Cloud Based Secure Web Application

Ahmed Hashim Mohammed
Education College ,University of Mustansiriya/Baghdad.
Email:Msc_ahmedHM@yahoo.com
Dr. Hanaa M. A. Salman.
Computer Science Department, University of Technology /Baghdad.
Email: salmanhanna2007@yahoo.com.
Dr. Saad K. Majeed
Computer Science Department, University of Technology /Baghdad.
Email:saad@yahoo.com

Received on:4/2/2015 & Accepted on:8/10/2015

ABSTRACT

The types of web attack continue to appear and add their impacts on web application security, SQL injection, and XSS is one type of these attack, that causes extremely high risk for web application through stolen critical information or broken web authentication.

The aim of this paper is to design and implement software as a service for securing web applications from attack which cover two directions, the first direction: propose modern black-box web application vulnerability scanners that diminish the false positive and false negative drawback of the current black-box web application vulnerability scanner, The second proposed subsystem cloud platform as a service web application firewall hosting web application which scan all http requests to deny or accept it according to dummy execution result.

The numbers of vulnerable web applications are selected to evaluate the capability of the proposed system in detect attack and protect vulnerably web application also efficiency has evaluated through measure the server performance when WAF (Web Application Firewall) was disabled and enabled on vulnerably web application and comparison made between these two cases. According to analyzing of the experimental result shows that the proposed system can effectively and efficiently protect web application and discover SQL injection, and XSS vulnerabilities.

Keyword: Cloud, SQL injection, XSS vulnerabilities, Web Application, Security

حماية المواقع الالكترونية بأستخدام السحابة الالكترونية

الخلاصة

ان ظهور أنواع من الهجمات على شبكة الإنترنت لا تزال تضيف تأثيراتها على أمن تطبيقات الويب، احد أنواع هذه الهجمات حقن SQL، وXSS التي تسبب مخاطر عالية للغاية لتطبيقات الويب من خلال سرقة المعلومات الهامة أوكسر عملية الوصول على شبكة الإنترنت.يهدف هذا البحثالى تصميم وتنفيذنظام سحابة الكترونية توفر خدمة حماية لتطبيق الويب . هذا النظام مقسم الى قسمين من النظم الفرعية المقترحة.

https://doi.org/10.30684/etj.2015.117204
2412-0758/University of Technology-Iraq, Baghdad, Iraq

أول نظام فرعي مقترح هو كاشف الثغرة على شبكة الإنترنت، بأستخدام تقنية تحاكي عملية القرصنة لكشف الثغرة في الويب للتغلب على هذه المشكلة ركز النظام المقترح على توفير خدمة برنامج كاشف الثغرة في تطبيق ويب، استنادا إلى تحليل قيمة hash لاستجابة الويب من خلال تغييرها . النظام الفرعي الاخر المقترح خدمة سحابة الكترونية كتطبيق منصة جدار حماية للويب التي توفر طبقة الأمان لاستضافة تطبيقات الويب من خلال تفحص جميع طلبات HTTP لإنكار أو قبول ذلك وفقا لنتيجة عملية تنفيذ وهمية. وتم اختيارعد من تطبيقات الويب المعرضة للهجوم لتقييم قدرة النظام المقترح في منع وحماية تطبيقات الويب من الاختراق كما تم تقييم كفاءة WAFمن خلال قياس أداء الخادم عندما كان WAF معطل و مفعل على تطبيق ويب قابل للاختراق وأجراء مقارنة بين ها ين الحالتين. وفقا لتحليلال نتيجة التجريبية اثبتت على أن النظام الفرعي لمقترح يمكنه بفعالية وكفاءة حماية تطبيقات الويب واكتشاف حقن SQL،ونقاط الضعف XSS.

INTRODUCTION

Cloud computing means using common networking standards and internet protocols to access virtualized resources distributed on network that is make the physical systems stand out hidden from the user who accomplishes on services and applications benefit from unlimited resources [1]. But security is the main challenging problems to consider for any business or organization that utilizes online functionality for a successful e-commerce service [2]. There are many difficulties facing in process of securing web applications. First: web applications are distributed through by using client/server architecture, with Hypertext Transfer Protocol (HTTP) request / response. Second: different programming languages are used to develop web applications, such as, Hypertext Markup Language (HTML), CSS (Cascaded Style Sheet), JavaScript on the client-side and Personal Home Page (PHP), Ruby, Java on the server-side. And third: dynamic nature that web applications holds [3]. Therefore, many types of hacking ways can disclosure private information of vulnerable web application, respond to hacker request violated in web application cost additional bandwidth, CPU time and administrator to recover, in addition to other kind of invisible cost [4]. Such use malicious data injection which exploits fault syntax in a web application's or incorrectly validate inputs from the client-side this type of weakness (SQL, [5], [6] Injection and XSS) expose the most dangerous vulnerabilities in web application [7].

The rest of the paper is organized as follows, Section (2) gives the related work of this paper, in section (3) begin by introducing system architecture. It also discusses the design and implementation of proposed black box testing and proposed web application firewall. In section (4) performs the effectiveness and efficiency of the proposed system by comparisons accomplished with another system and results discussed, in section (5) covers the conclusions of this paper and recommendations for future work.

Related Work

Web application security analysis is an area of research that has received considerable study. In this section, a related works to different area of detection and prevention web application vulnerabilities have discussed:

1. **In (2009), Dwen Y. et al,** have discussed string-based and command-based attacks and the corresponding defense methods using application firewall. They also differentiate between the current popular practices and proposed defense properties in accordance with its exposure to attack [8].

2. **In (2010), Zhang. L. et al,** suggest a method to generate test suites taking the weight of each test value into account, performed these test suites and analyzed subsequent result based on HTTP response code and response HTML [9].
3. **In (2011), Kapodistria. H.et al,** proposed a web application firewall that scans http requests and responses to discover attacks such as stored XSS, a property that cannot be found on other web application firewall [10].
4. **In (2011), Jeom G.,** introduced an effective and simple SQL Query removal technique that uses mixed static and dynamic analysis and evaluates the efficiency through various tests [11].
5. **In (2012) Atul S. and Dhore M.,** studied a new policy based Proxy Agent, which categorizes the request as a scripted request, or query based request, and then, detects the corresponding type of attack, if any in the request [12].
6. **In (2012) Pulei X,** proposed a model-driven penetration test framework for web applications that consists of a penetration test methodology, a grey-box test architecture, a web security knowledgebase, a test campaign model, and a knowledge-based Pen Test workbench [13].
7. **In (2013)Alattar, M. H., &Medhane, S. P.,** have proposed the detection model of SQL Injection vulnerabilities and SQL Injection Mitigation Framework. These approaches are based on SQL Injection grammar to identify the SQL Injection vulnerabilities during software development and SQL Injection Attack on web-based applications[14].
8. **In (2013) Vaishali M. and Waghmare. J.,** introduced WAF to detect both known and unknown attacks uses both the signature based model in which all the attack patterns are already stored and normal behavior model in which all the normal traffic that target the web applications are already stored in the database[15].
9. **In (2014)Vibhakti M. and Milind T.,** have proposed a framework for building secure and anti-theft web applications that must be secure from attacks by improving existing web prevention techniques [16].
10. **In (2014) Adam L. D.,** introduced an approach to automatically detect EARs in web and present deDacota, as a first step in the direction of making web applications secure [17].]

Proposed System Design

This section introduces the proposed system, provided software as a service, and can reach it anywhere, and anytime which divided into subsystems services first: Hash Black Box (HBB) Testing, which is depicted in section (3.1), Second, Web Application Firewall Server (WAF), which is depicted in section (3.2). The proposed system is shown in the Figure (1).

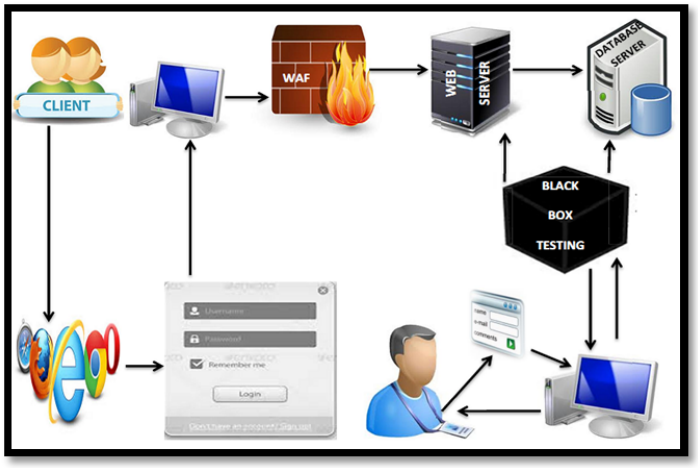

Figure (1): Proposed system.

HBBT

The architecture of the proposed HBBT testing is showed in Figure (2) to detect the web vulnerability. The subsystem is divided into three main functional models which include web crawler, injector and analyzer.

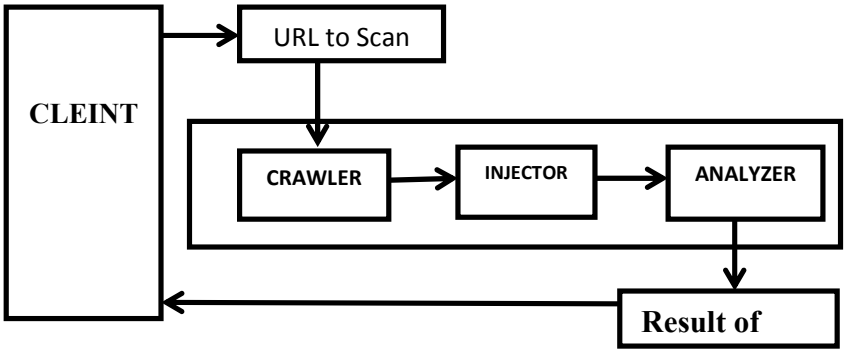

Figure (2): The HBB Testing.

1. Crawler Model: The web crawler provide interaction scenario between scanner system and web application through interface, user enter the URL of web to be tested, and crawler model generates a request to the web server for crawling all URLs of this web.
2. Injector Model: This model sends HTTP request with payload attack data such as ('OR '1," OR 1 = 1 -- -," OR "" = ", <script>alert (“aa”) ;< /script>) which also can be uploaded by tester to all input field (input field, text area, password field, checkbox, radio button) that extract from the web form.
3. Analyzer Model: This model analysis HTTP response according to hash value to detect web vulnerability so any error resulted by the malicious payload that can affect the response can be detected by the value of hash, even if the change in the response is two small and the error hasn’t known before.

WAF

This subsystem is setting on server side and watches HTTP conversations between a client browser and web server at layer seven to protect web application from the two main attacks SQLI and XSS, when a client requests a service from the web server WAF intercepts users' requests and analyzes it. If a request includes malicious input, WAF denied the request and redirected to another page, this proposed system consists of three stages (sanitization, validation and verification) as showed in the Figure (3).

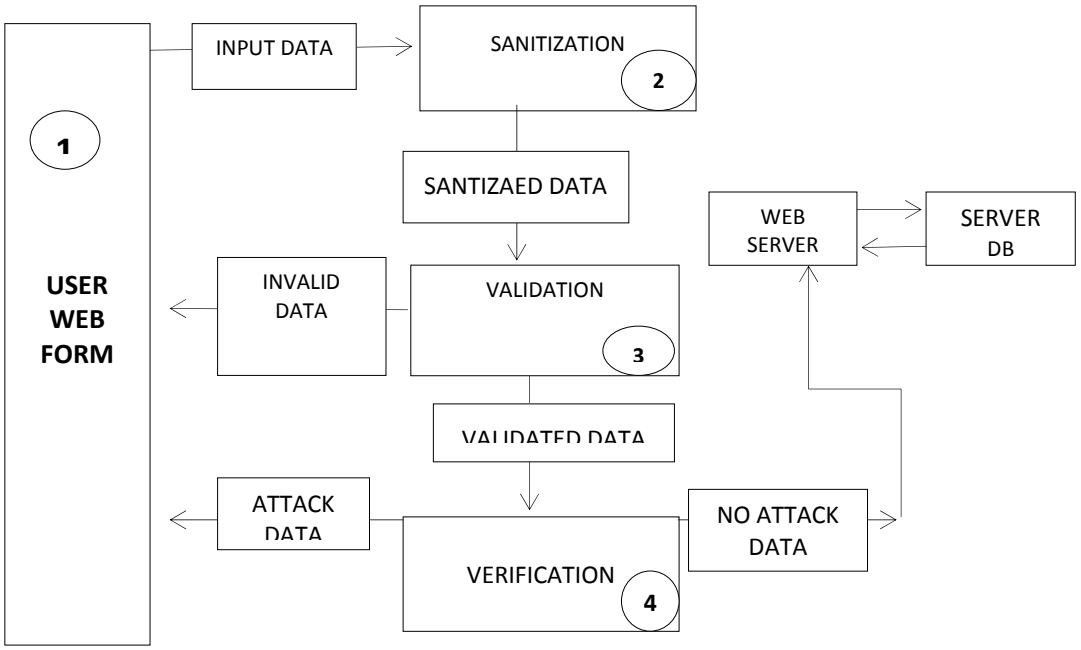

Figure (3) Proposed WAF Subsystem

Sanitization Stage: The basis of this approach always removes data that possibly cause damage for web application from all inputs that originate from client; the proposed phase is cleaning process of data according to the type of input and sends it to the web server. Validation Stage: Another defense line proposed to justify each input data is validate stage to protect web based on regular expression in design custom functions. Verification Stage: This stage signifies the core of proposed system to protect web application, one to detect and prevent XSS vulnerability. The proposed method to detect and prevent XSS script uses blacklist, white list, rules and DOM because offers get into the structure and content of the input HTML identical JavaScript access HTML page when a browser parsed it.

Evaluation Proposed System

The main factor that is used as the basis to evaluate any goal management system is subdivided into two broad categories of indicators: effectiveness and efficiency, these performance measurements are used to evaluate the HBBT, and WAF proposed subsystem.

Evaluate HBBT

This section is devoted to evaluate HBBT, originally vulnerable web application is developed and contained within intentional a number of known vulnerabilities' application to measure the false-negative and false-positive metrics correctly, this web is hosted in apache server and setting with the same configurations for all experiments.

Test SQL Injection Vvulnerability

To evaluate the HBBT capability in detect SQL injection attack a comparison made between number web applications vulnerability scanners with the proposed system in scanning the developed vulnerable web application. As shown in Table (1), HBBT scanner detected four SQL injection vulnerabilities out of four known implemented vulnerabilities and corresponding false negative is zero vulnerabilities in addition the false positive vulnerabilities of the detected was zero .

Table (1): SQL injection Result

Scanner	Detected	False Negative	False Positive
Proposed System	Four	Zero	Zero
Acunetix	Three	One	One
W3af	Three	One	One
Web security	Zero	Four	Zero
Vega	Two	Three	One
Owasp Zap	Three	Two	One

Test XSS Vulnerability

There is three XSS vulnerability known and detected manually in the developed vulnerable web application, also the same number of vulnerability has been detected when scanning the web application using HBBT through sending (603) number of http request contained attack value.

To evaluate the HBBT capability in detect XSS attack a comparison made between web applications vulnerability scanners tools that explained previously in section 3.5 with the proposed system in scanning the developed vulnerable web application. As shown in Table (2), HBBT scanner detected three XSS vulnerabilities out of three known implemented vulnerabilities and corresponding false negative is zero vulnerabilities in addition the false positive vulnerabilities of the detected was zero .

Table (2) Cross Site Script (XSS) Results

Scanner Tool	Detected	False Positive	False Negative
Proposed System	Three	Zero	Zero
Acunetix	Two	Zero	One
W3af	Two	Zero	One
Web Security	Two	Two	Two
Vega	Two	One	One
Owasp Zap	Zero	Zero	Three

Evaluate WAF

An insecure web application has been developed and implemented that contain common vulnerable web applications that help in evaluate the proposed WAF subsystem.

WAF Effectiveness

This evaluation refers to measures the capability of WAF in provide the appropriate protection on different websites, and the output meets the intended or expected result as planned activities, the automatic and manual penetration testing performed; the three different cases are selected.

A) Prevent SQL Injection Attack

Case one is scanned insecure web application hostile to SQLI vulnerability Firstly, when the proposed WAF was disabled and the obtained result from scanning result show there are three SQLI vulnerability found, The preceding experiment has performed for a second time but when WAF was enabled and scanned the unchanged insecure web application hostile to SQLI vulnerability under the same condition of the previous experiment, The obtained result gave reasons for the impact of WAF on insecure web application which, there is clearly no SQLI vulnerability found.

A case two have performed by comparison the proposed WAF with another software called PHP-Intrusion Detection System (PHPIDS) for PHP based web application was enabled and disabled on Damn Vulnerable Web App (DVWA) which has (8) SQL injection vulnerability known number before enable any protection. PHPIDS enabled to assessment capability on protect DVWA from SQL injection attack, the PHPIDS prevented some of SQL injection vulnerability detected previously but cannot prevent all SQL injection vulnerability In contrast when proposed WAF was enabled the number of SQL injection vulnerability detected after scanning DVWA was zero.

B) Prevent XSS Attack

Case one is scanned insecure web application hostile to XSS vulnerability, Firstly, when the proposed WAF was disabled and the obtained result from scanning result show there are two XSS vulnerability found Likewise, the previous experiment has accomplished; the unchanged insecure web application scanned hostile to XSS vulnerability but at what time WAF was enabled under the same condition of the previous experiment.

The obtained result explained obviously the power of WAF on protect insecure web application from XSS vulnerability, which there is clearly no XSS vulnerability detected.

A case two has performed by comparison the proposed WAF with PHPIDS when enabled and disabled on DVWA, this web have 40 XSS vulnerability known number before enforce any protection, The DVWA have restarted and PHPIDS enabled to assessment capability on protect the web, the number of XSS found when scanning DVWA was 20 vulnerability detected while when proposed WAF enabled zero was the number of SQL injection vulnerability detected after scanning DVWA.

C) Evasion WAF Attack

A case three produce different experiments performed through applied new discovered style of attack against developed vulnerable web application to compare the proposed WAF with different software called mode security. The idea of proposed style of attack based on send attack data consecutively to more than one input field at once, rather than the traditional style where it is sending only one attack and waiting for the answer. Hence, the proposed style of attack applied on vulnerable web application when mod-security was enabled first, send a segment of attack data (<iframesrc=http://localhost/web_system/signup.php/) to username input field and next send the remainder attack data (iframe>) to password input field.

The obtained result indicate mod-security failure in fighting proposed style of attack besides prove the power of style attack for bypass this firewall because the mod-security check first attack data disjointed of the second attack data while server processing as it integrated of two attacks data which represent the same attack data that detected and prevented when sent jointed, In contrast when proposed WAF was enabled and attack applied on vulnerable web application the obtained result confirm the ability of proposed WAF to fight attack and saves web application from new harm attack.

WAF Efficiency

The efficiency of the proposed system is measures when send normal query, and send malicious query, to indicate the WAF impact on the server

A) Normal Request

In the beginning, we send a normal request, to the vulnerable webpage specifically (logn.php), by 100 numbers of virtual users, when WAF disabled, all performance results recorded, and compared with the recorded result of send the same normal input data, to the same vulnerable webpage, by 100 numbers of virtual user but in the case when WAF enabled. In comparison response time for the (login.php) web page, and the number of user are (64) in two cases, as shown in Table (3). The first case is, when WAF is disabled, and the second case is, when WAF is enabled. The overall page load time when WAF is disabled is 134 ms compared to the page load time when WAF enabled, which is 245 ms, so the difference is 100 ms which, represents a short delay of response time. In addition, the time for first byte, when WAF enabled, is about twice the time required when WAF is disabled.

Table(3): Responses Page for normal request

WAF	Request	Virtual User	Response Transfer Time ms	First Byte Time ms	Last Byte Time ms	Page Load Time ms
Disabled	Login.Php	(64)	1	121	122	134
Enabled	Login.Php	(64)	1	242	243	245

Furthermore, it can be concluded that average response time for test all pages, as shown in the Table (4), when WAF disabled, is 134 ms while when WAF enabled, is 245 ms and the difference between these states was 111 ms, which indicates the affected time of WAF enabled on server response was too small.

Table(4): Slow Responses (Average) for Normal Request

WAF	Request	Time To First Byte ms	Time To Last Byte ms	Page Load Time ms
DISABLED	Login.Php	73	74	85
ENABLED	Login.Php	155	156	179

Another important indication, for web application's scenario completion time as shown in Figure (4), which explain send normal data, when WAF is disabling (insecure web) the start completion time is about (5.5) second whereas the start completion time when WAF is enabling is (14.74) second, the time difference between two cases because the WAF take about (10) second for checking http request before send http respond.

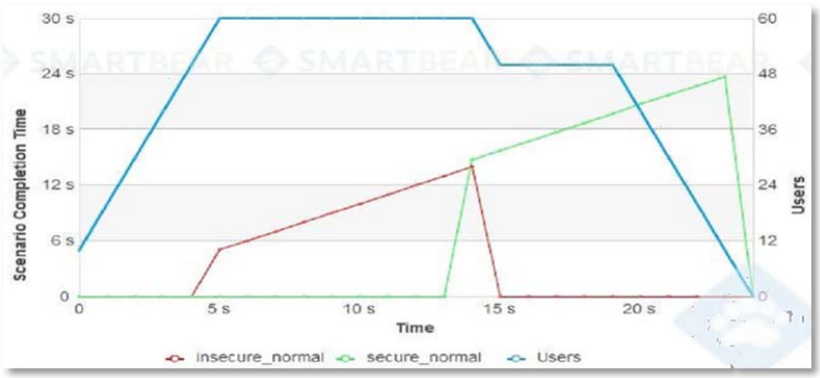

Figure (4): Completion Time for Send Normal Data

Malicious Request

The second case sent a malicious request to a vulnerable website by 100 numbers of virtual users when WAF disabled to compare with the results of send same input data to the same vulnerable website by 100 numbers of virtual user but when WAF is enabled.

The slow response time for specific page in two cases; the first case is when WAF is disabled, the second case when WAF is enabled shown in Table (5), the overall page load

time when WAF is disabled is 119ms for 28 numbers of user compared to the page load time when WAF enabled, which is 230ms for 95 numbers of user, so the difference is just about 100ms, which represent a short delay of response time. In addition time for first byte when WAF enabled is about twice the time required when WAF is disabled.

Table (5): Slow Response for Attack Request

WAF	Request	Virtual User	Response Transfer Time ms	First Byte Time ms	Last Byte Time ms	Page Load Time ms
Disabled	login.php	28	1	100	101	119
Enabled	login.php	95	1	216	217	230

Furthermore, it can be concluded that average response time for test all pages, as shown in Table (6) when WAF disabled is 78ms while when WAF enabled is 145ms and the difference between these states is 67ms time, which indicates the affected time of WAF enabled on server response is too small. Figure (5), explain the completion time for send malicious data.

Table(6): Average Response for Attack Request

WAF	Request	Time to First Byte ms	Time to Last Byte ms	Page Load Time ms
Disabled	login.php	65	66	78
Enabled	login.php	125	126	145

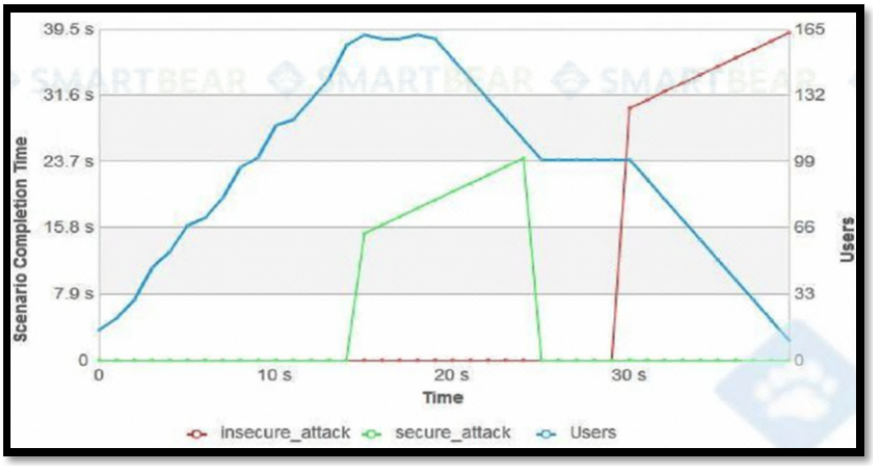

Figure (5): Completion Time for Malicious Data

When WAF is disabled the start completion time is 30.07 second while it is 15.11 second when WAF is enabled, this is because the malicious data cause delay in the function of server while in the case of enabling WAF this malicious data prevented so the

start time the same as start time in the normal request, this can be summarized in the Table (7).

Table (7): Completion Time Of Normal and Malicious Data

WAF	Data Type	Input Data	Start Time S	Last Times
DISABLED	Normal	aa	5.07	13.99
	Malicious	1' or '1'='1	30.07	39.04
ENABLED	Normal	aa	14.75	23.70
	Malicious	1' or '1'='1	15.11	24.09

CONCLUSIONS

By designing, and implementing the cloud base web application security system, which is proposed by this dissertation, and by inference results, many conclusions have been reached out as the following:

1. The WAF sanitization insures input data is correct, so it counted as enrichment level, because it helps server manages data accurately, and avoid user resubmit it again.
2. The WAF capability, increased when validation phase used because, denies any request invalidated pattern before gain access to the server.
3. Employing the WAF is supposed to be highest part of defenses in server hosting to protect any web application from vulnerabilities. Execute simulation of input data on WAF, before executed it on web application, be capable of prevent high vulnerability attack.
4. A WAF can protect a site from newly found vulnerabilities, or previously successful attacks.
5. The use of hash algorithm in scanning process, can detect the vulnerability in the web application with high precision and recall.
6. The timestamp of payload executed in the server has different value for each HTTP request.
7. The proposed cloud based secure web application system gives the capability of increasing web application security level because it uses various security techniques.

REFERENCES

[1] Barrie S.," Cloud Computing Bible", Wiley Publishing, Inc., 2011.
[2] Lee G., "A Critical Appreciation of Current SQL Injection Detection Methods", 2011.
[3] Vahid G., Ali M., Aysu B. and Shabnam M., "A Systematic Mapping Study of Web Application Testing", Information and Software Technology February 9, 2013.
[4] JanMin C., Fan Yukuan, and Jincherng Lin," Cost-Sensitive Defense Strategy Selection for Web Applications", In Proceedings of the IEEE International Symposium on Secure Software Engineering (2010).
[5] Dr. Mohammed Najm Abdullah "Active Directory Monitoring System Using Optimized Web Database Application and AJAX Techniques", Eng. & Tech. Journal Vol.28, No.18, 2010
[6] Ekhlas Khalaf Gbashi,'Improvement Personalization of Website Using Database and Cookies', Eng. & Tech. Journal, Vol. 28 , No. 8, 2010

[7] Pulei X.,"A Model-Driven Penetration Test Framework for Web Applications", Ph.D. in Computer Science, University of Ottawa, 2012.
[8] Joel S. and Mike S.," Hacking Exposed-Web Applications - Web Application Security Secrets & Solutions", McGraw-Hill/Osborne, 2002
[9] Paco H. and Ben w.," Web Security Testing Cookbook Systematic Techniques to Find Problems Fast", O'Reilly Media, Inc.,2009.
[10] Michal Z.," The-Tangled-Web-a-guide-to-securing-modern-Web-applications", San Francisco, No Starch Press, Inc., 2012
[11] Josh P., "The Basics of Web Hacking", Elsevier, Inc., 2013.
[12] DwenRen., Allen Y., Peichi Liu., and Hsuan C., "optimum tuning of defense settings for common attacks on the web applications", IEEE, 2009.
[13] Baig M. M., "new software testing strategy" , PhD in computer science and information technology, university of engineering & technology ,2009.
[14] Pulei X. and Liam P., "A Model-Driven Penetration Test Framework for Web Applications", IEEE Eighth Annual International Conference on Privacy, Security and Trust, 2010.
[15] Jan M., and Chia L., "An Automated Vulnerability Scanner for Injection Attack Based on Injection Point", IEEE, 2010.
[16] Xuruzhi G. and Deng L., "A Database Security Gateway to the Detection of SQL Attacks", IEEE 3rd International Conference on Advanced Computer Theory and Engineering ,2010.
[17] Lijiu Z., Qing G., Shushen P., Xiang C., Haigang Z. , and Daoxu C., "A Web Application Vulnerabilities Detection Tool Using Characteristics of Web Forms", IEEE Fifth International Conference on Software Engineering Advances,2010.