INTRUSION DETECTION IN MULTITIER WEB APPLICATIONS USING DOUBLEGUARD

Neha A. Mohadikar[1] and Prof. M. V. Nimbalkar[2]

[1] Research Scholar, Department of IT, Sinhgad College of Engineering, Pune, India
[2]Associate Professor, Department of IT, Sinhgad College of Engineering, Pune, India

ABSTRACT

Today, the use of distinct internet services and their applications by people are increase in very large amount. Due to its usage, it results the increase in data complexity. So, web services turn their focus on multi-tier design where web server acts as front-end and database server acts as back-end. Attackers try to hack personal data by targeting database server, hence it need to provide more security to both web server and database server. In this paper, the doubleguard system proposes an efficient intrusion detection and prevention system which detects and prevents various attacks in multi-tier web applications. This IDS system keeps track of all user sessions across both web server and database server. For this, it allocates the dedicated web container to each user's session. Each user is associated with unique session ID which enhances more security. The system built well correlated model for website and detects and prevents various type of attacks. The system is implemented by using Apache webserver with MySQL.

KEYWORDS

Attacks, Intrusion detection and prevention system, Multitier web application, Web Server.

1. INTRODUCTION

As we know today's world is the technology world. Hence usage of internet and web services increases tremendously. A web service as well as their different applications provides great usability to user. So, their popularity, availability and usage increased day by day. We used web services and applications in the various fields like banking, shopping, travelling. Since most of the web application is largely open, it is very easy to find security loopholes and turned into insecure web applications. Due to increase in data complexity, web services turns their focus on multi-tiered design approach where the web server runs web application at front end and data is deploy to database server [11] [13]. To provide more security to multitier web services intrusion detection systems (IDS) are largely used and it detects various attacks.

There are two IDS, web IDS and database IDS, both of them can be able to detect unusual network traffic individually sent to either of them. But these IDS can't identify cases in which traffic will be used to attack web server and database server [1] [10]. So, in this paper Doubleguard approach is described. Doubleguard is an IDS system which is used to detect and prevent attacks in multitier web services. This technique models the behavior of user sessions between front end web server and back end database server. It administers both web requests and database request and figure out attacks which independent IDS would unable to detect. This system provides a dedicated container to each user session and each user's session is associated with unique session ID. Also, there are different mappings used to detect and prevent different types of attacks.

DOI: 10.5121/ijci.2015.4604

2. RELATED WORK

As we know Information Technology is an essential part of our daily life. Various web services and its applications work on front end and back end server. Front end consist of application user interface logic and back end server consist of database for particular user data. All sensitive information is stored on database server, so attacker turns their focus from front end to back end. Hence, IDS systems are widely used in order to protect multi-tier web services where at the database side; we are not able to tell which client request deals with which client reply [14]. Also, the web server and database server have distinct communication between them. Hence, we can not figure out the relationship among them.

M Cova, D Balzarotti and Giovanni [3] proposed a different approach for detection of various attacks in web applications using anomaly based detection. This approach consists of different web application state, that is, the information related with single user session. This system operates in two phases, training and detection. Swaddler consists of sensors and analyzer where sensor collets data for web application state [9]. It collects values of state variables and encapsulates them into events which are sent to analyzer. In training phase, profiles for application blocks are established. In detection phase, these profiles are used to identify anomalous application state. But this technique is vulnerable to mimicry attacks.

Giovanni Vigna, Fredrik Valeur, Davide Balzarotti, William Robertson [4] proposed a system for anomaly detection which includes SQL query anomaly detector with Intrusion Detection and prevention system (IPS). This IPS blocks the web requests that are found to be anomalous and SQL anomaly detector detects the normal looking malicious request that generate anomalous database queries [5]. Anomaly detector sends the feedback about detected attacks back to IPS. IPS receives feedback information and updates its configuration to improve its capability of detecting attacks.

Andreas Kind, Marc Ph. Stoecklin, and Xenofontas Dimitropoulos [5] proposed an approach in which different traffic features are simulated with the help of histogram. This system model out histogram patterns, and identifies variations from the created models.

In the existing system, both web and database servers are vulnerable. In most of system, the attackers can bypass the web server to directly attack the database server. Also, attackers may take over the web server after the attack, and then they can obtain full control of the web server and try to launch subsequent attacks and steal sensitive data beyond their privileges [2]. The existing IDS systems are fixed at web server and at database server but they are unable to detect attacks if normal traffic is used to attack database. These IDS are protected from direct remote attacks; but then also the back-end systems are vulnerable to attacks as attacker uses web requests for exploitation of back-end. But, proposed doubleguard system uses a new container-based web server architecture that enables us to separate different information flows by each session. It track the information flows to database server. The main purpose of double guard system is to model the mapping patterns between database queries and http requests to detect malicious user sessions.

3. SYSTEM FRAMEWORK

In this section proposed system architecture is described. This section also describes how the proposed system detects and prevents various attacks such as privilege escalation, session hijack, SQL injection and direct database attack.

The proposed DoubleGuard system is Intrusion Detection and Prevention system (IDS) that models the behavior of user sessions over both front-end web server and back-end database. In this approach, it assigns different container to each user's session and unique session ID is assigned to each user's session. Hence, it separates the information flows between different users. We can run many copies of web server instances so that each one segregated from the rest and having separate information flow. This approach dynamically generates new session. This system chooses to separate communications at the session level and single user compromise with the same web server instances. Sessions represent different users and the communication of a single user goes through the same dedicated web session which allows us to identify fishy behavior of malicious. If detect abnormal behavior in a session, will treat all traffic within this session as tainted and the session will be destroyed immediately. These virtualized sessions can be discarded and quickly reinitialized to serve new sessions. The system includes different mapping policies to detect and prevent attacks. Figure 1 shows the system architecture of system.

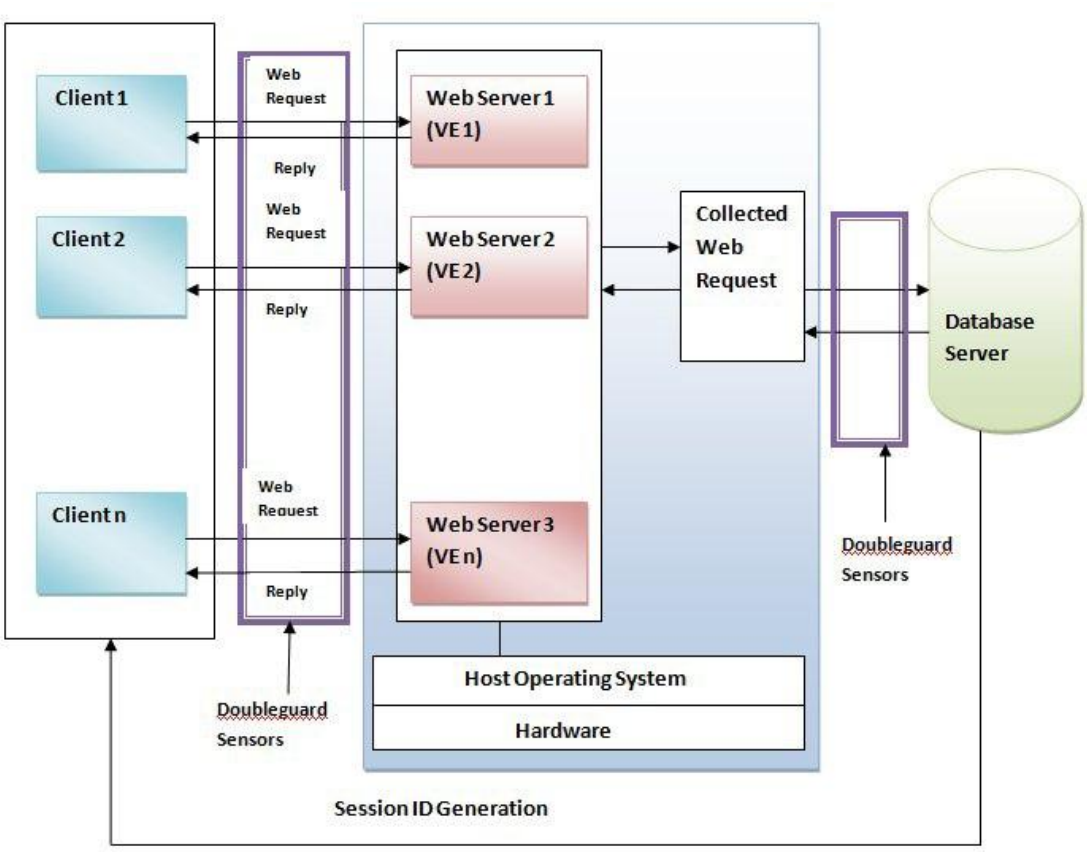

Fig.1 System Architecture

The Doubleguard system is effectual at capturing the following types of attacks.

3.1. SQL Injection Attack

SQL injection attack is a web security attack caused mainly due to improper validation of user input. It holds the confidentiality and integrity of user's sensitive data. This attack takes place between the user interface layer and the business logic layer [6] [12].

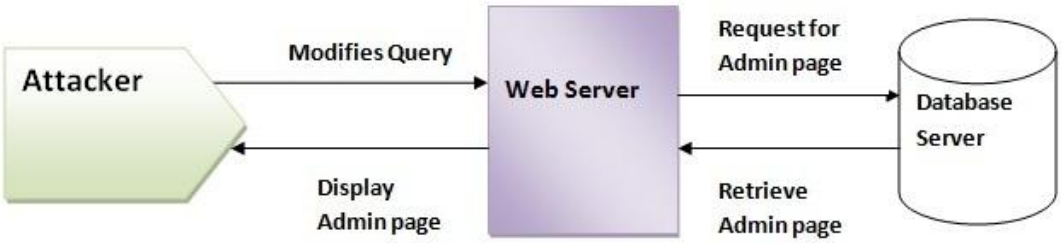

Fig.2 SQL Injection Attack

Figure 2 shows SQL injection attack. Let us see the following example.

SELECT * from table WHERE user=‘ ’ and password= ‘ ’;

This sample considers two input parameters as username and password. If hacker tries to attempt illegal access of database by entering input as SQL statements instead of actual input, then it is called SQL injection attempt.

For example if the hacker inputs,’ OR ‘1’=‘1’ - -, the statement becomes, SELECT * from table WHERE user=‘ ’ OR ‘1’=‘1’- - and password= ‘ ’ ;

Here the user gets unauthorized access to the system because 1=1 is true always and - - indicates the statements following it are comments.

3.2. Direct DB Attack

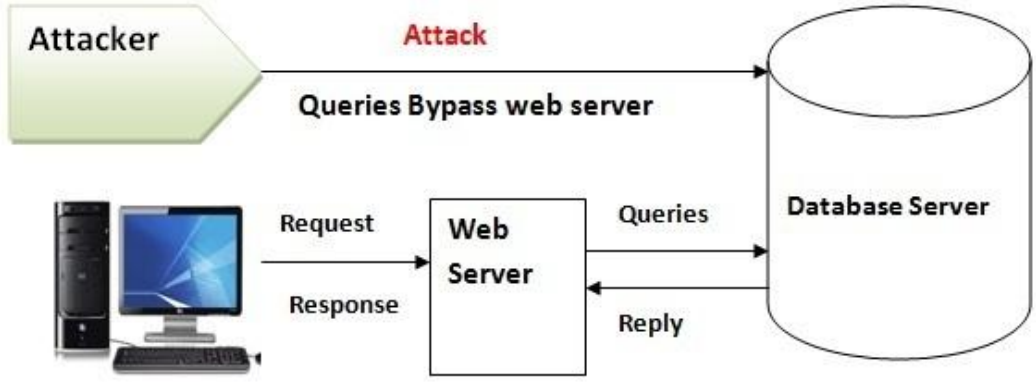

Fig3 Direct DB Attack

Figure 3 shows direct DB attack in which an attacker try to bypass the web server and connect directly to the database. Here, an attacker takes over the web server and submits the queries from the web server to database without sending web requests.

The proposed system uses encryption algorithm to prevent both SQL Injection and Direct DB attack.

Algorithm: Encryption algorithm to prevent SQL Injection and Direct DB attack.

Input: String array[]
Output: Encrypted String encryptedArray[]
Initialize:
numbers="1234567890";
smallAlphabets="abcdefghijklmnopqrstuvwxyz";
capitalAphabets="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
specialSymbols="; ."'+"\""+"[+*&=-]@#$(),_";
Initialize Arbitrary Arrays:
//A1 array
A1=numbers + smallAlphabets + specialSymbols +
Capital Alphabets;
//A2 array
A2=capialAlphabets + numbers + smallAlphabets + specialSymbols;
//A3 array
A3=numbers + capitalAlphabets + smallAlphabets + specialSymbols;
//A4 array
A4=smallAlphabets + specialSymbols + capitalAlphabets + numbers;
//all characters
all=smallAlphabets + capitalAlphabets + numbers + specialSymbols;
Algorithm Steps:-
for (int i=0 to length of input string)
    if(array[i+1]==' ' || array[i+1]==LowerCaseLetter)
        then encryptedArray[i]=A1
    else if (array[i+1]==UpperCaseLetter)
        then encryptedArray[i]=A2
    else if (array[i+1]==Number)
        then encryptedArray[i]=A3
    else if (array[i+1]==SpecialLetter)
        then encryptedArray[i]=A4
    else
        encryptedArray[i]=array[i];
return encryptedArray;

3.3. Session Hijack Attack

This attack is the exploitation of user's session i.e. obtaining session key to gain unauthorized access to information or various web services. It is usually done at the web server side. An attacker takes over the web server and try to hijacks all valid user sessions to steal sensitive information [7]. By hijacking other user's sessions, the attacker can perform spoofing, man-in-the-middle attack, replay attack, packet drop attack etc. Figure 4 describes Session Hijack attack.

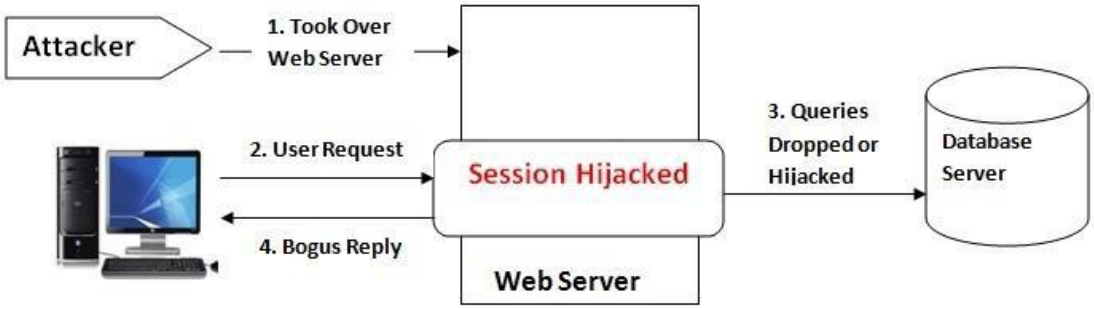

Fig.4 Session Hijack Attack

This attack can be prevented using strong and encrypted algorithm. There are two parts, session ID generation algorithm and session ID sharing algorithm. In first part, web server generates unique session ID using session ID generation algorithm and in second part, this session ID is encrypted at server side and decrypted at client side using session ID sharing algorithm. At the end both client and server have same key for communication.

Algorithm 1: Session ID Generation algorithm

Initialize variable:

```
res;
Buff;
randomNo=0;
res_byte_length=32;
sessionIdLength=32;
 for (int i = 0 to  sessionIdLength)
 {
        byte b1=(byte) ((randomNoUsingMessageDigest() & 0xf0) >> 4);
        byte b2 = (byte) (randomNoUsingMessageDigest() & 0x0f);
        if (b1 < 10)
                Buff.append((char) ('0' + b1)
         else
                 Buff.append((char) ('A'+ (b1 -10)));
         if (b2 < 10)
```

```
            Buff.append((char) ('0' + b2));
        else
            Buff.append((char) ('A'+ (b2 -10)));
        res_byte_length++;
}
```

Algorithm 2: Session ID sharing algorithm

1. The client originates the session with the server using his/her credentials (Login/password).
2. The client requests a server for RSA Public key.
3. The client encrypts the credentials with RSA Public key.
4. The server decrypts the credentials and stores it in the session.
5. The server encrypts the generated Session ID with AES and sends it to the client.
6. The client decrypts the Session ID using AES with the credentials.
7. Both the client and the server have now the same "session ID" which is used for communication.

Figure 5 shows the steps for session ID sharing algorithm.

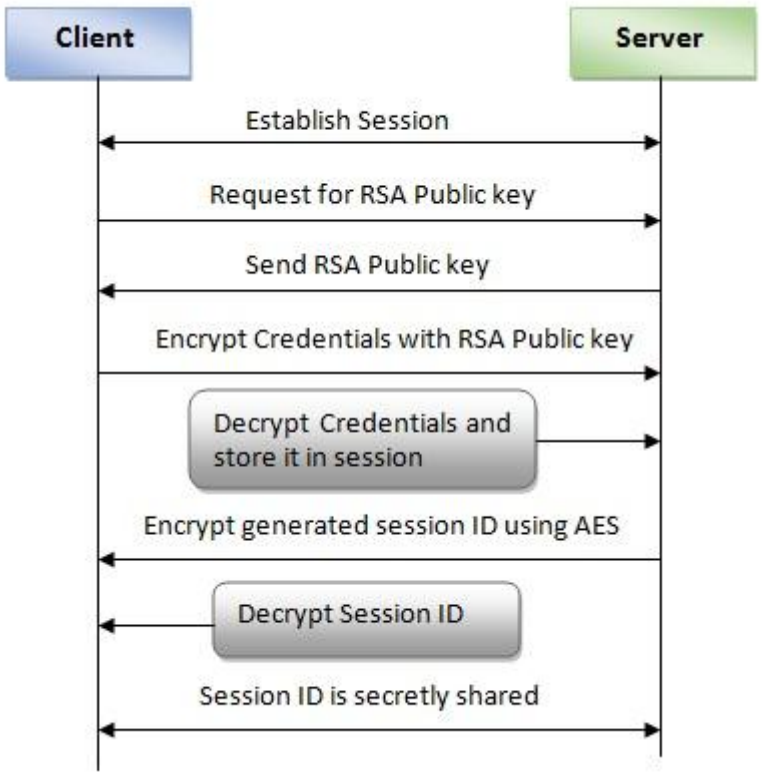

Fig.5 Session ID sharing algorithm

3.4. Privilege Escalation Attack

Web applications plays vital role in E-commerce. These web applications can be widely deployed and open nature, so attacker tries to get unauthorized access to user's data. Hence, to maintain security of web application is very important concern.

In privilege escalation attack, attacker performs login as a normal user, upgrades his role as admin or other user and triggers different queries to get admin access or other user account access [8]. Figure 6 shows privilege escalation attack.

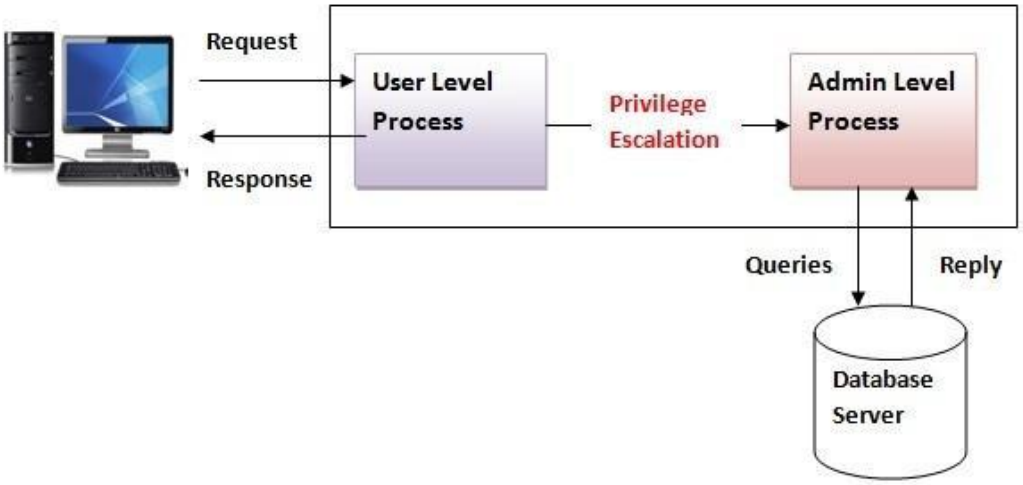

Fig.6 Privilege Escalation Attack

This type of attack cannot be detected by web server IDS and database IDS because both request and queries are authentic. But in the proposed approach, web applications uses certain access control policies. These policies are used to verify whether given legitimate user has required privileges to access given resources such as database table. Authorization is always expected before accessing every resource. This becomes the basis for web application security.This approach uses access control policies. These policies are based on adding articles, Delete articles or update articles on website.

This approach is able to detect two types of privilege escalation attack.

- Vertical privilege escalation attack, when an attacker or malicious user upgrades his/her privilege level (access more than they are entitled according to their role) and try to get admin access.
- Horizontal privilege escalation attack, when an attacker or malicious user tries to access the system resources of other users..

4. PERFORMANCE EVALUATION

In this section, experimental results are described. Experimental results were taken by 25 number of request i.e. total 25 user sessions. The performance of the system is evaluated using parameters namely, attack detection rate, and number of request per second VS number of replies per second.

a) Attack Detection Rate

It is the ratio between the total numbers of attack connections detected by proposed system to the total number of attacks currently available in the data set.

This graph in figure 7 shows Attack Detection Rate. If the number of user sessions increases, the rate of detected intrusion instances (%) varies. Here, for 0 number of session detected intrusion instances is 0% and varied by increasing number of session. Finally, for 20 sessions detected intrusion instances increases to 80%.

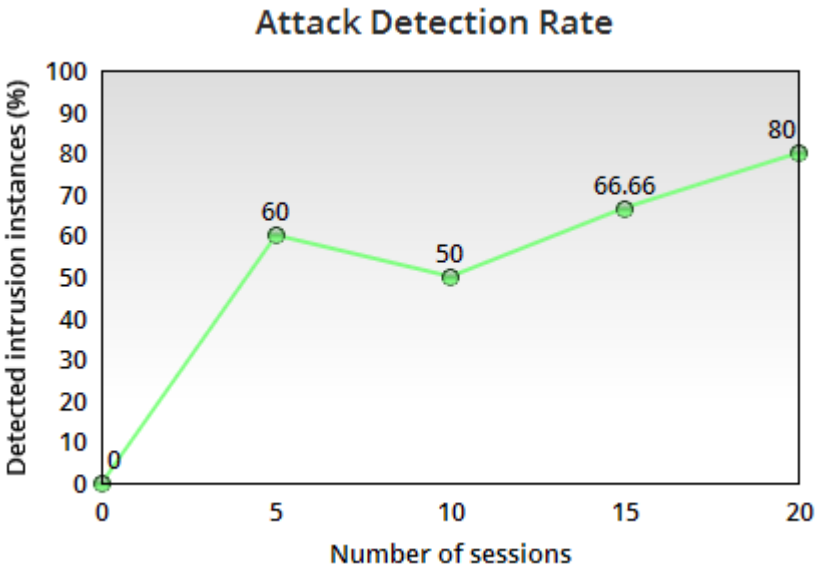

Fig.7 Attack Detection Rate

b) Graph of Proposed system Vs Existing system

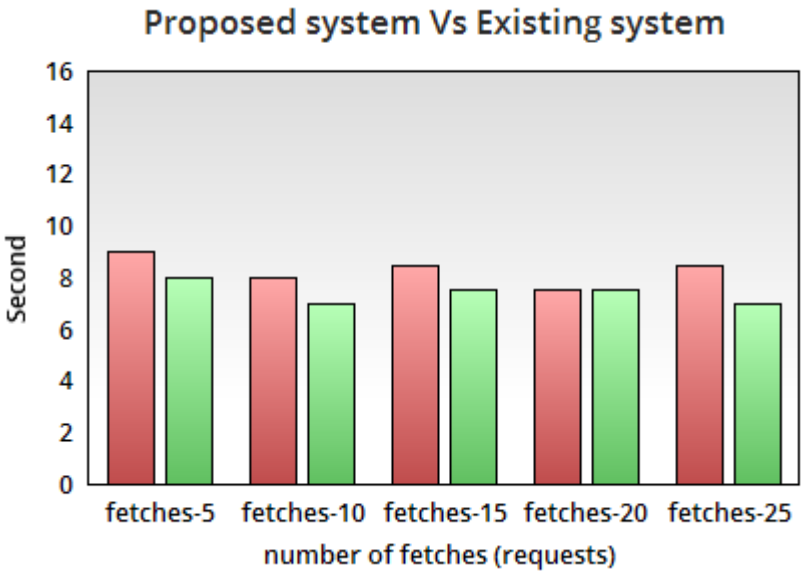

Fig.8 Graph of Proposed system Vs Existing system

The above figure 8 shows graph of proposed system Vs existing system. This evaluates the time at which number of requests are served by the system. To fetch 5 web requests, the old system requires 9 seconds but proposed system requires 8 seconds. Hence, to fetch number of request, the proposed system works more efficiently than old existing system.

5. CONCLUSION

DoubleGuard system is used to detect and prevent the intrusions in multi-tier web application. It is container-based IDS which builds particular model for multitier web applications and assigns a unique session ID to each user. The system separates the information flow from each web server session. Hence, there will be precise detection and prevention of attacks such as SQL Injection Attack, Direct DB Attack, Session Hijack Attack and Privilege Escalation Attack. Also the requests which does not match to given model that will be treat as an intruder.

It is an application independent system and used for both front-end as well as back-end. It is used for web server which provides better security for data and web application.

REFERENCES

[1] Meixing Le, AngelosStavrou, Brent ByungHoon Kang,"DoubleGuard: Detecting Intrusions in Multitier Web Applications", IEEE Transactions On Dependable And Secure Computing, Vol. 9, No. 4, March 2014.

[2] Felmetsger Viktoria, LudovicoCavedon, Christopher Kruegel, and Giovanni Vigna. "Toward automated detection of logic vulnerabilities in web applications." In USENIX Security Symposium ACM, pp. 143-160. 2010.

[3] Cova, Marco, DavideBalzarotti, Viktoria Felmetsger, and Giovanni Vigna. "Swaddler: An approach for the anomaly-based detection of state violations in web applications." In Recent Advances in Intrusion Detection, pp. 63-86. Springer Berlin Heidelberg, 2007.

[4] Vigna, Giovanni, Fredrik Valeur, Davide Balzarotti, William Robertson, et.al. "Reducing errors in the anomaly-based detection of web-based attacks through the combined analysis of web requests and SQL queries" Journal of Computer Security 17, no. 3 (2009): 305-329.

[5] Kind, Andreas, Marc PhStoecklin, and Xenofontas Dimitropoulos, "Histogram-based traffic anomaly detection." Network and Service Management, IEEE Transactions on 6, no. 2 (2009): 110-121.

[6] Avireddy, S., Perumal, et.al,"Random4: an application specific randomized encryption algorithm to prevent SQL injection" In Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on (pp. 1327-1333).

[7] Manivannan, S. S., and E. Sathiya moorthy. "A Prevention Model for Session Hijack Attacks in Wireless Networks Using Strong and Encrypted Session ID." Cybernetics and Information Technologies 14.3 (2014): 46-60.

[8] Monshizadeh, Maliheh, Prasad Naldurg, and V. N. Venkatakrishnan. "Mace: Detecting privilege escalation vulnerabilities in web applications." In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2014, pp. 690-701.

[9] SachinJ.Pukale, M. K.Chavan,"A Review of Anomaly Based Intrusions Detection In Multi-Tier Web Applications", International Journal Of Computer Engineering & Technology, Volume 3, Issue 3, October - December (2012), pp. 233-244.

[10] Sujitha, M., P. Suganya, T. Shampavi, and S. Anjanaa. "Dual Safeguard: Intrusion Detection and Prevention System in Web Applications." International Journal of Computer Applications 67, no. 9 (2013): 13-18.

[11] Kruegel, Christopher, and Giovanni Vigna. "Anomaly detection of web-based attacks." In Proceedings of the 10th ACM conference on Computer and communications security, pp. 251-261. ACM, 2003.

[12] Valeur, Fredrik, Darren Mutz, and Giovanni Vigna. "A learning-based approach to the detection of SQL attacks" Detection of Intrusions and Malware, and Vulnerability Assessment. Springer Berlin Heidelberg, 2005. 123-140.
[13] Egele, Manuel, TheodoorScholte, EnginKirda, and Christopher Kruegel. "A survey on automated dynamic malware-analysis techniques and tools." ACM Computing Surveys (CSUR) 44, no. 2 (2012): 6.
[14] Sekar, R. "An Efficient Black-box Technique for Defeating Web Application Attacks." NDSS. 2009.

AUTHORS

Neha A. Mohadikar appeared for the M.E. degree in Information Technology from Savitribai Phule University, Pune, India.