Quantitative Assessment on Broken Access Control Vulnerability in Web Applications

M. M. Hassan[1], M. A. Ali[1], T. Bhuiyan[1], M. H. Sharif[1], and S. Biswas[1]

[1] Daffodil International University, Dhaka/Bangladesh, maruf.swe@diu.edu.bd
[1] Daffodil International University, Dhaka/Bangladesh, asraf.swe@diu.edu.bd
[1] Daffodil International University, Dhaka/Bangladesh, t.bhuiyan@daffodilvarsity.edu.bd
[1]Daffodil International University, Dhaka/Bangladesh, sharif.swe@diu.edu.bd
[1] Daffodil International University, Dhaka/Bangladesh, saikatbiswas440@gmail.com

***Abstract* - Broken Access Control (BAC), ranked as 5th crucial vulnerability in Open Web Application Security Project (OWASP), appear to be critical in web applications because of its adverse consequence i.e. privilege escalation that may lead to huge financial loss and reputation damage of the company. The intruder of a web system can get an unauthorized access or upgraded access level by exploiting through the BAC vulnerability due to inadequate validation of user credential, misconfiguration of sensitive data disclosure, inappropriate use of functions in the code, unmanaged exception handling, uncontrolled redirection of webpage, etc. This paper presents the awareness regarding the risk for the existence of BAC vulnerability in the web application to its designer, developer, administrator, and web owner considering the facts and findings of the document before hosting the application on live. The experiment was conducted on 330 web applications using manual penetration testing method following double blind testing strategy where 39.09% of the sites were found vulnerable with the same. Access on redirection settings, misconfiguration of sensitive data retrieval, and unauthorized cookie access exploitation techniques performed on the sample sites among five sectors analyzed based on the reason of BAC, platform, domain, and operating system. Binary logistic regression, Pearson's $\chi 2$- value, odd ratios and p-value tests were performed for analyzing correlations among factors of BAC. This examination also revealed that ignoring session misconfiguration and improper input validation problems are the critical factors for creating BAC vulnerability in application.**

***Keywords* - Cyber Security, Web Application Vulnerabilities, Web Application Exploitation Techniques, Broken Access Control (BAC).**

I. INTRODUCTION

With the pace of revolution in modern technology, businesses have changed their ways for providing services to its consumer due to satisfy the growing expectations and rapid changing behaviors. Nowadays, web applications are becoming the key instrument for a business, in almost every sectors, to manage their business processes, that includes supply chain management, customer relationship management, employee management, etc. With the use of session management facility, a web application can easily response different service requests securely from its authorized users. Usually a session of the application has been initialized through authenticating the user by some factors of verification such as username and password. Therefore, the application can provide a user friendly customized environment for the consumers where they feel comfortable and find satisfaction in accessing only their authorized resources. Access control features ensures the restriction of accessing web resources such as web pages, database tables, etc. and it is the security configuration for preventing unauthorized access from the intruders. While the use of web applications in managing the business processes marked an important epoch in the history of modern business, risk of loss has been increased, at the same time, in case the existence of vulnerability remains in the application due to careless design and coding during its development. It is evident from the current OWASP list, Broken Access Control (BAC) has been marked as 5th rank vulnerability depending on its existence in the web applications and the adverse consequences [1]. Design flaws (such as Improper Input Validation, Sensitive Data Disclosure, Session Misconfiguration, Directory Readable, etc.) in access control area of the web application architecture may cause higher level privilege of the general user or intruder into the system. Effect of exploitation through BAC vulnerability may lead to serious damages to the application such as unauthorized access in the administrative privilege sections, complete compromise of a web application, etc.

It is not surprising that authentication and access control vulnerabilities are listed among the top ten vulnerabilities in OWASP 2017 list [1], and have been discovered in high-profile applications such as IIS [2] and WordPress [3]. An analysis of exploits for some specific vulnerabilities such as Structured Query Language Injection (SQLi) [3], Cross Site Scripting (XSS) [4], Cross Site Request Forgery (CSRF) [5], Local File Inclusion (LFI) [6], Remote File Inclusion (RFI) [7], Local File Disclosure (LFD) [8], are often found due to improper implementation of user authentication and management of active session which is one of the top two risks according to OWASP [1]. The user authentication and access control problem with the prevention process has been explored in several research. A study conducted on SQLi, Broken

Authentication, Session Management, and XSS web application vulnerability. The author discussed the code level problem analysis of those application layer weaknesses and recommended a guideline for the developers to secure the web application [20]. A study performed on root cause analysis to detect the Session Management and Broken Authentication vulnerabilities and prescribed solutions have been given to reduce the recurring attack of the web application [21]. Process of identifying the Broken Authentication vulnerability, attack procedure, and prescribed guidelines were discussed to protect the web-based system from the intruder [22]. A technique Nemesis, used for preventing access control vulnerabilities and Exploiting Authentication problem on web application are presented in the paper. The author implemented Nemesis through a tool by which the developer can control the given vulnerabilities in a small amount of time [20]. The study explaining the types of Broken Exploiting Authentication problem and Session Management attacks of web applications. Precautionary measures of the given problems were also illustrated at the end of the study [23]. A detailed comparison between Attributed Based Access Control (ABAC) model and traditional role-based models for showing the advantages of ABAC. The work also described ABAC logical architecture and securing policies that can be used in web service access control decisions [10]. An approach introduced a transformation tool FIX ME UP that finds access-control errors and produces candidate repairs that was evaluated by examining ten real-world PHP applications [11]. A prototype Nemesis implemented to protect all known authentication and access control bypass attacks. The evaluation confirmed that the prototype can protect real-world applications [12]. Some other research explained and approached types of Web access control policies. A logic-based policy management approach introduced focusing on XACML (eXtensible Access Control Markup Language) policies, which have become a standard for specifying and enforcing access control policies for various applications and services in current Web-based computing technologies [13].

The security intrusion process has been examined [14],[15]. Other researchers have focused on modeling and designing tools that make some degree of security assessment possible [16]. Alhazmi and co-workers have presented two models for the process of vulnerabilities discovery using data for Windows 98 and NT 4.0. In this work focused on the density of defects in software that constitute vulnerabilities, using data from five versions of Windows and two versions of Red Hat Linux [17],[18],[19]. After reviewing the above, it is found that insignificant research work has been conducted on Broken Access Control. This paper makes an assessment and analysis on Broken Access Control vulnerability and its several types of reasons and exploitation techniques. Also analyze risk factors of reason of BAC.

The rest of the paper is constructed as follows Section 2 describes the methodology of this research, and the experimental result analysis is shown in Section 3. Section 4 makes a discussion based on result analysis. Finally this paper conclude with the significance of result of the research and the future works.

II. METHODOLOGY

a) Data Collection:

330 websites were examined to conduct the experiment where the samples were collected from search engine using dork. A dork is a search string containing advanced search operators to find the exact information that a user look for. Some examples of the used dorks in this examination are “inurl admin/upload.php”, “inurl admin/config.php”, “inurl admin/dashboard.php”, “inurl admin/login.php”, “inurl site/backup.zip”, “inurl site/database.sql”, etc. Modification in syntax during producing dork may vary depending on the specific requirement as well as different search engines’ (such as bing, qwant, yahoo, etc.) basic requisite. Once the primary sites are observed, it forwarded to the pre-processing phase to ensure the availability of BAC vulnerability in the site.

b) Pre-Processing Phase:

Pre-processing of data is very important for making a proper and valid data-set. The collected raw data is incomplete, inconsistent and tempered if the data is not validate or examined in a proper way. After receiving the list of our selected sites from the output of the dork, it was examined through four exploitation techniques to verify the existence of BAC in those applications.

i) Access on Redirection Setting:

It is one of the best practices of the designer/ developer of web application to separate administrative section pages from the general user panel for avoiding unauthorized access. However, some exceptions have been observed in this study where any user can easily access to the super admin section without any restriction. The following sample code is the instance for which BAC vulnerability existed in the web application where there is no session required for accessing sensitive page(s).

```
Line 1: //** Code Start **//
Line 2: <?  // Access Controlling is not required here //
Line 3: include’inc/config.php’ ;
Line 4: include’inc/conn.php’;
Line 5: // Insecure DB Connentions for accessing Data //
Line 6: $update= mysql query(”UPDATE FROM close_bid where
        item.name = ’ ” . $item_name . ” ’ ” ) ;
Line 7: if($update) {
Line 8:    mysql.close($conn)
Line9: }
Line 10: ?>
Line 11: //** Code End **//
```

Line 6 of the code represents to create a database connection while user access the add_admin page. However, there is no exact sessions for controlling access on the add_admin page,

therefore, user can access that page without facing any authentication process.

Figure 1: User access super admin panel without restriction.

The above snapshot (in Figure 1.) presents the super admin panel in which user can access without restriction.

ii) Misconfiguration of Sensitive Data Retrieval

In some cases, developers or system administrators keep backup files (such as anyname.zip, database.sql, backup.zip, downloads.zip, fullbd.sql, etc.) readable and downloadable from the hosting server machine to access locally/ remotely for the purpose of troubleshooting/ recovering defacement in case any disruption takes place in web application. Therefore, the general user can read/ download those sensitive files without any constraint.

Figure 2: BAC vulnerability allows user to read/ download confidential files from host server of the web application.

The screen short of a BAC vulnerable website (shows in Figure 2.) where it allows general user to read/ download confidential files from the hosting server. Let consider a web application "http://www.xyz.com". In case the directory of the application is read-able, user can read any files of the directory inserting the directory name after the URL like "http://www.xyz.com/groups/". Now user can access all the files included into "groups" directory. Intruder may try to read different files e.g. "abc.com.zip" in the "group" directory i.e. "http://www.xyz.com/groups/abc.com.zip" that may lead to breach the access security of the confidential file. Once the "abc.com.zip" compressed file downloaded, intruder can access all backup files of the system after extraction.

iii) Unauthorized Cookie Access:

Authorization process deals with the level of access to a user of any application and also to limit their privileges among the system resources. Due to improper definition of session in the code of the application causes unauthorized session access in the web application. The following sample PHP code is the example of such coding.

```
Line 1: //** Code Start **//
Line 2: <? session_start();
Line 3: //    Checking For Access Validity //
Line 4: if(!$_SESSION['member']) {
Line 5:    header('Location:login.php') ;
Line: exit;
Line 7: }
Line 8: // Access Controlling is not required here //
Line 9: include'inc/config.php' ;
Line 10: include'inc/conn.php';
Line 11: // Insecure DB Connentions for accessing Data //
Line 12: $delete= mysql_query("DELETE FROM close_bid where
Line 13:  item.name = ' " . $item_name . " ' " ) ;
Line 14: if($delete) {
Line 15:  mysql.close($conn)
Line 16: }
Line 17: ?>
Line 18: //** Code End **//
```

In Line 2 of this code, the system creates a new session. Line 3 checks for the validity of the session. Once it validated, it will allow user to access header.php page. Line 9 includes the configuration file without any restriction. In Line 12, therefore, any user can delete/update or modify database information's without any authorized credentials.

The developers always follow a best practice for building database system. In maximum cases, they set the Administrative ID for Super_Admin Cookie ID value as ID=1 and then the Admin values are given sequentially such as 2, 3, 4, and so on. While browsing the cookie data, a user/intruder can change the ID e.g. value from ID=1004 to ID=1. It will then cause a serious change to their account in case the session is not properly defined for Admin Page which leads to get the all privilege of the changed ID=1(Main_Admin).

From 330 of sample, it is observed that the number of BAC affected web applications were 129 with 4 major risk factors. Those risk factors were categorized by some independent variable values which is the main cause for Broken Access Control (BAC) vulnerability, i.e. Disclosure of Sensitive Data, Unusual Redirections, Session Misconfigurations, Used Languages, Operating Systems, Server/Platforms. All data set were verified by the authority of the Cyber Security Centre, DIU.

a) Statistical Analysis of the Pre-Processed Data:

The prime objective of this investigation is to identify the association among the factors with BAC vulnerability and discover those factors which are statistically significant. Initially after pre-processing collected data, association and p-value among BAC with various factors has been encountered by $\chi 2$ - test. Binary logistic regression has been conducted among those factors which are statistically significant ($p<0.05$) having the odds ratio (OR) with 95% confidence interval (C.I). After that, the Pearson $\chi 2$ - test has been used to determine the association among factors. The whole analysis has been executed by IBM Statistical Package for Social Sciences

(SPSS version 22.0).

III. Results

Small sample technique is used as sampling method for this study. The technique has been constructed using the following Equation (1) [24]:

$$s = \frac{X^2 + NP(1-P)}{d^2(N-1) + X^2P(1-P)} \quad (1)$$

Here, required sample size is represented as '*s', 'N' is* the population size, '*P' is* the population proportion, '*d'* the degree of accuracy expressed as a proportion, and 'X_2' is the table value of chi-square for 1 degree of freedom at the desired confidence level (3.841). G*Power 3.1.9.2, a statistical tool, is used to identify the sample size of our investigation by using the Eq.1. Linear multiple regression test has been conducted under F tests family where number of predictors is selected as 4 in our case since the maximum predictors of the testing model is the types of exploitation. The value of α err prob was set as 0.05 and Power (1-β err prob) is selected as 0.95 in the tool. As per the result from the tool, minimum 129 valid samples was required. Figure 3 represents the percentage of sample between BAC free and vulnerable website.

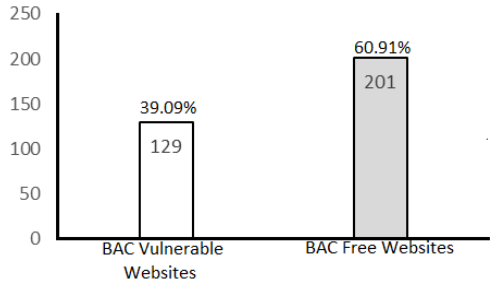

Figure 3: Percentage of sample between BAC free and vulnerable website.

Among the 330 samples, 39.09% websites were affected with BAC vulnerability whereas the remaining 60.91% applications were observed free from BAC. Access on redirection settings, misconfiguration of sensitive data retrieval, and unauthorized cookie access exploitation techniques were found effective for BAC vulnerable sample collection for this study. Manual penetration testing method [25] using double blinded strategy [26] was chosen to collect information for this examination. This dataset has been analyzed initially with the demographics i.e. sectors. Binary logistic regression, Pearson's $\chi 2$- value, odd ratios and p-value tests were conducted to analyze correlations among the factors of BAC that includes reason of BAC, exploitation techniques, platform, and application hosed the operating system. The analysis is discussed below.

Table 1 represents the frequency analysis for the existence of BAC vulnerability in five sectors (education, e-commerce, government counterpart, health, and private company) of our sample.

TABLE 1. FREQUENCY ANALYSIS FOR THE EXISTENCE OF BAC VULNERABILITY AMONG FIVE SECTORS

Sector	Frequency	Percentage
Education	33	25.58%
E-commerce	36	27.91%
Govt. Counterpart	28	21.71%
Health	10	7.75%
Private Company	22	17.05%
Total	**129**	**100.00%**

It is observed from the above table that the web applications of E-commerce are mostly affected by the BAC vulnerability with the percentage of 27.91% to compromise their access privileges whereas health sites are the least affected sector with only 7.75% for the given type of exploitation. Educational institution, government counterpart, and private company sites were vulnerable with BAC with the percentage of 25.58%, 21.71%, and 17.05% respectively.

The data of 330 web applications (where 39.09% of the applications were affected by BAC vulnerability and the rest of them were not affected) analyzed in the result section. Here the total analysis process take place at the three different tables includes frequency distribution with p-value, odd ratio with confidence interval of predictors, and association among factors.

TABLE 2. FREQUENCY DISTRIBUTION WITH P-VALUE OF RISK FACTORS BETWEEN REASON OF BAC, EXPLOITATION TECHNIQUES, SECTORS, PLATFORMS, AND OSS VS. THE PRESENCE OF BAC VULNERABILITY IN THE WEB APPLICATION.

Factors		BAC Vulnerability Status		P-Value
		Found	Not Found	
Reason of BAC	Improper Input Validation	71	79	0.021*
	Sensitive Data Disclosure	10	67	
	Session Misconfiguration	48	10	
	Directory Readable	0	45	
Exploitation Techniques	Access on Redirection Setting	60	109	0.000*
	Misconfiguration of Sensitive Data Retrieval	42	74	
	Unauthorized Cookie Access	27	18	
Sectors	Education	33	97	0.917
	Ecommerce	36	38	
	Govt. Counterpart	28	20	
	Health	10	8	
	Private Company	22	38	
Platform	PHP	113	65	0.000*
	JAVA	4	37	
	.NET	12	99	
OS	UNIX	72	101	0.000*
	Windows	23	38	
	Cent-OS	34	62	

Table 2 represents the frequency distribution of Broken Access Control (BAC) web applications vulnerability with

significant variation among risk factors of BAC vulnerability. Here it clearly shows that the factor "Exploitation Techniques", '"Platform'', and ''Operating System" ($p<0.0000$) is highly associated with BAC vulnerability. This analysis also reveals that "Reason of BAC" ($p<0.021$), is also associated with BAC vulnerability. At the same time, it has been observed from the result that "Sectors" are not associated with the given vulnerability.

The impact of the predictors (risk factors) on BAC vulnerability in web applications have been illustrated in the Table 3. To compare different groups with 95% confidence interval (CI), Odds ratio (OR) has been used in Table 2. In this table, it is clearly represented that ".NET" and "Java" platform have statistically significant relationships ($p<0.05$) with BAC vulnerability in the sample web applications.

TABLE 3. ODDS RATIO (OR) WITH CONFIDENCE INTERVAL (C.I.) OF PREDICTORS

Predictors	Category	Sig.	OR	95% C.I. for OR	
				Lower	Upper
Reason of BAC	Improper Input Validation	0.0000	1.8904	1.2081	2.958
	Sensitive Data Disclosure	0.0000	0.1681	0.0827	0.341
	Session Misconfiguration	0.0000	11.3185	5.4589	23.47
	Directory Readable	0.5010	0.0000	n/a	n/a
Exploitn. Technique	Access on Redirection Setting	0.0000	0.7339	0.4710	1.1436
	Mis-config. of Sensitive Data Retrieval	0.0000	0.8285	0.5196	1.3212
	Unauthorized Cookie Access	0.5041	2.6912	1.4138	5.1227
Platform	PHP	0.8262	14.7769	8.1000	26.9577
	Java	0.0104	0.1418	0.0493	0.4084
	.Net	0.0399	0.1057	0.0549	0.2035
Operating System	UNIX	0.0000	1.2507	0.8022	1.9498
	Windows	0.0000	0.9307	0.5250	1.6502
	Cent-OS	0.0000	0.8024	0.4901	1.3136

Reasons of BAC such as "Improper Input Validation", "Sensitive Data Disclosure", and "Session Misconfiguration", Exploitation Techniques of BAC i.e. "Access on Redirection Setting", and "Misconfiguration of Sensitive Data Retrieval"; Development platform like "Java" and ".Net"; and Operating Systems like "UNIX", "Windows", and "Cent-OS" are highly significant factors for BAC vulnerability in web applications which have 1.8904, 0.1681, 11.3185, 0.7339, 0.8285, 0.1418, 0.1057, 1.2507, 0.9307, 0.8024 times higher risk respectively than those are avoiding the given factors. Similarly except the factors, the reason of Directory Readability, Unauthorized Cookie Access exploitation technique, and site developed with PHP platform are respectively 3.6812, 2.6912, and 14.7769 times higher than those who are not related with those factors.

TABLE 4. ASSOCIATIONS AMONG THE FACTORS OF BAC VULNERABILITY

	Reason of BAC	BAC Exploitation Technique	Platform	OS
BAC_Existance				
χ^2	101.648	100.745	132.601	67.185
P-Value	0.000	0.000	0.000	0.000
Reason of BAC				
χ^2		17.658	20.719	31.657
P-Value		0.007	0.002	0.000
BAC Exploitation Technique				
χ^2			327.907	284.629
P-Value			0.000	0.000
Platform				
χ^2				402.080
P-Value				0.000

The Pearson's $\chi 2$ with p-value among the factors have been discussed in Table 3. P-value with less than 0.01 (<1%) among association factors are treated as highly significant whereas p-value less than 0.05 (<5%) are denoted as significant. It is observed from the table that all factors are highly significant association among themselves i.e. highly significant relationship between "BAC_Existance - Reason of BAC ($p=.000$, $\chi 2=101.648$)", "BAC_Existance - BAC Exploitation Technique ($p=.000$, $\chi 2=100.745$)", "BAC_Existance – Platform ($p=.000$, $\chi 2=132.601$)", "BAC_Existance - OS ($p=.000$, $\chi 2=67.185$)", "Reason of BAC - OS ($p=.000$, $\chi 2=31.657$)", "BAC Exploitation Technique - Platform ($p=.000$, $\chi 2=327.907$)", "BAC Exploitation Technique – OS ($p=.000$, $\chi 2=284.629$)", and "Platform – OS" ($p=.001$, $\chi 2=402.080$)". There is also significant relationship between "Reason of BAC - Platform ($p=0.002$, $\chi 2=20.719$)".

IV. DISCUSSION

The study has been performed over 330 web applications including 129 BAC vulnerable and 201 non BAC vulnerable web applications where the participant sectors of web application include Education, E-commerce, Govt. Counterpart, Health, and Private Company that were developed with PHP, Java, and .Net platform. The hosting server of the sample applications were found operating with UNIX, Windows and Cent-OS respectively. This study has revealed that "Reason of BAC", "BAC Exploitation Techniques", "Platform" and "OS" are the leading factors for the applications to get exploited through BAC vulnerabilities. It has been observed from the sample data that ".Net" developed application are found BAC vulnerable with the 68.22% whereas application built on "PHP" and "JAVA" platform are affected by BAC with 16.28% and 15.50% respectively. At the same time, "Cent-OS" hosted application are more prone to BAC with "51.16%" whereas "UNIX" and "Windows" are affected by BAC with 25.58% and 23.26%

respectively. From the above result, it cannot be claimed that "Platform" and "OS" are directly responsible for creating BAC vulnerability in a web application as they are separate entity for producing the given vulnerability in the application. However, it can be stated that .Net developers are less careful about the reasons of BAC to be fixed rather than other platform developers. In this investigation, "Reason of BAC" and "BAC Exploitation Techniques" are only be considered for analysis.

It is notable that the web applications that have "Session Misconfiguration", "Improper Input Validation", and "Sensitive Data Disclosure" problem, are more prone to be affected by BAC vulnerability which leads to get privilege access for an unauthorized user. On the contrary, "Access on Redirection Setting" and "Misconfiguration of Sensitive Data Retrieval" are the significantly effective technique to exploit BAC vulnerability. From this analysis it is found that web application having session misconfiguration problem have more high risk (OR=11.3185) to be exploited through BAC vulnerability than an application with no session misconfiguration problem. The sites having improper input validation problem have vigorous risk (OR=1.8904) compare to the applications that are adequately validated the input.

The above mentioned five factors are found significant both in $\chi 2$ test and binary logistic regression analysis. It has been explored a factor (i.e. BAC Exploitation Technique) in the study which is significant in binary logistic regression but insignificant ($p<0.007$) in chi square test. One the other hand, BAC reason i.e. "Directory Readability" does not act as a responsible factor for BAC.

Limitations: The statistical data analysis in this study was conducted with only 330 data containing the web applications both BAC vulnerable and BAC free information. The result of the analysis may vary if large amount of data be considered for the investigation.

V. Conclusion

Lack of following secure designing practices such as enforce adequate input validation, taking measures to restrict sensitive data disclosure, secure session configuration and management, ensure control on directory readability, etc. while building web applications by the designers and developers, are the root cause of having BAC vulnerability in the application. The designers and developers are not intentionally do such things, perhaps, it is the lack of awareness regarding the consequences of the harmful BAC vulnerability in web applications. Different best practices of have been recommended by the professionals to reduce the risk of attack through BAC vulnerability by this time. Web development and management professional should always keep an eye on the latest web problems and its solutions to be secured from the intruder. In the paper the associative relation of factors have been detected for BAC vulnerability and the possibility of preferences among the obtained factors has been estimated. The results can be used to increase awareness among the web designers and developers about different factors. It will be helpful for them to take preventive measures before the site to be hosted on live. Moreover the research work would guide the future researchers to find out some new other important factors of BAC vulnerability.

Acknowledgment

We acknowledge the authorities of the organizations who have given us the permission to conduct our examination on their websites. Also, we recognize Cyber Security Centre, DIU for verifying our data to conduct this study.

References

[1] "*Top 10 2017-Top 10 - OWASP*", Owasp.org, 2017. [Online]. Available: https://www.owasp.org/index.php/Top_10_2017-Top_10. [Accessed: 15- June- 2018].

[2] *Microsoft Internet Information Server Hit Highlighting Authentication Bypass Vulnerability*. Available: http://www.securityfocus.com/bid/24105, [Accessed: 20- June- 2018].

[3] *WordPress Cookie Integrity Protection Unauthorized Access Vulnerability*. http://www.securityfocus.com/bid/ 28935, 2008. [Accessed: 17- June- 2018].

[4] J. Thome, L. K. Shar, D. Bianculli, and L. Briand, "Security Slicing for Auditing Common Injection Vulnerabilities," 2017, *Journal of Systems and Software*.,to be published.

[5] I. Hydara, A. B. M. Sultan, H. Zulzalil, and N. Admodisastro, " Current state of research on cross-site scripting (XSS)–A systematic literature review, " *2015 Information and Software Technology*, pp. 170-186.

[6] A. Z. M. Saleh, N. A. Rozali, A. G. Buja, K. A. Jalil, F. H. M. Ali and T. F. A. Rahman, "A Method for Web Application Vulnerabilities Detection By Using Boyer-Moore String Matching Algorithm," *2015 Procedia Computer Science*, pp.112-121.8

[7] A. Begum, M. M. Hassan, T. Bhuiyan and M. H. Sharif, "RFI and SQLi based local file inclusion vulnerabilities in web applications of Bangladesh," *2016 International Workshop on Computational Intelligence (IWCI)*, Dhaka, 2016, pp. 21-25.

[8] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens & G. Vigna, "You are what you include: largescale evaluation of remote javascript inclusions, " *2012 In Proc. of ACM conf. on Computer and communications security*., pp. 736-747

[9] M. I. Ahmed, M. M. Hassan, T. Bhuyian, "Local File Disclosure Vulnerability: A Case Study on the Web Applications of Public Sector, *10th International Conference on Computer and Electrical Engineering (ICCEE 2017)* ", Edmonton, Canada, October 2017.

[10] E. Yuan and J. Tong, "Attributed based access control (ABAC) for web services," In Web Services, 2005, ICWS 2005. Proceedings. 2005 *IEEE International Conference on. IEEE*, 2005.

[11] S. Son, S. Kathryn, McKinley, and V. Shmatikov, "Fix Me Up: Repairing Access-Control Bugs in Web Applications," In NDSS. 2013.

[12] Dalton, Michael, C. Kozyrakis, and N. Zeldovich. "Nemesis: Preventing Authentication & [and] Access Control Vulnerabilities in Web Applications," (2009).

[13] G. Ahn, H. Hu, J. Lee and Y. Meng, "Representing and Reasoning about Web Access Control Policies," *2010 IEEE 34th Annual Computer Software and Applications Conference*, Seoul, 2010, pp. 137-146.

[14] E. Jonsson and T. Olovsson. "A quantitative model of the security intrusion process based on attacker behavior." *IEEE Transactions on Software Engineering 23*, no. 4 (1997): 235-245.J

[15] B. B. Madan, K. Gogeva-Popstojanova, K. Vaidyanathan and K. S. Trivedi, "Modeling and quantification of security attributes of software systems," *Proceedings International Conference on Dependable Systems and Networks*, Washington, DC, USA, 2002, pp. 505-514.

[16] J. Schultz, E. Eugene, D. S. Brown, and T. A. Longstaff, "Responding to computer security incidents: Guidelines for incident handling," No.

UCRL-ID-104689, *Lawrence Livermore National Lab*., CA (USA), 1990.

[17] O. H. Alhazmi and Y. K. Malaiya, "Quantitative vulnerability assessment of systems software," *Annual Reliability and Maintainability Symposium, 2005. Proceedings*, Alexandria, VA, USA, 2005, pp. 615-620.

[18] O. H. Alhazmi and Y. K. Malaiya, "Modeling the vulnerability discovery process," *16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)*, Chicago, IL, 2005, pp. 10 pp.-138.

[19] O. Alhazmi, Y. Malaiya, and I. Ray, "Security vulnerabilities in software systems: A quantitative perspective," *In IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 281-294. Springer, Berlin, Heidelberg, 2005.

[20] O. B. Al-Khurafi and M. A. Al-Ahmad, "Survey of Web Application Vulnerability Attacks," *2015 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, Kuala Lumpur, 2015, pp. 154-158.

[21] D. Huluka and O. Popov, "Root cause analysis of session management and broken authentication vulnerabilities," *World Congress on Internet Security (WorldCIS-2012)*, Guelph, ON, 2012, pp. 82-86.

[22] L. Murphey, "Secure Session Management: Preventing Security Voids in Web Applications," *The SANS Institute 29* (2005)

[23] N. B. Nagpal, and B. Nagpal, "Preventive measures for securing web applications using broken authentication and session management attacks: A study," *International Conference on Advances in Computer Engineering and Applications (ICACEA)*, vol. 2014. 2014.

[24] V.K.Robert, W.M.Daryle, "Morgandeter Mining sample size for research activities,"*Educational and Psychological Measurement*, The NEA Research Bulletin, December, 1960, Vol. 38 , p. 99.

[25] Y. Stefinko, A. Piskozub and R. Banakh, "Manual and automated penetration testing. Benefits and drawbacks. Modern tendency,"*13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*, Lviv, 2016, pp. 488-491.

[26] B. H. Kang "About Effective Penetration Testing Methodology," 2008, *Journal of Security Engineering*, JSE Vol. 5, No.5,