable Privacy and Security (SOUPS). 117{126}.

[5]. Finifter, M., Weinberger, J., And Barth, A.. Preventing capability leaks in secure JavaScript sub-sets. In Proceedings of the Network and Distributed System Security Symposium (NDSS) 2010.

[6]. D. Florencio, C. Herley, 2007. A Large Scale Study of Web Password Habits. In Proceedings of the International Conference on World Wide Web(WWW). 657-666.

[7]. Kals, S., Kirda,E., Kruegel,C., And Jovanovic, N. 2006. SecuBat: A web vulnerability scanner. In Proceedings of the International Conference on World Wide Web (WWW). 247{256}.

[8]. Chen, S., Meseguer, J., Sasse, R., Wang, H. J., and Wang, Y. M. 2007. A systematic approach to uncover gui logic flaws for web security. In Proceedings of the IEEE A Symposium on Security and Privacy. 71-85.

[9]. Zhao, R. And Yue, C. 2013. All your browser-saved passwords could belong to us: A security analysis and a cloud-based new design. In Proceedings of the ACM Conference on Data and Application Security and Privacy (CODASPY).

[10]. Ball, T. and Larus, J. R. 1994. Optimally profiling and tracing programs. ACM Trans. Program. Lang. Syst. 16, 4, 1319–1360.

[11]. Barth, A., Jackson, C., and Mitchell, J. C. 2008a. Robust defenses for cross-site request forgery. In Proceedings of the ACM Conference on Computer and Communications Security (CCS). 75–88.

[12]. Barth, A., Jackson, C., and Mitchell, J. C.. 2008b. Securing frame communication in browsers. In Proceedings of the 17th USENIX Security Symposium. 17–30.

[13]. Baxter, I. D., Yahin, A., Moura, L., Santanna, M., and Bier, L. 1998. Clone detection using abstract syntax trees. In Proceedings of the International Conference on Software Maintenance.

[14]. Curtsinger, C., Livshits, B., Zorn, B., and Seifert, C. 2011. Zozzle: Low-overhead mostly static JavaScript malware detection. In Proceedings of the USENIX Security Symposium.

[15]. Egele, M., Wurzinger, P., Kruegel, C., and Kirda, E. 2009. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. In Proceedings of the Annual Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA). 88–106.

[16]. Dileep Athavale, 2014. Online retail, finance firms care less for client data safety : Survey. In SUNDAY TIMES OF INDIA, Pune. 8.

[17]. Survey by Cenzic Inc. Application Vulnerability Trends Report . Vulnerability Report 2014.

[18]. https://cirt.net/Nikto2

[19]. https://addons.mozilla.org/en-US/firefox/extensions/

[20]. https://www.owasp.org/index.php/DOM_Based_XSS

[21]. http://www.xssed.com/xssinfo

[22]. http://www.sans.org