IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 12, Issue 5 (Jul. - Aug. 2013), PP 19-23
www.iosrjournals.org

Pattern Mapping Approach for Detecting Xss Attacks In Multi-Tier Web Applications Using Double Guard

A. Krishna Chaitanya[1], Mounika. B[2]
[1]*Associate Professor in IT, VCE, Hyderabad, India,*
[2]*M.Tech (C.S.E), VCE, Hyderabad,*

Abstract: *with the advent and explosive growth of the global Internet and electronic commerce environments, adaptive/automatic network intrusion and anomaly detection in wide area data networks and e-commerce infrastructures is fast gaining critical research and practical importance. Many researchers have been performed research for detecting intrusions in multi tier web applications. This paper focuses on Double guard that deploys the IDS at both front end web server and back end data base server has been developed. Virtualization technique has been used for creating containers for each user sessions. This strategy mainly focuses on detecting XSS attacks in multi tier web applications by employing a pattern mapping step wise algorithm. Cross Site Scripting Attack (XSS) belongs to top ten web application vulnerabilities. This paper proposes a mechanism to secure java web applications from XSS by applying a framework based on pattern mapping approach.*
Key words : *Pattern mapping, XSS attacks, Double guard, Virtualization.*

I. Introduction

A very important part of our day-to-day life, nowadays, is Information Technology. It lies at the heart 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 are becoming rampant. These attacks include illegal access of data, illegal interception of data, eavesdropping of unauthorized data over an information technology infrastructure, etc. Popular Web attacks include Spam, Phishing Attacks, Information warfare, Nigerian Scams, and Denial-of-Service attacks.

In order to protect multi tiered web services, Intrusion detection systems have been widely used to detect known attacks by matching misused traffic patterns or signatures. A class of IDS that leverages machine learning can also detect unknown attacks by identifying abnormal network traffic that deviates from "normal" behavior previously profiled during the IDS training phase. The web IDS and the database IDS can detect abnormal network traffic sent to either of them. But these IDSs cannot detect attacks wherein 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 issues privileged data base queries to the web server for exploiting web server. In order to detect these types of attacks an intrusion detection system is implemented both at web server and data base server.

A lightweight process referred to as "containers," (disposable servers for client sessions) are created by using virtualization. It is possible to initialize thousands of containers on a single web server, and these virtualized containers can be discarded, reverted, or quickly reinitialized to serve new sessions. A single physical web server runs many containers, each one an exact copy of the original web server. Double gueard approach dynamically generates new containers and recycles used ones. As a result, a single physical server can run continuously and serve all web requests. However, from a logical perspective, each session is assigned to a dedicated web server and isolated from other sessions. This allows us to identify suspect behavior by both session and user. If we detect abnormal behavior in a session, we will treat all traffic within this session as tainted.

1.1 Double guard

Double guard represents the deployment of IDS(intrusion detection system) both at front end web server and back end data base server. This simply represents a virtual container web server architecture where multiple containers are created for each user session using virtualization. This container based and session separated architecture enhances security performance as well as provides the isolated information flows that are separated in each container session. This allows us to identify the mapping between web server requests and data base queries. In multi tiered web architecture client sends HTTP requests to the web server and then web server issues SQL queries related to the clients request to the data base server to retrieve or update data depending on the HTTP request. The SQL queries that are issued by the web server depends on the type of HTTP request. Double guard models such a mapping relationships from all the legitimate users so as to detect web attacks.

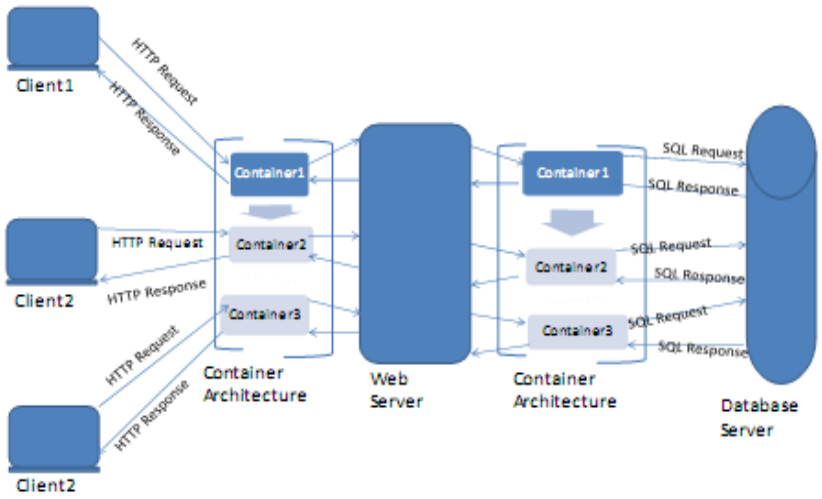

Fig 1: virtual container architecture

With this virtual container based approach it is possible to build a pattern mapping between web requests and data base queries. Fig 1 depicts the virtual container architecture that created containers both at front end and back end.

1.2 Pattern mapping

Pattern mapping is the assignment of a label to a given input value. As illustrated in the fig 1 all the requests from clients to the data base server are separated by sessions. Each session is assigned with a unique session ID. Double guard normalizes the variables values in both HTTP request and DB queries and substitutes actual values of the variables with symbolic values. As a results session i will have set of requests R i and set of queries Q i. If total number of sessions are N, we have total web requests REQ and queries SQL across all sessions.

Double guard is employed with four different types of pattern mapping techniques.

1.2.1 Deterministic Mapping:

Web request appears in all traffic with the SQL queries set Q_n . The mapping pattern is then $r_m \rightarrow Qn$ (Qn). For any session in the testing phase with the request rm, the absence of a query set Qn matching the request indicates a possible intrusion. On the other hand, if Q_n is present in session traffic without the r_m, then this refers to as an intrusion..

1.2.2 Empty Query Set:

In special cases, the SQL query set may be the empty set. It means that the web request neither causes nor generates any database queries. For example, when a web request for retrieving an image GIF file from the same web server is made, a mapping relationship does not exist because only the web requests are observed. This type of mapping is called $r_m \rightarrow \phi$. During the testing phase, we keep these web requests together in the set EQS.

1.2.3 No Matched Request

In some cases, the web server may periodically submit queries to the database server in order to conduct some scheduled tasks, such backup. This does not require any web request. This is similar to the reverse case of the Empty Query Set mapping pattern. These queries cannot match up with any web requests, and we keep these unmatched queries in a set NMR. 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.

1.2.4 Nondeterministic Mapping

Based on input parameters or the status of the webpage at the time the web request the same web request may result in different SQL query sets. In fact, these different SQL query sets do not appear randomly, and there exists a pool of query sets. Each time that the same type of web request arrives, it always matches up

with one (and only one) of the query sets in the pool. The mapping pattern is $r_m \rightarrow Q_i$. Therefore, it is difficult to identify traffic that matches this pattern.

1.3 XSS attacks

Cross Site Scripting is yet another variety of attacks on Web applications. In this malicious data is injected into a database so as to gain unauthorized access to a network connection of an authorized user. Websites, generally, employ scripts written in JavaScript coupled with HTML, which runs on a client side rendering application for seamless user experience. Attackers utilize the fact that there is a trust relationship between a Web server and a browser. Such attacks can occur when data sent to the server are put onto the web site without being properly analyzed for possible security threats. If the data input in a form is a nasty script, it will be run by the browser. In the simplest case, a user will be shown pop-up window with its *session ID* completely recognizing it.

II. Proposed System

2.1 PATTERN MAPPING STEP WISE ALGORITHM FOR DETECTING XSS ATTACKS

Cross Site Scripting (XSS) is a typical attack method where in the attackers inject malicious client scripts via legitimate user inputs. In Double Guard, all of the user input values are normalized so as to build a mapping model based on the structures of HTTP requests and DB queries. Once the malicious user inputs are normalized, Double Guard cannot detect attacks hidden in the values. So in order to detect XSS attacks a pattern mapping step wise algorithm is presented in this paper.

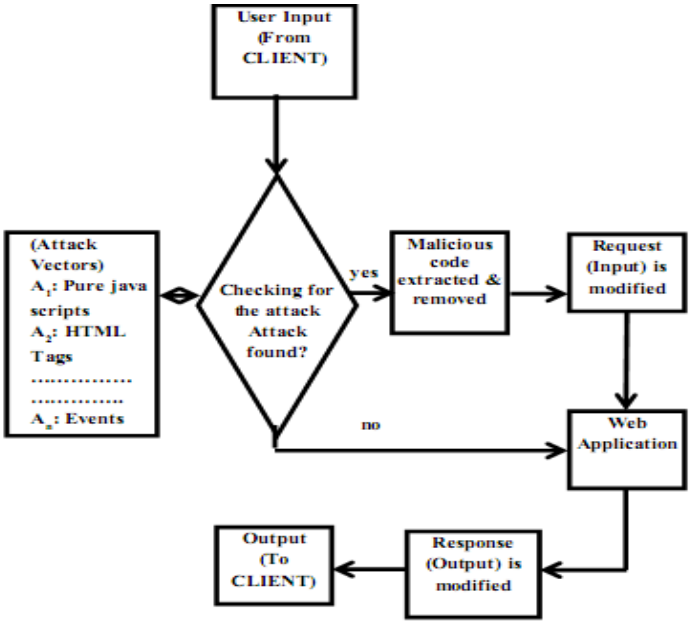

Fig 2: Flow diagram of XSS attacks

If request comes with some unknown attack signature, crafted rules R1, R2, R3, and R4 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.

Regular Expressions and Wrapper classes has been used for Detection and Prevention of XSS Attack in web application. Rule Description can be summarized as-

R1. HEX value with semicolons encoding.
R2. DEC value without semicolons encoding.
R3. Mixed encoding with HEX and DEC values.
R3. Normal annotation like —/*xss*/‖. .
R4. Complicated annotation like —expre/* xss*/ssi/*xss*/on‖.
Some other notations are also provided for insertion, like —/*/*/‖, —/*/*xss*/‖.

Pattern Mapping step wise algorithm-

2.1.1 Pre-processing Steps

Step 1 Compile Patterns P1, P2, P3, P4…

Step 2 Allocate and populate an Array, Map and List for storing allowed and disallowed values –

```
private final Map<String, List<String>> vAllowed = vAllowed = new HashMap<String, List<String>>();
private final Map<String, Integer> vTagCounts = new HashMap<String, Integer> ();
private final String[] vSelfClosingTags = new String[]{"img"};
private final String[]  vNeedClosingTags = new String[]{"a", "b", "strong", "i", "em"};
private final String [] vDisallowed = new String[]{};
private final String[] vAllowedProtocols = new String[]{"http", "mailto"}; // no ftp.
private final String [] vProtocolAtts = new String[]{"src", "href"};
private final String [] vRemoveBlanks = new String[]{"a", "b", "strong", "i", "em"};
private final String[] vAllowedEntities = new String[]{"amp", "gt", "lt", "quot"};
private final boolean stripComment = true;
private final boolean encodeQuotes = true;
private final boolean alwaysMakeTags = true;  ArrayList<String> a_atts = new ArrayList<String>();
```

Input = Request(s)

Step 3 Create Matcher Final Matcher m = P.matcher(s);

2.1.2 Processing steps-

Step1:

Invoke Escape comments(s)
Matcher m = P1.matcher(s)
If m.find ()
Invoke quote replacement(s)

Step 2:

Make a call to checkTags(s)
Matcher m = P2.matcher(s)
Make a call to checkTags(s)
If m.find () Invoke Process Tags(s)
Process for P3 (P_start Tag) and P4 (P_end Tag)

Step 3:

a. If the tag is opening tag Extract the name, body and ending of start tag(P_StartTag) by group(1),group(2) and group(3)
check the presence of this tag in Allowed tag list
If present Check the 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 allowedAttributes (tag name, param name) Check the presence or absence of this tag in disallowed tag list(in vallowed map) If this tag is allowed
Extract the attribute name which has been retrieved from body of tag If attribute is associated with this tag and if it is present in vProtocolAtts array
then invoke processparamprotocol (param value)

c. Invoke decodeEntities(s) // It will decode the encoded entities as unicode, hexadecimal to decimal

d. validate these entities by invoking validate entity and check entity methods.

Step 4:

return to process tag(s) If this tag name is present in vSelfclosingTag Array end it with **—** /‖ Else If it needs closing tag end it = **—** ‖ Else close the tag

Step 5: Invoke processRemove blanks(s)

Output- Sanitized Request

Table: web applications protected by proposed approach

s. no	Web application	Injection point	Vulnerable to attack	Attack prevention
1	Web application created by us	Guest book	yes	yes
2	Online recruited web application	Form	Yes	yes
3	Online music web application	Search field	Yes	yes
4	My dream home web application	Search field	No	No need

III. Conclusion

Pattern mapping step wise algorithm can detect the typical XSS web attacks that are occurred in multitier web applications. This gives a mechanism to secure java web applications from XSS (Cross Site Scripting Attack) by applying a framework based on pattern matching approach. Some comparisons have been made between existing solutions and the proposed algorithm. We tested the pattern mapping based framework on some java web applications. The strength of the framework is that it can be applied on any existing java web application without source code modification.

References

[1]. Meixing Le, AngelosStavrou, Brent ByungHoon Kang," *Double Guard: Detecting Intrusions in Multitier Web Applications*", IEEE Transactions on dependable and secure computing, vol. 9, no. 4, July/august 2012.

[2]. M. Johns, B. Engelmann, and J. Posegga, ―*XSSDS: Server-Side Detection of Cross-Site Scripting Attacks*,‖ in Computer Security Applications Conference, 2008. ACSAC 2008. Annual. IEEE, pp. 335–344, 2008.

[3]. E.Kirda, C.Kruegel, G.Vigna, and N.Jovanovic,‖*Noxes: A ClientSide Solution for Mitigating Cross-Site Scripting Attacks*‖ Dijon, France SAC'06 April 23-27, 2006.

[4]. A. Klein, ―DOM *based cross site scripting or XSS of the third kind,*‖*Web Application Security Consortium*, Articles, vol. 4, 2005.

[5]. Anley. Advanced sql injection in sql server applications. Technical report, Next Generation Security Software, Ltd, 2002.

[6]. A.K. Ganame, J. Bourgeois, R. Bidou, F. Spies, "*A global security architecture for intrusion detection on computer networks*," Montbeliard, vol. 27, pp.30-47, March 2008,

[7]. G.W Dunlap, S.T King, S.Cinar, M.Basrai, " Enabling intrusion analysis through virtual-machine logging and replay," *Boston, MA, USA*, pp. 211-240, December 2002. http://www.sans.org/top-cyber-security-risks/.