International Journal of Computer Applications (0975 – 8887)
Volume 95– No. 9, June 2014

WebGuard: Enhancing Intrusion Detection in Multi-tier Web Applications

Asha U. Patil
PVPIT, Pune, India

Arati M. Dixit
PVPIT, Pune, India

ABSTRACT

In today's world web applications and internet services have become an integral part of daily life, enabling communication and the management of personal information from anywhere. In order to accommodate humungous increase in demand and data complexity, web applications are moved to multitier design. With the increase in popularity of web applications, these applications also become target of various attacks. To protect multitier web applications several intrusion detection systems (IDS) have been proposed. This paper focuses on development of *WebGuard* – designed to deploy the IDS at the front end web server as well as the back end data base server. *WebGuard* generates the containers for isolating user sessions using virtualization technique. This strategy mainly focuses on detecting SQL Injection, Privilege Escalation, Session Hijacking, Direct DB and Cross Site Scripting (XSS) Attacks in multi-tier web applications by using a pattern mapping algorithm. This paper also proposes a preventive system to secure web applications from XSS attacks.

General Terms

SQL Injection, Session Hijacking, Privilege Escalation, Direct DB, Cross Site Scripting Attacks.

Keywords

Intrusion Detection System, Pattern Mapping, Virtualization.

1. INTRODUCTION

Now a day's web services and applications have increased in terms of quantity, popularity and complexity, because of the rapid rise in information technology era. Daily tasks such as banking, travelling, social networking and online shopping are all done by using the web. So it resides at the core of almost all advanced technologies that make human life simplified. The number of e-commerce sites, Social networking sites and other web portals are increasing day by day. As a result cyber-attacks too are increasing along with the growth of web services and web applications. These web attacks try to access secure data with an endeavor of interception of unauthorized data over an information technology infrastructure. Popular web attacks include many attacks like Injection attack, Session Hijacking attack; Denial-of-Service attacks and many more.

Intrusion detection System (IDS) is generally used to protect web applications [21]. This system detects known attacks by matching misused traffic patterns or signatures [16]. A class of IDS based on machine knowledge can be used to detect unknown attacks by finding abnormal network traffic that vary from normal behavior, before found during the IDS training phase[23][17]. The web IDS and the database IDS can find abnormal network traffic sent to either of them. But these IDS cannot determine attacks where in normal traffic is used to attack the web server and the database server. For example if an intruder enters into a web server as a normal user but by using web serve weakness issues privileged data base queries from the web server to attack database server. In order to detect these types of attacks an association between web server request and data base queries needed. For that intrusion detection system is implemented both at the web server and the database server.

A *Container* is generated by using virtualization technique [1], [14], referred it as a lightweight process. It looks like a disposable server for client sessions. It is possible to create thousands of containers on a single web server, and these virtualized containers can be deleted, removed or quickly reassigned to serve new sessions. A single method with passion develops new containers and recycles used ones. It means a single physical server can run constantly and serve all web requests. On the other hand from a logical viewpoint each session have dedicated web servers and isolated from other sessions. This allows finding out suspect behavior by both session and user. If it detects abnormal behavior in a session, then all traffic within this session is treated as polluted traffic.

WebGuard represents the deployment of intrusion detection system (IDS) for both ends; in which front end is web server and back end is database server. This simply represents a virtual containers web server architecture where multiple containers are created for each user session using lightweight process. This containers based and session separated architecture enhances security performances as well as provides the isolated information flows that are separated in each container session. This allows finding out the mapping between web server request and database queries. In multi-tier web architecture client sends HTTP request to the web server and then web server issues SQL queries related to the client request to the data base server to retrieve or update data depending on the HTTP request.

WebGuard models such mapping relationships from all the legitimate users so as to detect web attacks. With this virtual container based approach it is possible to build a pattern mapping between web request and database queries.Fig.1depicts the virtual container architecture, which created containers both at front end and back end. In which client gives web request as CR_q and has associated database query DQ. Web server receives response from database as DR then web server sends response to client as CR_s. This whole transaction is isolated in one session it called as a container and it is denoted by VE in Fig. 1.

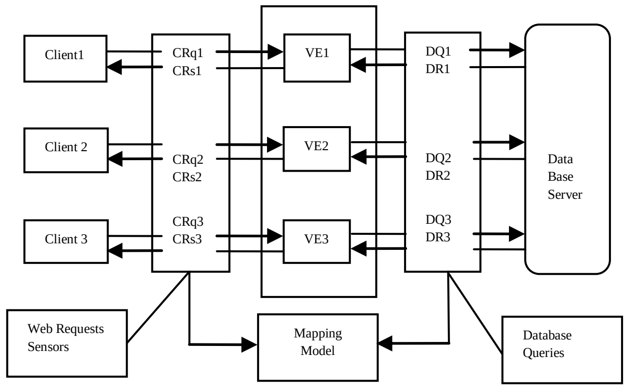

Fig.1 *WebGuard* Virtual Container Architecture

2. RELATED WORK

Web applications are become more vulnerable today, so there is need to find out new way to secure them. According to OWASP (Open Web Application Security Project) attacks like SQL Injection, Cross Site Scripting (XSS) are more dangerous to web applications. This OWASP present top ten list of web applications vulnerabilities, in this attack SQL Injection, and Cross Site Scripting attacks are included [9].

Before this more work done on the security of the web applications. On the basis of web application architecture Intrusion Detection Systems (IDS) for web server and database server is used. But these IDS have two types according to its work nature. First one is anomaly detection detects the unknown attacks by identifying abnormal behavior. Second one is the misuse detection detects the only known attacks by matching the signatures of the attack.

Rule Based Systems proposed a new open source intrusion detection systems based on the misuse detection type. Here manually have to characterize the attack for that there is need to study and analyze the attack. After analyzing this signatures used to detect the attacks by matching the signature with the data collected from real traffic. Main disadvantage of this system is rules are generated manually, therefore traffic not included in rule is considered as a abnormal [22].

Two methods are used in this paper to detect attacks. First method is used to create profile from different user behavior; it is called as training method. Second method detects attack on the basis of malicious score and reported anomalous queries, this method called as detection method [23].

This approach based on the stateful analysis of multiple event streams. So here intrusion is defined as the sequence of malicious actions that convey system from normal state to compromised state through a number of in-between states. State transition analysis build signatures of attacks by analyzing sequence of actions performed by an attacker to attack the system. By using this system easily find out the attacks [24].

In this approach first detailed characterization of web application is done. For that web application internal state is defined as information that survives single client server session or here simply minimum state information is passed as a cookie to a browser. This approach model out attack state for that it requires state information in which that attacks is generally executed. This system works in two modes training and detection. At the time of training mode attack signatures are generated and in detection mode this signatures are used to detect attacks [25].

Here illustrated a new method in which simulation of different traffic features is done with the help of histogram. Therefore first collect the real traffic from sensors and then use that for create pattern by histogram. And during detection phase these pattern used to find out malicious states in applications [26].

Above described method of intrusion detection are not efficient and powerful. So there is need to develop the system to detect attack and prevent some of them. Here in this paper presented a new approach *WebGuard* it enhance the intrusion detection and it is most power full in detection of intrusions in multi tier web applications.

The easiest and the most effective client- side solution to the XSS attack for user is to disable JavaScript in their browsers. Unfortunately, this solution is often not feasible because a large number of web sites use JavaScript for navigation and improved presentation of information. Noxes, a tool is a client-side web -proxy that relays all web traffic and serves as an application-level firewall. The approach works without attack-specific signature. Noxes works as a personal firewall which allows or blocks connections to websites based on filter rules. Filter rules are mainly the white list and blacklist of URLs specified by the user. Whenever a browser sends a HTTP request to an unknown website not listed in filter rules Noxes instantly shows a connection alert to client who can then decide to allow or reject the connection and it remembers the client's exploit for future use. Noxes requires user configuration, as well as user communication when a doubtful event occurs. This is a disadvantage of this tool[5].

Another client-side approach is present in , which aims to recognize information outflow using tainting of input in the browser. All client side solutions contribute one weakness, the requirement to install updates or additional components on each user’s workstation. While this might be a sensible prerequisite for skilled, security-aware computer users, it is supposed as an obstruction or is not even considered by the enormous bulk of users. Thus, the level of protection such a system can offer is severely limited in practice[11].

Server side solution makes helpful contribution in the field as XSS-Guard transforms the server programs such that they produce a shadow page for real response page. The key idea in the approach is to learn the purpose of the web application while creating the HTTP response page. This is done through shadow pages, which are generated every time a HTTP response page is generated. This pages are similar to the real HTTP response returned by the web application with mainly one important difference only retain the script that were intended by the web application to be included, and do not contain any injected scripts. Given the real and shadow pages, one can match up to the script content present in the real page with web application intended content, present in the shadow page. Any difference detected here indicates a variation from the web application's intentions and therefore signals an attack[10].

A Multi-agent system has been explored for the automated scanning of websites to detect the presence of XSS vulnerabilities usable by a stored XSS attack. It works by finding the input points of the application disposed of being vulnerable to a stored XSS attack then injecting selected attack vectors at the previously detected points. Finally it checks the web application for the injected scripts in order to confirm the accomplishment of the attack. It is not able to run-time detection and prevention of attack; also it can be used for attack detection only, with no method for prevention [12].

Other Server Side solution also has some E-guard algorithm approach, there is no system to handle scripts which are stored in Grey list, and these are left for future analysis. So this algorithm does not give a reasonable or can say total prevention from XSS attack. This is a passive method which does not provide dynamic detection and prevention of XSS attack. Also these solution do not provide a correct framework, some of them have partial implementation [13].

3. *WebGuard* MODEL

WebGuard builds the normality model to detect various attacks like Injection, Session Hijacking, Privilege Escalation, Direct DB attacks. To build the model it uses different pattern mapping techniques such as Deterministic Mapping (DM), Empty Query Set (EQS), No Matched Request (NMR) and Non Deterministic Mapping (NDM).We define following symbols for developing the mapping structure:

r_i : request for any session ‘i’.

Q_i : query set for session ‘i’.

ф : empty set.

Q_T : query for all sessions.

These mapping are represented in Table 1 below.

Table 1. *WebGuard* Pattern Mapping Techniques.

Sr.No	Pattern Name	Description
1	Deterministic	$r_i \rightarrow Q_i$
2	Empty Query Set	$r_i \rightarrow$ ф
3	No Matched Request	ф $\rightarrow Q_i$
4	Non Deterministic	$r_i \rightarrow Q_T$

Pattern mapping is the assignment of a label to a given input value. As illustrated in the Fig.1 the entire request from clients to the data base server are separated by sessions. Each session is assigned with a unique session ID. *WebGuard* normalizes the variables values in both HTTP request and DB queries and substitutes actual values of the variables with symbolic values. As a result session *i* will have set of request r_i and set of queries $Q_{i.}$ If total number of session are N, We have total web request REQ and SQL queries across all sessions. In DM Web request r_i appears in all traffic with the SQL queries set Q_i. The mapping patterns is then $r_i \rightarrow Q_i$ .For any session in the testing phase with request r_i, the absence of a query set Q_i matching the request indicates a possible intrusion. On the other hand, if Q_i is present in session traffic without the r_i, then this refers to as an intrusion. In special case, the query set may be the empty set, thus forms EQS pattern mapping technique. It means that the web request neither causes nor generates any database queries. For example, when a web request foe retrieving an image GIF file from the same web server is made , a mapping relationship does not exist because only the web request are observed. This type of mapping is represented as: $r_i \rightarrow$ ф. During the testing phase, we keep these web requests together in the set EQS.

In some case, the web server may periodically submit queries to the database server in order to conduct some scheduled tasks, such as backup. This does not require any web request we call it as NMR. This is similar to the reverse case of the empty Query set mapping pattern. These queries cannot match with any web request, and keep these unmatched queries in a set NMR. It is denoted like this ф $\rightarrow Q_{i.}$ During the testing phase, any query within set NMR is considered legitimate. The size of NMR depends on web server logic, but it is typically small. In NDM based on input parameters or the status of the web page at the time of the web request the same web request may result in different query sets. In fact, these query sets do not appear randomly, and there exists a pool of query sets. Each time they the same type of web request arrives, it always matches up and there exists a pool of query sets. Each time that the same type of web request arrives, it always matches up one of the query sets in the pool. The mapping pattern is denoted as $r_i \rightarrow Q_T$ Therefore, it is difficult to identify traffic that matches this pattern.

WebGuard is employed with four different types of pattern mapping techniques. These techniques are shown in Fig.2 systematically. As shown in Fig.2 when web request r_m comes at the web server logic(WSL) then according request it belongs to any one of the pattern mapping technique(PMT) as deterministic mapping(DM), non-deterministic mapping(NDM), But some web request are not having associated data base queries then it is included in empty query set(EQS). Sometimes web server have to do some special task like a backup or Corn jobs at that time there is no need of web request. So we include this type of queries of SQL Query Set (SQS) in No matched request (NMR) mapping pattern.

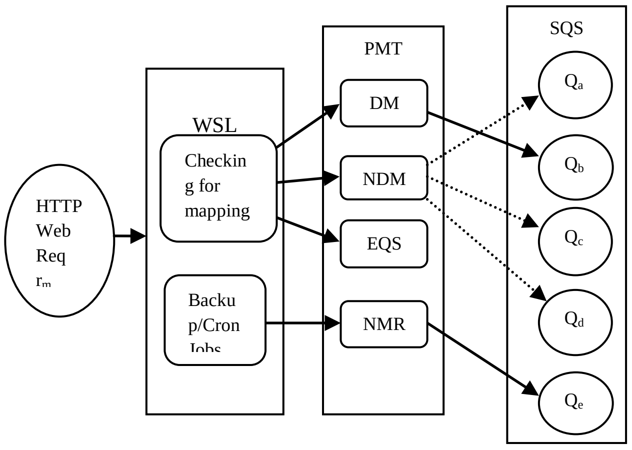

Fig. 2 *WebGuard* Pattern Mapping Architecture.

4. *WebGuard* METHODOLOGY

In *WebGuard* containers are created for each user session widely using virtualization technique. This strategy focuses on the detecting following attacks in multi-tier web applications by using a pattern mapping architecture.

4.1 Privilege Escalation Attack

WebGuard consider both users normal and administrator, so website serves both regular and administrator. This both users have different rights, therefore fetch different database queries. Administrator cause the set of admin level queries, and normal user triggers queries is in his rights. Now think that an attacker logs into the web site as a regular user and by using web server vulnerabilities update his rights and send admin queries to the database to get the administrator data. This type of attack cannot be detected by the IDS that is web server IDS or database IDS. Since both web request and database queries are not associated with each other means they are illegal. *WebGuard* is capable of detect this type of attack because according to mapping model database queries does not match with web request.

4.2 Hijack Future Session Attack

This category of attacks mostly occurs at the web server side. In this type of attack an attacker takes over the whole web server and therefore hijacks all resulting sessions and release attack. In this attack attacker hijack all unauthorized user sessions and send spoofed replies, drop user requests and eavesdrop. A session hijacking attack can also be called as Spoofing or man-in the-middle attack, an Exfiltration Attack, Denial-of Service or Packet Drop or Reply attack. *WebGuard* easily detect this attack also by using mapping model.

4.3 SQL Injection Attack

These types of attack do not need to compromise the web server [4]. Simply an attacker can use available vulnerabilities in the web server and the use the web server to send this malicious code to attack the back end database [20]. *WebGuard* provides a two-tier detection, even if the vulnerabilities are accepted by the web server .The relayed content to the database server would not be capable to take on the predictable structure for the given web server request, so by using mapping model this type of also easily detected.

4.4 Direct DB Attack

In this type of attack simply attacker avoids the web server and or firewalls and access directly to the database. Suppose attacker could also have already taken over the whole web server and send queries from the web server without the web request [15]. Therefore web server IDS or database IDS can not this attack but *WebGuard* detects because there is no any matching web request present.

4.5 Cross Site Scripting (XSS) Attack

Cross Site Scripting is up till now another type of attack on the web applications. In this malicious data is injected into a database so as to achieve unauthorized access to connection of an authorized user.

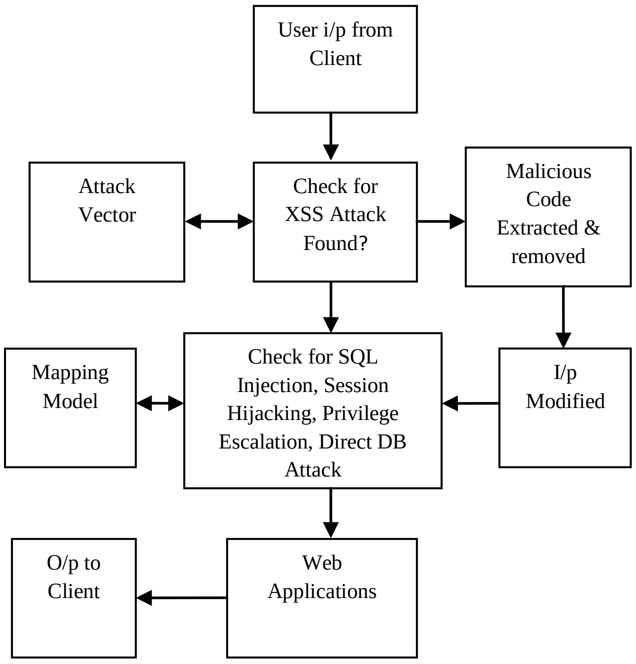
User i/p from Client
Attack Vector
Check for XSS Attack Found?
Malicious Code Extracted & removed
Mapping Model
Check for SQL Injection, Session Hijacking, Privilege Escalation, Direct DB Attack
I/p Modified
O/p to Client
Web Applications

Fig.3 *WebGuard* attack flow architecture.

Websites normally utilize scripts written in JavaScript coupled with HTML, which runs on a client side depiction application for faultless user experience [2],[3]. Attackers make use of the fact that there is a true relationship between a web server and a browser. Such attacks can take place when data sent to the server are located on the web site without being well analyzed for realistic security threats. If the data input in a form is a malicious script, it will be run by the browser. In the simplest case, a user will be shown pop-up window with its session ID entirely recognizing it.

Cross site scripting (XSS) is a usual attack method where in the attackers injects malicious client scripts via valid user inputs. In *WebGuard,* the entire user input values are normalized so as to construct a mapping model based on the structures of HTTP request and DB queries. Once the nasty user inputs are normalized, *WebGuard* cannot detect attacks hidden in the values. So in order to detect XSS attacks a pattern mapping step wise algorithm is offered in this paper. Also to detect Injection, Privilege Escalation, Session Hijacking, and Direct DB attack pattern mapping algorithm is presented here[1]. Data flows in the system to detect these attacks as shown in the Fig.3 this diagram systematically shows the working of proposed system.

5. *WebGuard* ALGORITHMIC STRATEGY

WebGuard protect web application from attacks like SQL Injection, Privilege Escalation, Session Hijacking, Direct DB and XSS. So it provides various algorithms for that, XSS attack algorithm used for XSS attack detection and prevention. This algorithm uses attack vector, once attack is detected it is removed from the input value. For detection of SQL Injection, Privilege Escalation, Session Hijacking and Direct DB attack pattern mapping algorithm is used. To map the pattern we require session ID for web request and associated database query, for collection of this session ID Session Handling algorithm is used. Once the session ID is collected it is used for mapping, for that it uses four different pattern mapping techniques. Last intrusion detection algorithm is used for detection of these attacks.

5.1 XSS Attack Algorithm

If request comes with some unknown attack signature, crafted rules R_{u1}, R_{u2}, R_{u3} and R_{u4} applied on the input data. These rules help to identify new generated attacks. A pattern based approach followed by some crafted rules has been used [18].

Regular expression and wrapper classes have been used for detecting and prevention of XSS attack in web application [19]. Rule Description can be summarized as-

R_{u1}. HEX value with semicolon encoding.

R_{u2}. DEC value without semicolon encoding.

R_{u3}. Mixed encoding with HEX and DEC values.

R_{u4}. Normal annotation like “/*xss*/”.

R_{u5}. Complicated annotation like “expre/*xss*/ssi/*xss*/ on”.

Some other notations are also provided for insertion, like “/*/*/”, “/* /* xss*/”.

Here presented pattern mapping step wise algorithm for XSS attack detection.

Algorithm: XSS Attack

A) Preprocessing Steps for XSS attack algorithm

Step 1: Compile patterns P_{a1}, P_{a2}, P_{a3}, … $P_{an.}$

Step 2: Declaration of allowed and disallowed input values.

Input = Request (S)

Step 3: Create Matcher Final Matcher for user defined input.

B) Processing Steps for XSS attacks algorithm

Step 1**:** Comment is replaced with quotes.

Step 2: Check all tags in input as pattern p_3 for open tag and p4 for end tag.

Step 3:

a. If the tag is opening tag extract the all attributes of tag. Check the presence of this tag in Allowed tag list. If present check body of the tag. If body contains attributes whether quoted or unquoted then add these name and values in param name and value list respectively.

b. Invoke allowed Attributes Check the presence or absence of this tag in disallowed tag list. If this tag is allowed. Extract the attribute name which has been retrieved from body of tag. If attribute is associated with this tag if it is present in vProtocolAtts (array used to store allowed tag values) array then invoke process param protocol.

c. Convert Unicode, hexadecimal in to decimal values.

d. Validate these Entities by invoking validate entity and check entity methods.

Step 4: If self-closing tag is not closed then close it.

Step 5: If blank spaces are present then remove it.

Output: Clean Request

5.2 Session Management Algorithm

WebGuard first collect the traffic from sensor, it is called as training data set. This data is then easily classified according to the pattern mapping techniques to build the mapping model. For Building the mapping model session ID is required, to collect the session ID *WebGuard* use Session Management Algorithm. This algorithm is responsible for providing correct and unique session ID to mapping model building algorithm. For any input web request data is available on the web server itself then there is no need to send query to the database. This type of web request is included in the Empty Query Set (EQS) according to pattern mapping techniques. If web request is not in the set TWR means the input web request is new then adds that request r into the TWR that is total web request. By taking r as a key session ID is appended in to the SR means variable to store the value of request ID. Similarly for each new query that is query absent in the TQS, add q in the TQS means total query set. Then by taking a as a key session ID is appended in to the set SQ means session query.

Algorithm: Session Management

Input: Training data set

Output: Session ID for web request r and database query q in the sets SR_r and SQ_q

```
1.  for each session separated traffic $T_i$ do
2.  Get different HTTP request r and Database query
    q in this session
3.  for each different r do
4.  if r is request to static file then
5.  Add r into the set EQS
6.  else
7.  if r is not in set TWR then
8.  Add r into TWR
9.  Append session ID I to the set $SR_r$ with r as the key
10. for each different q do
11. if q is not in the set TQS then
12. add q into TQS
13. Append session ID I to the set $SQ_q$ with q as the
    key
```

5.3 Mapping Model Algorithm

WebGuard generate the mapping model on the basis of web request and database queries. It build the mapping model after getting the session ID for web request and associated data base queries. *WebGuard* do mapping by using pattern mapping techniques. For mapping the pattern *WebGuard* use the threshold value t. So that if the mapping pattern appears more than t sessions that is cardinality of SR_r and SQ_q is greater than t then and then only mapping pattern has been found. Here for mapping model algorithm value of t is considered as 3.After iterating all sessions any query is left from query set TQS then that query is added into the NMR. NMR is the one of the pattern mapping technique, and means no matched request. Similarly for web request r is not added any where then it is moved into the EQS, EQS means empty query set.

Algorithm: Mapping Model

Input: Set of SR_r and SQ_q, Cardinality t.

Output: Mapping of HTTP web request and associated Database queries.

```
1. for each distinct HTTP request r in TWR do
2. for each distinct database query q in do
3. Compare the set SRr with set $SQ_q$
4. if $SR_r = SQ_q$ and Cardinality ($SR_r$) > t then
5. found a deterministic mapping from r to q
```

```
6.  Add q into mapping model set MSR of r
7.  Mark q in Set TQS
8.   else
9.   Need more training sessions
10.  returns False
11.  for each database query q in do
12.  if q is not marked then
13.  Add q into the Set NMR (No Matched Request)
14.  for each HTTP request r in set TWR do
15. if r has no deterministic mapping model then
16. Add r into the Set EQS (Empty query Set)
17. Return true
```

5.4 Intrusion Detection Algorithm

Once the mapping model is build it can be used for detection of abnormal behavior. Every session is compared with the mapping model that is web request will have only one rule at a time in the model. In this testing simply compare the web request with the rules provided by the mapping model. According to that testing *WebGuard* detects the intrusions present in the web applications.

Algorithm: Intrusion Detection

Input: HTTP web request r and database query set Q_T

Output: Shows intrusion detected

Step 1: If the rule for the request is deterministic mapping then $r \rightarrow Q_T$ ($Q_T \neq \Phi$), test whether Q_T is a subset of a query set of the session. If so, this request is valid then mark the queries in the Q_T. Otherwise, a violation is detected and considered to be abnormal and the session will be marked as suspicious.

Step 2: If the rule is Empty Query Set then $r \rightarrow \Phi$, then the request is not considered to be abnormal and it will not mark any database queries. No intrusion will be detected.

Step3: For the remaining unmarked database queries, see if they are in the NMR. If so, mark the query as no matched request.

Step 4: Any untested web request or unmarked database query is considered to be abnormal. If it exists within a sessions, then that session will be marked as suspicious.

WebGuard can be applied on web application created by us, it able to detect attacks like Privilege Escalation, SQL Injection, Session Hijacking, Direct DB, and Cross Site Scripting (XSS). *WebGuard* also able to prevent some attack like Session Hijacking, Cross Site Scripting (XSS).These results are partial as shown in Table 2.

Table 2.Web application protected by proposed approach

Sr. No.	Web Application	Vulnerable to attacks	Attacks Detected	Attacks Prevented
1	Simple web application created by us	Privilege Escalation, SQL Injection, Session Hijacking, Direct DB, Cross Site Scripting(XSS)	Privilege Escalation, SQL Injection, Session Hijacking, Direct DB	Session Hijacking, Cross Site Scripting(XSS)

6. CONCLUSION

The proposed *WebGuard* as an intrusion detection system detects typical web attacks like SQL Injection, Privilege Escalation, Session Hijacking, Direct DB and XSS (Cross site scripting attack) that occur in a multi-tier web application. *WebGuard* uses pattern mapping algorithm for detection purpose. This gives a mechanism to secure web application from XSS by using a framework based on attack vector and pattern matching approach.

The power of the proposed framework is that it can be applied on any existing web application without source code modification. The proposed *WebGuard* framework best at enhance and strengthen the multi-tier web application security.

7. REFERENCES

[1] Meixing Le, Angelos Stavrou, Brent Byoung Hoon Kang , "Double Guard : Detecting Intrusion in Multitier Web Applications", IEEE Transactions on dependable and secure computing volume 9, no 4, July/August 2012.

[2] M. Jons, B Engelmann, and J. Posegga, "XSSDS : Server-side Detection of Cross-site Scripting Attacks",Computer Security Applications Conference,2008 ACSAC 2008, Annual IEEE,pp 335-344,2008.

[3] A. Klein "Dom based cross site scripting or XSS of the third kind", Web Application Security Consortium, Articles, Vol. 4, 2005

[4] Anely. "Advanced injection in server applications", Technical report, Next Generation Security Software, Ltd,2002

[5] E. Kirda, C. Kruegel,G Vigna, and N. Jovanovic "Noxes : A Client side solution for Mitigating Cross-Site Scripting Attacks", Dijon France SAC' ACM 06 April 2006.

[6] A. K. Ganame, J. Bidou, F. Spies, "A Global Security Architecture for intrusion on Computer Networks", Montbeliard Volume 27, March 2008

[7] G.W. Dunlap, S. T. King, S. Cinar, M Basrai, "Enabling intrusion analysis through virtual-machine logging and reply", Boston, MA, USA, December2002

[8] http:// www.san.org/top-cyber-security-risk/

[9] A. Stock, J. Williams, and D. Wichers, OWASP *TOP 10*, OWASP Foundation 2013.

[10] Chiristoper Kruegel, G. Vigna, William Robertson, "A mutimode- approach to the detection of web-based attacks", Computer Networks 48 ELSEVIER pp.717-738.2005.

[11] P.Vogt,F Nentwich, N Jovanovic,C Kruegel,E. Kirda and G vigna. "Cross site scripting prevention with dynamic data taining and static analysis", 14th Annual network and Distributed System Security Symposium (ndss),2007.

[12] E.Gal an A.Alcaide A. Orfila, J blasco, "A multi-agent scanner to detect stored XSS vulnerabilities", IEEE International Conference on Internet Technology and Secure Transactions (ICITST)JUNE 2010

[13] M.James Stephen P.V.G.D. Prasad Reddy, ch Demudu Naidu, "Prevention of cross site Scripting with E-guard Algorithm", International Journal of Computer Application Volume22- No5 May2011.

[14] Y. Huang, A. Stavrou, A. K. Ghosh, and S.J ajodia. "Efficiently tracking application interactions using lightweight virtualization". In Proceedings of the 1st ACM workshop on Virtual machine security, 2008

[15] Y. Hu and B. Panda. "A data mining approach for database intrusion detection". In H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock, editors, SAC. ACM, 2004.

[16] Liang and Sekar. "Fast and automated generation of attack signatures: A basis for building self-protecting servers", In SIGSAC: 12th ACM Conference on Computer and Communications Security, 2005.

[17] Yi Xie and Shun-Zheng Yu, "A Large-Scale Hidden Semi-Markov Model for Anomaly Detection on User Browsing Behaviors", IEEE/ACM transactions on networking, vol. 17, no. 1, February 2009

[18] XSS Attack Vectors at http://ha.ckers.org/xss.html

[19] Bates, D., Barth, A., and Jackson, C. "*Regular expressions considered harmful in client-side XSS In* WWW", Proceedings of the 19th international conference on World wide web (New York, NY, USA, 2010), ACM,.

[20] A S Yeole, B B Meshram, "Analysis of Different Technique for Detection of Injection", International Conference and Workshop on Emerging Trends in Technology (ICWET 2011) – TCET, Mumbai, India,ACM

[21] T. Lane and C.E. Brodley. "Temporal sequence learning and data reduction for anomaly detection". In Proceedings of the 5th ACM conference on Computer and communications security, pages 150 to158. ACM Press, 1998.

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

[23] C. Kruegel and G. Vigna "Anomaly detection of web-based attacks", In Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS '03), Washington,DC, Oct. 2003. ACM Press.

[24] G. Vigna, W. K. Robertson, V. Kher, and R. A. Kemmerer. "A stateful intrusion detection system for world-wide web servers",In ACSAC 2003.IEEE Computer Society.

[25] M. Cova, D. Balzarotti, V. Felmetsger, and G. Vigna."Swaddler: An Approach for the Anomaly-based Detection of State Violations in WebApplications", In RAID 2007.

[26] Andreas Kind, Marc Ph. Stoecklin, and Xenofontas Dimitropoulos "Histogram-Based Traffic Anomaly Detection" IEEE transactions on network service management, vol. 6, no. 2,June 2009.