International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 03, March-2015

Detection and Prevention of Intrusions in Multi-tier Web Application using Static and Dynamic Algorithm

Vaibhavi Kamble.[1], Ritupanna Hazra[2] , Rupali Gajbhiv[3] , Prof . Nilesh Thorat[4]
[1]Dept. of Information Technology, BSIOTR, Pune, India
[2] Dept. of Information Technology, BSIOTR, Pune, India
[3] Dept. of Information Technology, BSIOTR, Pune, India
[4] Assistant Professor, Dept. of Information Technology, BSIOTR, Pune, India

Abstract **- In today's world there is enormous use of Internet services and applications. It has become an inextricable part of our lives and makes communication and management of personal information possible irrespective of time and place. To make the increase of applications and data complexity manageable web servers have moved to a multi-tiered design where the web server runs the application front-end and data is outsourced to database. There is a growing need to protect personal data hence Intrusion Detection System is required. This paper proposes an Intrusion Detection & Prevention System that models the network behavior of user session across both the front-end (web server) and the back-end (database). Efforts are made to attack both web servers as well as database individually. Scrutinizing both database and its preceding web request we are able to detect attacks that independent IDS would not be able to identify. For static websites, a well-correlated model is build for effectively detecting different types of attacks. This will be true for dynamic requests as well where both retrieval of information and updates to the back-end database occur using the web-server front-end. This paper focuses on session hijacking attack, brute force attack, MongoDB injection attack MongoDB Null Byte injection attack, cross scripting attack.**

General Terms **- Session hijacking attack, Brute force attack, MongoDB-injection attack, Cross scripting attack.**

Keywords **- Intrusion Detection System, Intrusion Prevention 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. Most of the daily tasks such as social networking, travelling, banking 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 Social networking and e-commerce sites and other web portals are increasing day by day which in turn increase the frequency of cyber-attacks along with the growth of web services and web applications. Efforts are made by these web attacks to access secure data with an endeavor of interception of unauthorized data over an information technology infrastructure. Such web attacks popular nowadays include Injection attack, Denial-of-Service attacks, Session Hijacking attack and many more.

Intrusion detection System (IDS) is generally used to protect web applications. This system detects known attacks by matching misused traffic patterns or signatures. 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. 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 database and the web server For example, when an intruder enters into a web server as a normal user but by using web server 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 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 removed, deleted 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. Looking from a logical viewpoint each session has dedicated web servers and isolated from other sessions which 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.

AppSecure 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 queries related to the client request to the data base server to retrieve or update data depending on the HTTP request.

AppSecure 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 Rq and has associated database query DQ. Web server receives response from database as DR then web server sends response to client as Rs. This whole transaction is isolated in one session it called as a container and it is denoted by VE in Fig. 1.

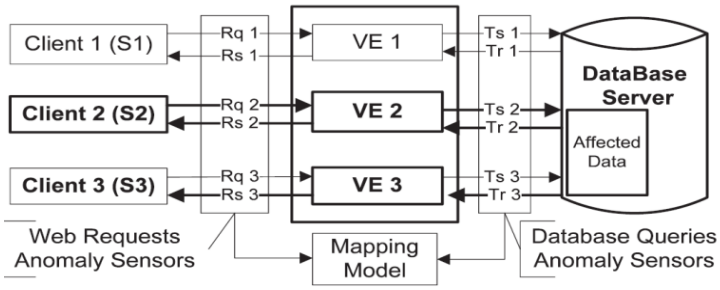

Fig: Normality Model

2. LITERATURE REVIEW

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 MongoDB Injection, Cross Site Scripting (XSS) are more dangerous to web applications. This OWASP present top ten list of web applications vulnerabilities, in this attack MongoDB Injection, and Cross Site Scripting attacks are included.

Before this more work done on the security of the web applications. Based on the 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.

Attack detection method is on the basis of malicious score and reported anomalous queries, this method called as detection method.

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. And easily find out attacks using this system.

In this approach first detailed characterization of web application is done by defining web application internal state 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. Working of this system takes place in two modes fist is training and second detection. At the time of training mode attack signatures are generated and in detection mode this signatures are used to detect attacks.

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 and user communication when a doubtful event occurs which turns as a disadvantage of this tool.

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.

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.

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 .Other Server Side solution also has some E-guard algorithm approach, there is no system to handle scripts which are stored in Grey list, hence 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.

3. APPSECURE MODEL

AppSecure builds the normality model to detect various attacks like Injection, Session Hijacking. 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:

ri : request for any session ‘i’.

Qi : query set for session ‘i’.

ϕ : empty set.

QT : query for all sessions.

Table 1. *AppSecure* Pattern Mapping Techniques.

Sr.No	Pattern Name	Description
1	Deterministic	$r_i \rightarrow Q_i$
2	Empty Query Set	$r_i \rightarrow \phi$
3	No Matched Request	$\phi \rightarrow Q_i$
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. AppSecure 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 ri and set of queries Qi. If N are the total number of session, We have total web request REQ and queries across all sessions. In DM Web request ri appears in all traffic with the queries set Qi. The mapping patterns is then ri → Qi .For any session in the testing phase with request ri, the absence of a query set Qi matching the request indicates a possible intrusion. On the other hand, if Qi is present in session traffic without the ri, 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: ri → ϕ. 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 and 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 ϕ → Qi. 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. There exists a pool of query sets, so every time the same type of web request arrives, it matches up one of the query sets in the pool. The mapping pattern is denoted as ri → QT Therefore, it is difficult to identify traffic that matches this pattern.

AppSecure 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 rm 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.

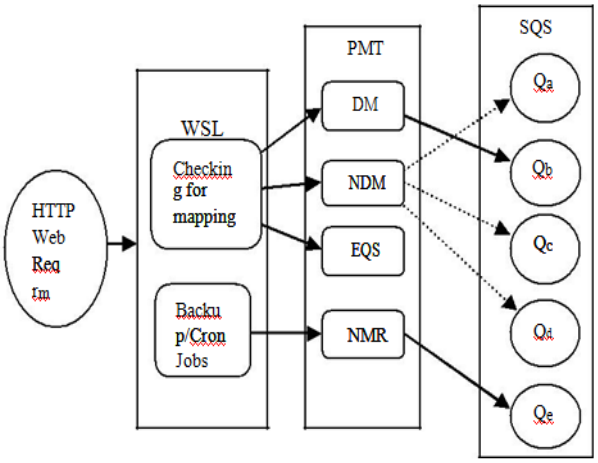

Fig.2 AppSecure Pattern Mapping Architecture

4. APPSECURE METHODOLOGY

In *AppSecure* 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 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. *AppGuard* easily detect this attack also by using mapping model.

4.2 Brute Force Attack

A password attack that does not attempt to decrypt any information, but continue to try different combinations for passwords. For example, a brute-force attack may have a dictionary of all words or a listing of commonly used passwords. A brute force attack tries all words it has to gain access to the account . Another type of brute-force attack is a program that runs through all letters or letters and numbers until it gets a correct match. Eventually a brute-force attack may be able to gain access to an account however, these attacks can take several hours, days or even months to run. The time taken to complete these attacks is dependent on how complicated the password is and how well the attacker knows the target.

To help prevent brute-force attacks many systems will only allow a user to make a mistake in entering their username or password three or four times. If the user exceeds the limited number of attempts provided, the system will either lock them out of the system or prevent any future attempts for a set amount of time

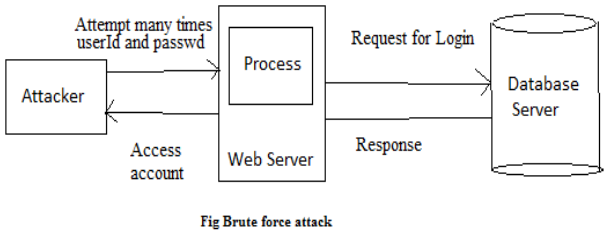

Fig 3.: Brute Force Attack

4.3 MongoDB Injection Attack

Its a common misconception that as MongoDB does not use SQL it is not vulnerable to SQL injection attacks. Objects are used in PHP rather than SQL to pass queries to the MongoDB server; for example the following script selects an item form MongoDB where the username equals 'bob' and the password equals 'password'.In a normal injection attack we can replace either of the two input parameters with a string such that the query always returns true. That wont work with MongoDB; however if we can pass in an object to the PHP MongoDB driver we could alter the query in a similar fashion. PHP provides a way to pass objects as GET or POST parameters.

4.4 MongoDB Null Byte Injection Attack

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

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.

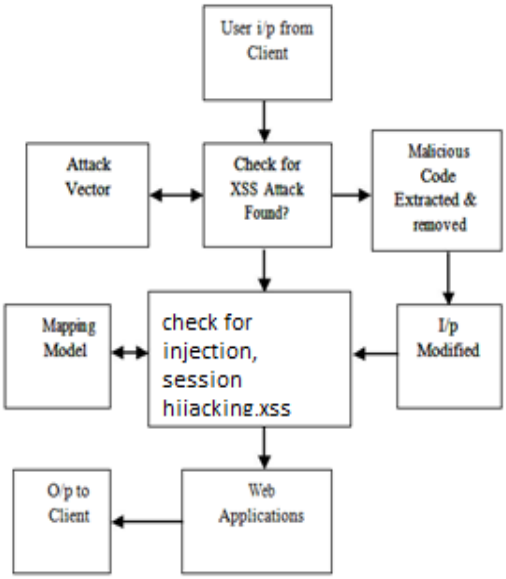

Fig.3 AppGuard attack flow architecture.

Websites normally utilize scripts written in JavaScript coupled with HTML, which runs on a client side depiction application for faultless user experience. 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 *AppGuard*, 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, *AppGuard* 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 and Session Hijacking, attack pattern mapping algorithm is presented here.

5. APPGUARD ALGORITHMIC STRATEGY

AppGuard protect web application from attacks like Injection, Session Hijacking and XSS. So it provides various algorithms for that, XSS attack algorithm used for XSS attack detection and prevention. Algorithm uses attack vector, once attack is detected it is removed from the input value. For detection of Injection and Session Hijacking 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.Intrusion detection algorithm is used for detection of these attacks.

Algorithm 1. Static Model Building Algorithm
Require: Training Data set, Threshold t Ensure: The Mapping Model for static website
1: for each session separated traffic Ti do
2: Get different HTTP requests r and DB queries q in this session
3: for each different r do
4: if r is a request to static file then
5: Add r into set EQS
6: else
7: if r is not in set REQ then
8: Add r into REQ
9: Append session ID i to the set ARr with r as the key
10: for each different q do
11: if q is not in set SQL then
12: Add q into SQL
13: Append session ID i to the set AQq with q as the key
14: for each distinct HTTP request r in REQ do
LE ET AL.: DOUBLEGUARD: DETECTING INTRUSIONS IN MULTITIER WEB APPLICATIONS 519
15: for each distinct DB query q in SQL do
16: Compare the set ARr with the set AQq
17: if ARr ¼ AQq and CardinalityðARrÞ > t then
18: Found a Deterministic mapping from r to q
19: Add q into mapping model set MSr of r
20: Mark q in set SQL
21: else
22: Need more training sessions
23: return False
24: for each DB query q in SQL do
25: if q is not marked then
26: Add q into set NMR
27: for each HTTP request r in REQ do
28: if r has no deterministic mapping model then
29: Add r into set EQS
30: return True

Detection for Dynamic Websites

Once we build the separate single operation models, they can be used to detect abnormal sessions. In the testing phase, traffic captured in each session is compared with the model. We also iterate each distinct web request in the session. For each request, we determine all of the operation models that this request belongs to, since one request may now appear in several models. We then take the entire corresponding query sets in these models to form the set CQS. For the testing session i, the set of DB queries Qi should be a subset of the CQS. Otherwise, we would find some unmatched queries. For the web requests in Ri, each should either match at least one request in the operation model or be in the set EQS. If any unmatched web request remains, this indicates that the session has violated the mapping model.For example, consider the model of two single opera-tions such as Reading an article and Writing an Article. The mapping models are READ ! RQ and

WRITE ! WQ, and we use them to test a given session i. For all the requests in the session, we then find that each of them either belongs to request set READ or
WRITE. (You can ignore set EQS here.) This means that there are only two basic operations in the session, though they may appear as any of their permutations. Therefore, the query set Qi should be a subset of RQ [WQ, which is CQS. Otherwise, queries exist in this session that do not belong to either of the operations, which is inconsistent with the web requests and indicates a possible intrusion. Similarly, if there are web requests in the session that belong to none of the operation models, then it either means that our models haven't covered this type of operation or that this is an abnormal web request. According to our algorithm, we will identify such sessions as suspicious so that we may have false positives in our detections.

AppGuard can be applied on web application created by us, it able to detect attacks like session hijacking attack, brute force attack, MongoDB-injection attack and Cross Site Scripting (XSS). *AppGuard* also able to prevent some attack like Session Hijacking, Cross Site Scripting (XSS).

6. CONCLUSION

The proposed *AppGuard* as an intrusion detection system detects typical web attacks like MongoDB Injection, Session Hijacking, MongoDB Null Byte injection, Brute force and XSS (Cross site scripting attack) that occur in a multi-tier web application. *AppGuard* 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 *AppGuard* 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.