Intrusion Detection System in Web Services

Muthu Kumara Raja[1], Bala Sujitha.T.V[2]

[1,2]Student, Assistant Professor
Department of Computer Science & Engineering,
Hindustan University, Chennai, India
[1, 2]kumar16jun@gmail.com, balastv@hindustanuniv.ac.in

Abstract: ***Today, web applications have become a ubiquitous use of daily tasks, such as banking, travel, and social networking, etc., Due to their ubiquitous use for daily tasks, web applications have always been the target of attacks. In the existing system, communication between the web server and the database server is not separated, and can hardly understand the relationships among them. We propose an efficient IDS system as Double Guard with lightweight virtualization that models the network behavior for multi-tiered web applications of user sessions and able to identify ferret out attacks. Casual Mapping profile model is newly developed to map between the web server requests and the subsequent DB queries, identifies the various types of attacks and minimize the false positives for both static and dynamic web application.***

Keywords: Attacks, Lightweight Virtualization, Multitier Web Application, SQL Injection, Vulnerable.

1. Introduction

Web applications have become a ubiquitous use of daily tasks, such as banking, travel, and social networking, etc., to manage this web applications have moved to multi-tiered architecture.

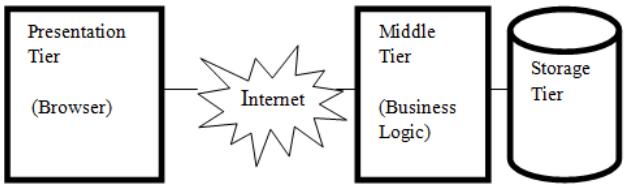

Figure 1: Multi-tiered Client/Server Architecture

Applications are usually broken into logical chunks called "tiers", where every tier is assigned a role. Existing applications consist only of 1 tier, which is the client machine only, but web applications manage the daily tasks to move an n-tiered approach. The structure is the similar to three-tiered application. The three tiers are called *presentation*, *application* and *storage*, in this order Figure1. A web browser is the first tier (presentation), dynamic Web content technology is the middle tier (application logic), and a database is the third tier (storage).Due to their ubiquitous use of daily tasks, web applications have always been the target of attacks [10].Attacks can be generally categorized into two types of attacks. They are active attacks (attempt to destroy/alter) and passive attacks (attempts to learn not alter). Let discusses with various positions affected in the multi-tiered web application.

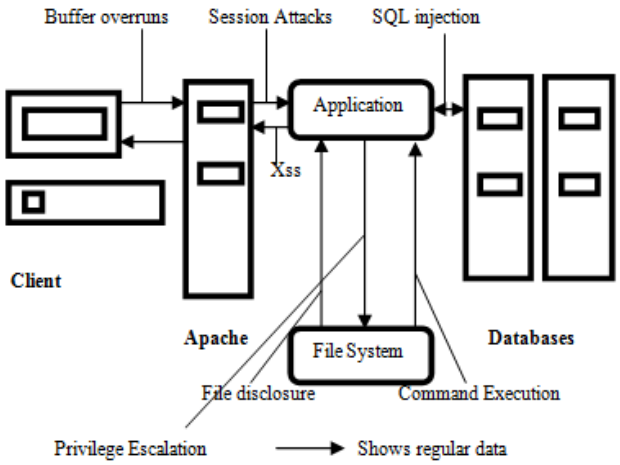

Figure 2: Positions affected by various attacks

In the multi-tiered web application attacks are done various places. In the attacks are classified into two categories. They are web server side or database server side. Figure2. Clearly explains the various positions of attacks captured and performed well.

- Session Hijacking Attacks
- Cross site Attacks
- Privilege Escalation Attack
- SQL Injection Attack
- Buffer overruns

Session hijacking attacks are mainly happening at the web server side. The intruder hijacks the web server and all legitimate user sessions to launch attacks with dump/rogue reply. Cross site scripts occur when a web application sends user data to a web browser without first encoding or validating it. XSS permits third-party to pass a script as user data that runs in the user's browser.SQL injection mainly focused on back end. SQL injection is a technique often used to attack the databases by inserting the dump SQL statements. It injecting malicious SQL commands into the content of the parameter, the attacker can trick into the web application into forwarding a malicious (rogue) query to database. It corrupts the database, destroy or alter the database content. A privilege escalation attack is mainly aimed at web server. It takes advantage of

programming errors or design flaws. Buffer overflows attacks done in front end. It causes the denial of service and remote command execution. Similarly file disclosure and command executions are attacks a web server side. Observing ongoing attacks and using this filter out future occurrences of these attacks.

In the present system depicts how communications are categorized as sessions using IP address [2], and how database transactions can be related to a corresponding session. Virtualization is used to isolate objects and enhance security performance. Lightweight virtualization are use to isolate the object such as OpenVZ [5], [6]. It is based container concept. On the other hand, lightweight containers can have considerable performance advantages over full virtualization or Para-virtualization. Thousands of containers can run on a single physical host. It is an efficient Doubleguard system implement using Apache web server with My Sql that monitoring the network behaviors of the front ends as well as back end system. It's able to capture the various types of attacks, with the help of isolation and lightweight virtualization techniques.

DoubleGuard [6], utilized the container ID to separate session traffic as a way of extracting and identifying causal relationships between web server requests and database query events. Use the container ID matching the web request to DB queries accurately. Thus doubleguard can build a casual mapping profile by taking both the web server and DB traffic into account, and minimize the false positives.

2. Related Work

The existing system, classic three-tier model at the database side, unable to tell which transaction corresponds to which client request. The communication between the web server and the database server is not separated, and can hardly understand the relationships among them.

Bryan Parno et.al.[7], proposed CLAMP is architecture for preventing data leaks even in the presence of attacks. By isolating code at the web server layer and data at the database layer by users, CLAMP guarantees that a user's sensitive data can only be accessed by code running on behalf of different users. In contrast, DoubleGuard focuses on modeling the mapping patterns between HTTP requests and DB queries to detect malicious user sessions. There are additional differences between these two in terms of requirements and focus. CLAMP, requires modification to the existing application code, and the Query Restrictor works as a proxy to mediate all database access requests. Moreover, resource requirements and overhead differ in order of magnitude: DoubleGuard uses process isolation whereas CLAMP requires platform virtualization, and CLAMP provides more coarse-grained isolation than DoubleGuard.

Giovanni Vigna et.al.[8], proposed reducing the errors web based attacks combined through web requests and Sql queries, The system composes both web IDS and database IDS to achieve more accurate detection, and it also uses a reverse HTTP proxy to maintain a reduced level of service in the presence of false positives. However, heavy traffics not detect by certain types of attacks either wed IDS or database IDS. In such cases, there would be no alerts to correlate.

D.Bates et.al.[3], proposed regular expressions in client side XSS Filters, Assume the attacker's goal is to run arbitrary script in the user's browser with the privileges of the target web site. Typically, an attacker will run script as a stepping stone to disrupting the congeniality or integrity of the user's session with the target web site. The attacker can inject script into a website, and the intruder can induce the user into actions. In this paper, attackers who seek to success their goals with zero interaction. A new alters design that enhance both high performance and high precision under blocking scripts after HTML [4], parsing but before execution. Compared to previous approaches, approach is faster, protects against more vulnerability, and is harder for attackers to abuse, have contributed an implementation of alter design to the Webkit open source rendering engine, and the alter is now enabled by default in the Google Chrome browser.

2.1 Problem Statement:

Existing Intrusion Detection system (IDS), network packets individually monitoring by web server either database side only. It doesn't construct virtual environment, so not able to provide Security in Web Application. Not isolate information flow in each Container Session can hardly identify among the relationship. It's not able to detect different types of attacks, and not able to enhances the security in web application.

3. Proposed System

The proposed System communications are separated as sessions and database transactions can be related to a corresponding session. Sessions are separated using Session ID and IP Address [2]. Session is an activity that a user with unique IP Addresses spends on a website during a specified period of time. Amount of traffic measuring depends upon the number of users on a website.

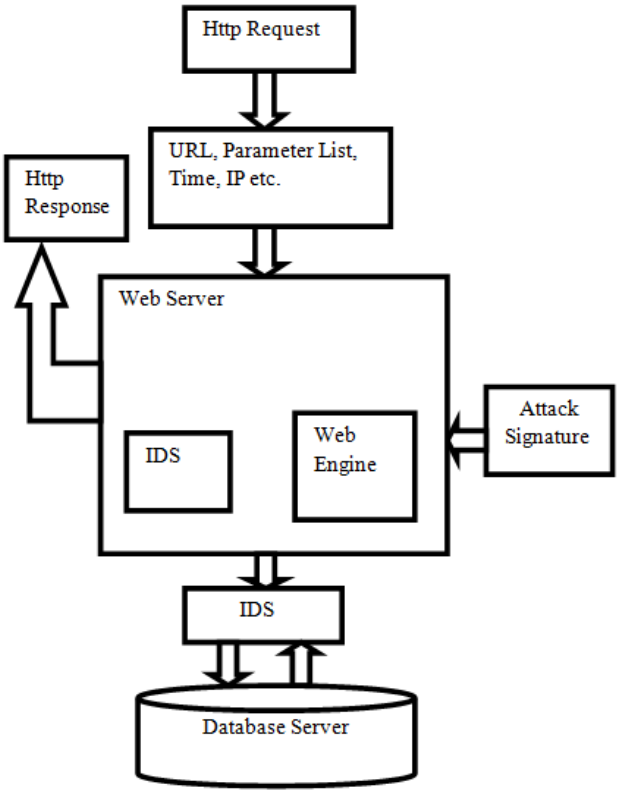

Figure 3: Proposed Architecture Diagram

3.1 Proposed Architecture Description:

The Client (End user) accesses the websites via request such as Http request (URL), parameter list, IP address. In the requests are receives by the web server. Goal of the system is detect the attacks in front end and back end and produce to alerts. Figure.3, the front end such as act as web server, in the server identifies the attacks using IDS and minimizes the false positive. Similarly monitoring the network behaviors at the back-end (Database) Server before receives request, its may be web server or direct request (without) causing web server to the database. In the back end act as Database server, it's identified the attacks using IDS and minimizes the false positive at the back end. In this system suppose any intruder try to access the web application, mapping model capture the Intruder with the help of doubleguard. In Doubleguard model only allows legitimate users can access the web application and achieves the efficiency.

3.2 Attacks Selection:

The proposed system is capturing the various types of attacks. But in this paper we consider SQL injection attacks and Cross Site Scripting Attack. Because this two types of attacks only occurs 80% in web Application.

3.2.1 Sql Injection Attack:

SQL injection [6], is a technique often used to attack the databases by inserting the dump SQL statements. This is done by inject the portions of rogue SQL statements into web form field to pass the dump SQL commands to database. SQL injection code exploits vulnerability in a website's. SQL statements are injected from the web form into database application to dump the database information like passwords, credit cards to the intruder. It is mostly known as attack vectors for websites and used for any types of SQL database attacks.

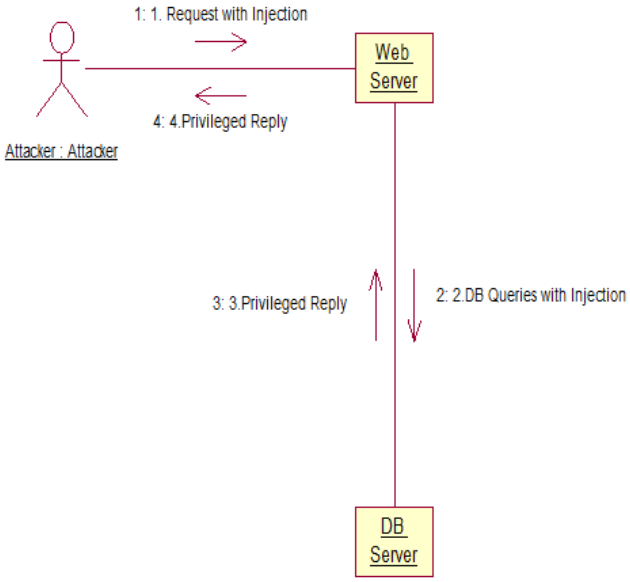

Figure 4: Collaboration Diagram for Sql Injection Attack

Figure.4. Shows how a dump queries to pass the database server [1]. It may be change database content or dump the database information via injecting the newly formed rogue SQL command.

3.2.1.1 Observations of SQL Injection Attack:

They are four main kinds of SQL injection attacks against database.

SQL Manipulation: manipulation is the process of modifying by the SQL statements using various operations such as UNION, WHERE.

Code Injection: Code injection is process of inserting new dump SQL query into vulnerable DB query.

Function Call Injection: Function call injection is process of inserting various DB function calls into a vulnerable DB query.

Buffer Overflows: By using function call injection the buffer overflows is been attained. For most of the commercial and open sources databases, patches are available.

3.2 Cross Site Scripting Attack:

Cross site scripting attack [9], is similar to SQL injection. It injects the dump scripts into the unsuspecting user.

3.2.1 Observations of XSS Attack:

XSS attacks are broadly classified into 2 types:
1. Non-Persistent (reflected)
2. Persistent (Storage)

3.2.1.1Non-Persistent XSS Attack:

Non-Persistent attack, it requires a user to visit the specially made malicious link by the attacker. When the user visits the link, the malicious code will get executed by the user's browser. Let us understand this attack better with an example.

Example for Non-Persistent XSS:

index.php:
<?php
$name = $_GET['name'];
echo "Welcome $name
";
echo "Click to download";
?>

Now the intruder will click an URL as follows and send it to the victim.
index.php?name=guest<script>alert ('attacked') </script>

When the above URL loads into the browser, he/she will see an alert box which says 'attacked'. Even though this example doesn't do any damage, other than the alerts box show to "attacked".

3.2.1.2 Persistent XSS Attack:

Persistent attack, the code injected by the attacker will be stored in a secondary storage device. The damage attained by persistent attack is more than the non-persistent attack.

Example of Persistent XSS:

Transfer of a cookie:
<img src="image.jpg">
<Script>
document.images[0].src="http://evilserver/image.jpg?Stolencookie="+document.cookie"
</script>

This code contains the malicious javascript code, which stored in database. The user's browser then executes the malicious script, which in turn, sends the user's cookie to a server by the attacker's control. The goal is to ensure that a javascript program can send *Sensitive Information* only to the site from which it was loaded.

4. Mapping Model

Build an accurate model of the mapping relationships between web requests and database queries since the links are static and clicking on the same link always returns the same information but the links are dynamic information can differ. In Proposed System IDS used to detect the attacker basis on mapping models and it achieves the *false positives* in Web Application. It classify as the four possible mapping patterns. They are,

Deterministic Mapping: deterministic mapping is web request R_m appears to the SQL queries set Q_n.

$$R_m \rightarrow Q_n (Q_n \neq \phi)$$

Empty Query Set: the SQL query set may be the empty set. This neither implies that the web request nor generates any DB queries.

No Matched Request: the SQL query not match with subsequent web requests.

Non Deterministic Mapping: web request may result in different SQL query.

$$R_m \rightarrow Q_n (Q_n \in \{Q_i, Q_p, Q_s\})$$

5. Attack Detection Using Mapping Model

For example, our approach can detect the SQL injection attacks. We wrote a simple asp login page that was vulnerable to Sql injection attack. As we need legitimate username and password to successfully login. After the legitimate login process we launched an SQL injection attack.

Example:

Generalized Captured HTTP Request:
http://sqlinjection.aspx?username=guess &
password=2%34+or+341%2E1

Generalized captured DB query:
Select * from users (table1) WHERE username (col1) = 'X' AND **password (col2) ='1' or '1'**

Above underlined contents are injected as the dump queries as always 'or '1=1' treated as condition true. However the DB queries received do not match with the mapping model. It also mitigated by input validation and parameterized queries. We establish the mapping model it clearly defining which request belongs to which queries.

6. Conclusion

In this paper, we have proposed efficient IDS system that models the network behavior in multi-tiered web application and builds casual mapping model for identifying various types of attacks and minimize the false positives in both static and dynamic web application. We achieved this with the help of doubleguard with lightweight virtualization (isolated session using session ID) and enhances the security in web application. This is useful in web application such as daily tasks such as banking, travel, and social networks.

References

[1] Ali Bahrami, "Object Oriented Systems development using the unified Modeling Language", Tata McGraw-Hill Edition, pp.36-74, 2008.
[2] Behrouz A. Forouzan, "Data Communication and Networking", Second Edition Update, Tata McGraw-Hill Edition, pp.710-179, 2003.
[3] D.Bates, A.Barth and C.Jackson, "Regular Expressions Considered Harmful in Client-side XSS Filters," Proc.19th Int'l conf. World Wide Web, 2010.
[4] http://www.w3schools.com/html/html_intro.asp
[5] OpenVZ, http://wiki.openvz.org, 2011.

[6] Meixing Le, Angelos Stavrou, Brent ByungHoon Kang, "DoubleGuard: Detecting Intrusions in Multitier Web Applications", IEEE Transactions on dependable and secure computing, vol.9. no.4, July/August, 2002.
[7] B. Parno, J.M. McCune, D. Wendlandt, D.G. Andersen, and A.Perrig, "CLAMP: Practical Prevention of Large-Scale Data Leaks,"Proc. IEEE Symp. Security and Privacy, pp.154-169, 2009.
[8] 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.
[9] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kru gel, and G.Vigna, "Cross Site Scripting Prevention with Dynamic DataTainting and Static Analysis," Proc. Network and Distributed SystemSecurity Symp. (NDSS '07), 2007.
[10] William Stallings, "Cryptography and Network Security Principles and Practices", Fourth Edition, Pearson Education, pp.13-16, 2002.

Author Profile

Muthu Kumara Raja, received bachelor's of technology degree in Information Technology from "Oxford Enginering College" Affliated by Anna University of Tiruchirappalli in 2011 and doing master's degree in Computer Science and Engineering in Hindustan University, Chennai, India