www.ijaret.org Volume 1, Issue II Mar. 2013
ISSN 2320-6802

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY

WINGS TO YOUR THOUGHTS…..

Intrusion Detection Using Multitier Web Application Analyzer

J.Rethna Virgil Jeny [1], Deepa Baban Ekhande[2]

[1]Associate Professor, Dept. of IT, AVCOE Sangamner, Maharashtra
jenyvir20@yahoo.com

[2]Student, M. E. IT, AVCOE, Sangamner, Maharashtra.
ekhandedeepa@gmail.com

Abstract: *Web services and applications have smoothened the way of users for the faster data access. To meet the user's requirements for faster data retrieval, the web applications have moved towards multitier design wherein the back-end server contains the data base and application interface or web server acts as the front end. The wide-spread use of web services has made them a target of attackers. In this paper, we are using the multitier web application analyser for securing the data storage at the back-end as well as the front-end applications. The multitier web application analyser protects the behaviour of the sessions performed by the users from the huge range of attacks. It helps in monitoring the web applications as well as database requests to safeguard the attacks that would not be done by an independent intrusion detection system. Therefore, enhances the system performance to provide high efficiency.*

Keywords: *Multitier web analyzer, Virtualization, container- based architecture, anomaly detection.*

1. INTRODUCTION

Internet services have been speedily flourished across the globe in relation to its popularity and complexity. The World Wide Web provides varied applications such as the social media, educational, finance etc. But, the wide use of these services has made them reside on the peak of attackers. Different types of Attacks [1] like SQL injection and hijacking session of users are carried out on the web servers where the webserver is taken over by the attacker. Hence, all the subsequent user sessions are also being hijacked.
The anomaly detection systems [2] detect the abnormal behaviors of the system. The network packets are individually analyzed by the intrusion detection systems. But for the system designed to be multitier a very modest task is carried out. Hence, due to the lack of such architecture even if we make use of firewall to guard the database backend, they are likely to be attacked by the attackers who employ the web request for exploiting the back end data. The misused detection [3] can be used for matching the changed patterns in order to prevent the multi tier web services. But, when the network traffic is transfer from the web to the database or vice-versa, the intrusion detection system is unable to identify the anomalous traffic by just using the web intrusion detection system or the database intrusion detection system. For exploiting the vulnerabilities in the web, the attacker uses the nonadmin rights by issuing a privileged database query. This type of attack can neither be discovered by the database intrusion detection system nor by the web intrusion detection system. It is be impossible to detect the casual mapping between the database and the web server with the current web server multithreaded architecture.
Here, a solution to the problem which deals with the intrusion detection and prevention in the multitier web architecture is presented. The multitier web application analyser is being developed for preventing the system from wide number of attacks. Normality Model of the isolated user sessions is being built which consists of the web application front end that is in form of HTTP requests and the data base backend [4] i.e. the SQL or file server. The intrusion prevention system builds up a casual mapping model which is to be done by taking under consideration both the web server and database traffic. The intrusion prevention system is the conservatory of intrusion detection system. In the intrusion prevention system, the vulnerabilities are being blocked and therefore do not affect the system or network any longer. Here, we present a lightweight virtualization model for assigning every user's session to a container. To specifically relate the web request with the consequent database queries we formulate use of the container ID. The IPS builds a casual mapping profile by undertaking the web server and database traffic. OpezVZ [5] is used for implementing the IPS container architecture. This container based architecture along with casual mapping also provides an

isolation which helps in safeguarding the session hijacking attacks of the users. The lightweight virtualization [6] helps in running innumerous copies of the web server instances in different containers. Hence, everyone is differentiated from the rest. Each client is assigned a dedicated container so that even when the attacker attacks the session, it is limited to that session only and doesn't damage the other user session.

2. EXISTING WORK

Person At present, the Intrusion Detection System is a device or software application that helps to monitoring network and system vulnerabilities. It generates reports for a management station. The anomaly based detection requires the system to describe and organize the right and acceptable form and nature that can be used for finding the behaviours that are different from others. Another approach is the Intrusion Detection Alert correlation [7] method which involves combination of all security alert events like the replicated alerts, false alarms and other non relevant data to the users. But, in place of correlating the alerts from the individual detection system our methodology works on multiple feed of traffic in the network involving the use of just single IDS. The IDS uses temporal information [8] to detect intrusions that needs to correlate events on time basis which increases the threat to the system. The Intrusion Prevention Systems are the expansion of the intrusion detection system where they not only monitor the network traffic and system activities but also consign in-line and are able to actively prevent or obstruct the intrusions that are affected.

3. SYSTEM ARCHITECTURE

3.1 Normality Model Based Architecture

The network traffic from both legal and illegal users is received at an alike web server. The attacker can affect the future sessions by compromising the web server. It is not a legitimate option to allot each session to a dedicated web server, since it may diminish the web server. Hence to gain similar internment when maintaining a low performance and overhead of resources, light-weight virtualization is used. The light-weight process containers that are disposable servers for the client are being used. Thousands of containers can be initialized only on a single machine. These containers can be terminated, reverted or reinitialized for every new client session. Here every session assigned to a dedicated web server is isolated from every other session. The initialization of each virtualized container can be completed using read-only template that is presumed to be clean and hence gives assurity that at initialization every session is being served with a clean web server instance. The communication statement of a single user is expected to go to the alike dedicated web server. Different users can be shown by the sessions to some extent where by permitting us to recognize the suspectable behaviour by the session and user. We will treat the traffic with the session as infected if we sense any abnormal behaviour in a session. The course of information flow is being separated in every container session. This offers us with high security performance. It also maps all the web server requests with the database queries. The normality model shown in fig. 1 helps us to map the relationship between the unauthorised user accesses. The normality model divides the session of the users and thus helps to detect the attacks. It is possible to recognize the web requests and the resulting SQL queries and hence possible to evaluate them in the session. Sensors are placed on both sides of the server which detect anomalies by capturing the traffic at both the end.

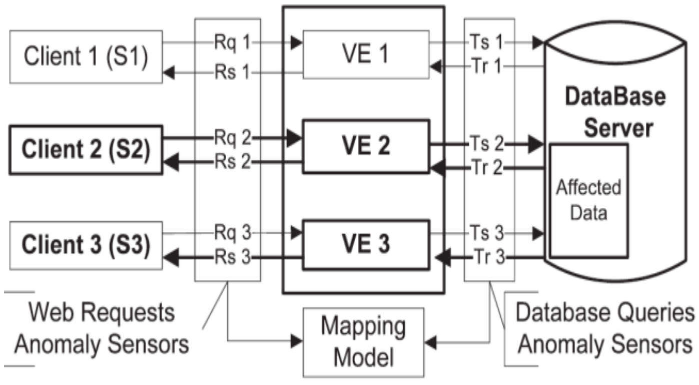

Figure.1 Normality Model

Types of Attacks
SQL Injection Attack

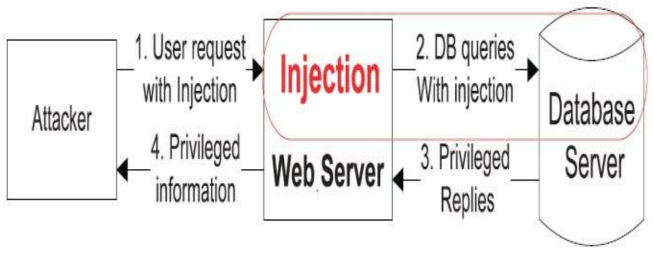
Attacker
1. User request with Injection
4. Privileged information
Injection
Web Server
2. DB queries With injection
3. Privileged Replies
Database Server

Figure.2 SQL Injection Attack

SQL injection attack [9] does not need compromising the web server. The vulnerabilities present in the web server logic are used by the attackers for injecting the exploited data content in order to attack the backend database. Even though the exploits are accepted by the web server, as our approach presents a two tier detection, the contents which are related to the database server are unable to take the expected structure for the given web server request. Even if the injected data passes through the web server side, the SQL queries are produced in a different manner which can be detected from normally following SQL query structure.

3.1.1 Privilege Escalation Attack

In this, we suppose that the website is serving both the administrators and the regular users. For an administrator, the request is triggering the set of admin level queries and for the regular users; the web request triggers the set of SQL queries. The attacker marches into the webserver as a normal user, raises the user privileges and obtains the admin queries to get an admin data. The attack of such type can never be detected by the webserver or the database intrusion detection system as both are valid queries. Our approach uses mapping model for detecting such type of attacks because here the database query does not match the request according to the mapping model.

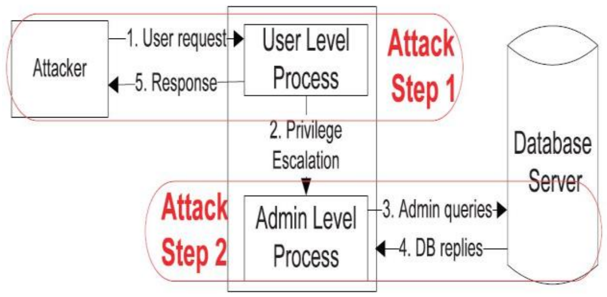

Figure.3 Privilege Escalation Attack

3.1.2 Hijack Future Session Attack

This type of attack happens mostly at the web server side. Attacker invades over the webserver and hijacks all resulting unauthorized user sessions to initiate the attacks. By hijacking other user session for a period of time, the attacker can send or drop replica of messages or replies or even drop user requests.

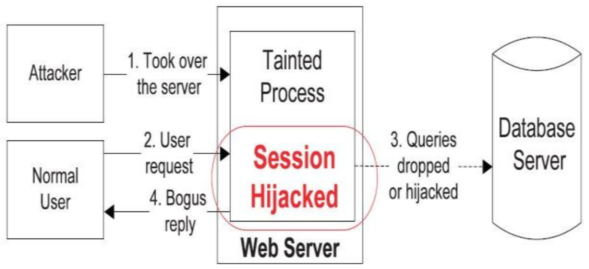

Figure.4 Hijack Future Session Attack

3.1.3 Direct DB Attack

There is a chance that an attacker can bypass the webserver or firewall and directly get connected to the database. The attacker can also surrender queries from the webserver without sending web request by taking over the webserver. So for such queries, with the unmatched web request, webserver IDS is not able to detect them. In addition, if the database queries are within the set of valid queries, the database IDS would itself not be able to identify the attack.

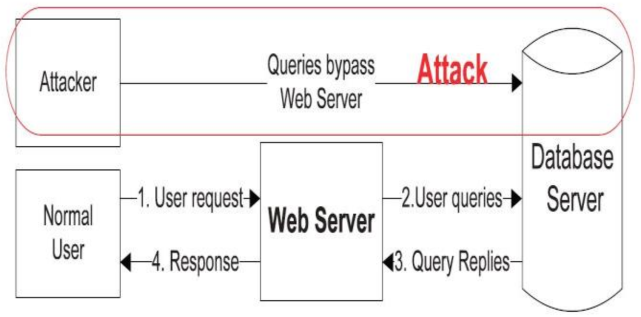

Figure.5 Direct DB Attack

3.1.4 Man In Middle Attack

Here, the attacker looks out to create independent connections with the one who is prone to be vulnerable. But it is not able to authenticate itself with the verifier which has no clue of the fingerprint of the two end nodes

4. MODELLING DETERMINISTIC MAPPING AND PATTERNS

For static website, the links are static wherein clicking on the same link will always return the same information. Hence we can build an accurate model for mapping relationships between web request and database queries. In place of one-to-one mapping relationship between database queries and web request there can be numerous SQL queries for single web request. For some cases, it may happen such that no queries will be generated by the web request i.e. some of the requests will just retrieve data from web server. On the other hand, one request may let that the web server may invoke number of queries. Hence, we organize the mapping patterns into four classes given below. The request is at the source of the data flow. Consider a set of request r_m and set of queries Q_p. The total number of sessions in the training phase is N. We can then achieve the set total web request *REQ* and set of SQL queries *SQL* across the whole session. The mapping of the model is in form of one request to a query set r_m->Q

Deterministic Mapping

This form is most common type mapping and perfectly matched pattern. Here, the web request r_m appears in all traffic with SQL query set Q_p. In case, there is an absence of query set Q_p for any session in the testing phase with request r_m, it indicates that intrusion is present.

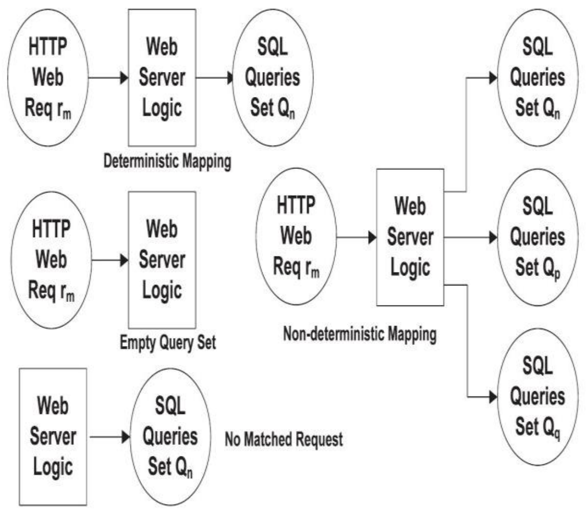

Figure.6 Representation of Mapping Patterns

Empty Query Set
It is a special case i.e. EQS wherein the SQL query can be an empty set. It indicates that the web request can neither cause nor generate any database queries. The mapping of this kind can be r_m->ø. Here web request are put together in the set EQS during the testing phase. Consider an example, that when a web request for retrieving a GIF image file is made from the identical webserver, then a mapping relationship doesn't exist as only the web requests are observed.

4.1 No Matched Request
Sometimes the queries cannot match-up with the web requests. These unmatched queries are placed in a set NMR. Any query in a set NMR is considered to be legitimate. The NMR size depends on the webserver logic but it is typically small.

4.2 Non Deterministic Mapping
The identical web requests can arrive each time. It matches with one query inside the pole of query set. The mapping pattern can be r_m-> Q_i (Q_i € {Q_n, Q_p, $Q_{q...}$}). So it is difficult to classify the matched traffic pattern. This happens mostly in case of dynamic websites

5. STATIC WEBSITES MODELLING

The non-deterministic mapping is not present for static websites as for this no input variables or states are available for the static content. The traffic gathered by servers can be organized into three patterns to create a mapping model. As the traffic is already isolated by sessions, we commence by iterating all the sessions from 1 to N. For the request which belongs to REQ, we maintain a list AR_m to record the session ID's in which r_m appears. For the database queries we enclose set AQ_s for every query which belong to SQL to record the session ID's. We look for AQ_s for each AR_m. If AR_m= AQ_s then it signifies that every time r_m appears in a session, Q_s will also appear in that same session. To extract a mapping pattern r_m->Q_s confidentially, threshold value t is used, so that if in case the mapping appears in more than t times sessions then a mapping pattern is been acknowledge. If the patterns are not as much as the threshold t, then this indicates that the number or training session is insufficient.

Model Building Algorithm

```
• for each session separated traffic $T_i$ do
•     Get different HTTP requests r and DB queries $q$ in
  this session
•       for each different r do
•           if $r$ is a request to static file then
•               Add r into set $EQS$
•           else
•               if $r$ is not in set $REQ$ then
•                   Add $r$ into $REQ$
•               Append session ID $i$ to the set $AR_r$ with
  $r$ as the key
•       for each different $q$ do
•           if $q$ is not in set $SQL$ then
•               Add $q$ into $SQL$
•           Append session ID $i$ to the set $AQ_q$ with $q$ as the
  key
• for each distinct HTTP request $r$ in $REQ$ do
•   for each distinct DB query $q$ in $SQL$ do
•     Compare the set $AR_r$ with the set $AQ_q$
•       if $AR_r = AQ_q$ and Cardinality($AR_r$) > t then
•           Found a Deterministic mapping from r to q
•           Add q into mapping model set $MS_r$ of r
•           Mark $q$ in set $SQL$
•       Else
•           Need more training session
• return False for each DB query q in SQL do
• if q is not marked then
• Add q into set NMR
• for each HTTP request r in REQ do
• if r has no deterministic mapping model then
• Add r into set EQS
• return True
```

Detecting Intrusions
After the normality model has been generated it can be worked out for training and detection of the abnormalities. Each session is then compared to normality model. We measure the web request with the mapping rule. Each discrete web request in the session will consist of only one mapping rule in the model. It can be done using the following algorithm:

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS…..

Intrusion Detection Algorithm:
1. In the deterministic mapping rule, where r->Q, we check whether the query Q is a subset of the query set of the session. If it is there, then we mark those queries in Q as a valid request.
2. If the EQS i.e. r->ø, rule persists then the request is not being considered to be abnormal and we do not require to mark any database queries. So, no intrusion is being reported.
3. If the left out queries are unmarked, then there is a need to look out whether they are in set NMR. If it is so then they are marked as such.

The queries which are new or unmarked are considered to be abnormal. If it exists within a session then that session will be marked as vulnerable.

6. FUTURE WORK

The cloud data storage tenders a great convenience to the users since there is no time to care about the complexities of hardware management. The inexpensive and powerful processors, along with the software as a service computing architecture are transforming the data centres into pools of computing devices. Due to enhancement in bandwidth and flexible network giving high reliability it feasible for users to pledge high quality services form software and data that is residing onto the remote data centres. From the point of view of data security, which is a vital aspect of the quality of service, cloud computing certain generates new challenging security threats for various reasons.

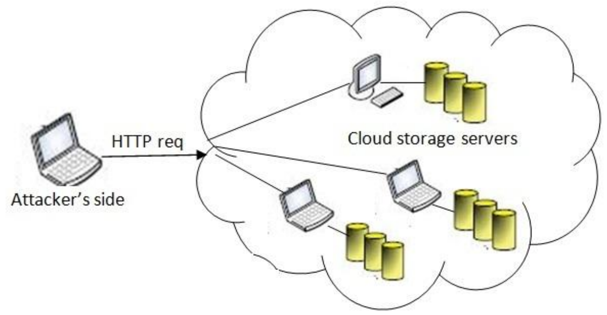

Figure.7 Attack Scenario in Cloud data storage

A distributed protocol is most essential in achieving a robust and safe cloud data storage system. For the cloud data, the attacker can aim to send the request to any system that is in the network. This can prove to be unsafe for the system's working. The attacks like the SQL injection use the database queries with injection to database server. So there is a need to defend these systems in the cloud storage. The multitier web analyzer must be designed in accordance that it must be able to detect intrusions of the server's front end as well as back end. The SQL filter algorithm can be put up for this purpose.

7. CONCLUSION

The multitier web analyser is developed to model the behaviour of the web applications. To detect the anomalous behaviour of the multitier web applications both at the front end as well as the back end data, the mapping model is used. The container-based architecture provides diverse session ID's for different HTTP requests which is helpful in isolating the information flow of all webserver sessions. Thus, the multitier web analyser is able to recognize wide range of attacks invading the system.

References

[1] "Five Common Web Application Vulnerabilities," http://www.symantec.com/connect/articles/five-common-web-applicationvulnerabilities,2011
[2] H. Debar, M. Dacier, and A. Wespi, "Towards a Taxonomy of Intrusion-Detection Systems," Computer Networks, vol. 31, no. 9, pp.805-822, 1999.
[3] H.-A. Kim and B. Karp, "Autograph: Toward Automated Distributed Worm Signature Detection," Proc. USENIX Security Symp., 2004 Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.
[4] G. Vigna, F. Valeur, D. Balzarotti, W.K. Robertson, C. Kruegel, and E. Kirda, "Reducing Errors in the Anomaly-Based Detection of Web-Based Attacks through the Combined Analysis of Web Requests and SQL Queries," J. Computer Security, vol. 17, no. 3, pp. 305-329, 2009.
[5] OpenVz, http:/wiki.openvz.org, 2011.
[6] Y. Huang, A. Stavrou, A.K. Ghosh, and S. Jajodia, "Efficiently Tracking Application Interactions Using Lightweight Virtualization," Proc. First ACM Workshop Virtual Machine Security, 2008.
[7] F. Valeur, G. Vigna, C. Kru¨ gel, and R.A. Kemmerer, "A Comprehensive Approach to Intrusion Detection Alert Correlation," IEEE Trans. Dependable and Secure Computing, vol. 1, no. 3, pp. 146-169, July-Sept. 2004.
[8] Seleznyov and S. Puuronen, "Anomaly Intrusion Detection Systems: Handling Temporal Relations between Events," Proc. Int'l Symp. Recent Advances in Intrusion Detection (RAID '99), 1999.
[9] Anley, "Advanced Sql Injection in Sql Server Applications, technical report, Next Generation Security Software, Ltd., 2002.