Enhanced Security Approach for Multitier Web Applications

Madhuri Bagal[1], Priyanka Lawand[2], Harshad Ghegadmal[3], Mayur Deshmukh[4]

Dept. of Computer Science & Engineering, Sinhagad Institute of Technology, Lonavala, Tal. Maval, Dist. Pune, India.

[1]madhuribagal@gmail.com
[2]priyankaslawand@gmail.com
[3] hghegadmal@gmail.com
[4]enggmayur.deshmukh@gmail.com

***Abstract*— In today's era, huge amount of data is stored at network system but modern network systems have much trouble in security vulnerabilities such as buffer overflow, security flaws of applications, and different attacks. Various IDS systems are used to protect the data from these attacks at network system.**

In this paper, we present an IDS system that controls the behaviour of network system at both front end (web server) and back end database. In this system, we keep eye on both web request and database request, and find out different attacks on network system. To implement the system this paper has used container based architecture. This architecture maps all request coming to the web server to fire database query by increasing system security.

***Keywords*—IDS, Double Guard, Multitier Architecture.**

I. INTRODUCTION

Web-Based services and applications have enlarged in both popularity and complexity over the past few years. Web applications are used in our daily tasks, such as e-commerce, social networking, education etc. Such applications uses multitier web architecture which in holds of web server as front end logic and back end comprises database or file server. Unfortunately it has been found easy to disturb the functionality of Internet by attacking its infrastructure taking advantage of Internet services and protocols. The vulnerability of web application has attracted the attention of malicious hackers to exploit and access to sensitive information which might lead to enormous gain. The attacks have recently become more different, as attention has shifted from attacking the front end to abusing vulnerabilities of the web applications in order to corrupt access the back-end database system (e.g., SQL injection attacks). The security of web-based applications should be addressed by means of careful design and thorough security testing. A superfluity of Intrusion Detection Systems (IDSs) currently examines network packets individually within both the web-server and the database system. However, in multitier architectures, the back-end database server is often protected behind a firewall while the web-servers are remotely accessible over the Internet. Intrusion detection systems have been widely used to protect multitier web services, such as to detect known attacks by matching misused traffic patterns or signatures Individually, the web IDS and the database IDS can detect abnormal network traffic sent to either of them. But, we found that these IDSs cannot detect cases wherein normal traffic is used to attack the web server and the database server. For example, if an attacker with non admin privileges can log in to a web server using normal-user access credentials, he/she can find a way to issue a privileged database query by exploiting vulnerabilities in the web server. Neither the web IDS nor the database IDS would detect this type of attack since the web IDS would merely see typical user login traffic and the database. IDS would see only the normal traffic of a privileged user. This type of attack can be easily detected if the database IDS can identify that a privileged request from the web server is not associated with user-privileged access. Regrettably, within the current multithreaded web server architecture, it is not feasible to detect or profile such causal mapping between web server traffic and DB server traffic since traffic cannot be clearly attributed to user sessions.

II. THREE TIER ARCHITECTURE

Web Sphere Application Server provides the application logic layer in three tier architecture, enabling client components to interact with data resources and legacy applications. Collectively, three tier architectures are programming models that enable the distribution of application functionality across three independent systems.

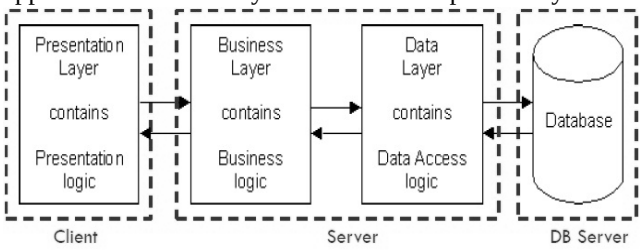


Fig. Three Tier Architecture

A. First Tier:

Responsibility for presentation and user interaction resides with the first tier components. These client components enable the user to interact with the second tier processes in a secure and intuitive manner. Web Sphere Application Server supports several client types. Clients do not access the third tier services directly. For example, a client component provides a form on which a customer orders products. The client component submits this order to the second-tier processes, which check the product databases and perform tasks that are needed for billing and shipping.

B. Second Tier:

The second tier processes are commonly referred to as the application logic layer. These processes manage the business logic of the application, and are permitted access to the third tier services. The application logic layer is where most of the processing work occurs. Multiple client components can access the second tier processes simultaneously so this application logic layer must manage its own transactions.

C. Third Tier:

The third tier services are protected from direct access by the client components residing within a secure network. Interaction must occur through the second tier processes.

III. RELATED WORK

A network Intrusion Detection System is mainly categorized into the two types: anomaly detection and misuse detection. In Anomaly detection , the correct and acceptable static form and dynamic behavior of the system is defined and characterized first. This can be used to detect the changes or anomalous behaviors . Then an anomaly detector compares actual usage patterns against models that are already established in order to identify abnormal events. We follow the anomaly detection approach since we depend on a training phase to build the correct model.

DoubleGuard uses the container ID for each session to causally map the related events, whether they be concurrent or not. Databases should be given highest level of security as they contain vital information. The system presented is composed of both web IDS and database IDS in order to achieve more accurate detection. It makes use of a reverse HTTP proxy for maintaining a reduced level of service when false positives are present.

Nevertheless there are certain types of attacks that utilize normal traffics and can be detected by neither the web IDS nor the database IDS. We used Virtualization in order to isolate objects and improve security performance. A lightweight virtualization, such as OpenVZ, Parallels Virtuozzo, or Linux-VServer are some of the alternatives. Thousands of containers can run on a single physical host. In our DoubleGuard, we utilized the container ID so that we would be able to separate session traffic for identifying causal relationships between webserver requests and database query events.

We can initialize thousands of containers on a single physical machine. These virtualized containers can be discarded, reverted, or quickly reinitialized for serving new sessions. As a single physical webserver runs many containers, each one is an exact replica of the original webserver. We follow an approach that not only dynamically generates new containers but also recycles used ones. Therefore a single physical server can run continuously while serving all web requests. Each session is logically separated from other session. Using a read-only clean template, we initialized each virtualized container. We assure that each session will be served with a clean webserver instance at initialization.

IV. PROPOSED SYSTEM

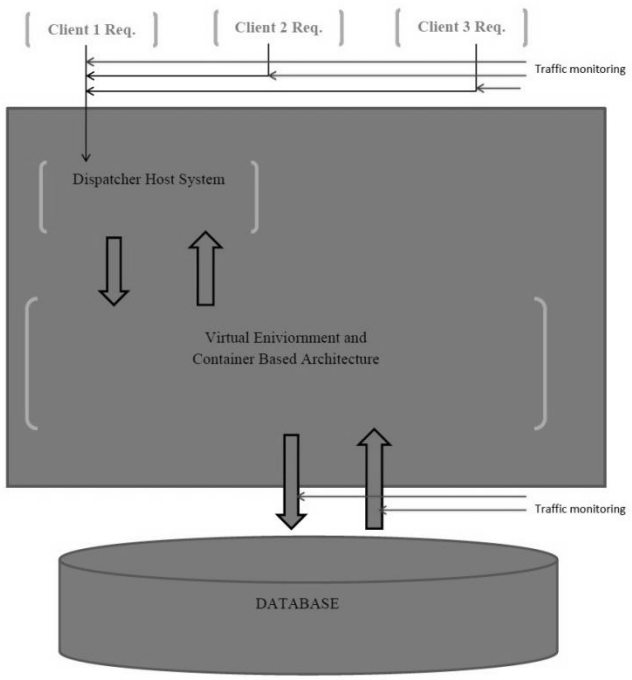


Fig. System Architecture

Initially, this system set up our threat model which will include our assumptions and the types of attacks which we are aiming to protect against. We assume that both the web and the database servers are defenseless. Attacks are network borne and come from the web clients; they can launch application layer attacks to compromise the web servers they are connecting to. The attackers can bypass the web server to directly attack the database server. We consider that the attacks can neither be detected nor be prevented by the current web server IDS, those attackers

may take over the web server after the attack, and that afterward they can obtain full control of the web server to launch subsequent attacks. For example, the attackers could modify the application logic of the web applications, eavesdrop or hijack other user's web requests, or intercept and modify the database queries to steal sensitive data beyond their privileges.

On the other hand, at the back end, we assume that the database server will not be completely taken over by the attackers. Attackers may strike the database server through the web server or, more directly, by submitting SQL queries, they may obtain and pollute sensitive data within the database. These assumptions are reasonable since, in most cases, the database server is not exposed to the public and is therefore difficult for attackers to completely take over. We assume no prior knowledge of the source code or the application logic of web services deployed on the web server. In addition, we are analyzing only network traffic that reaches the web server and database. Our assumption is that no attack would occur during the training phase and model building.

V. COMMON WEB APPLICATION ATTACKS

1. *Privilege Escalation Attack*

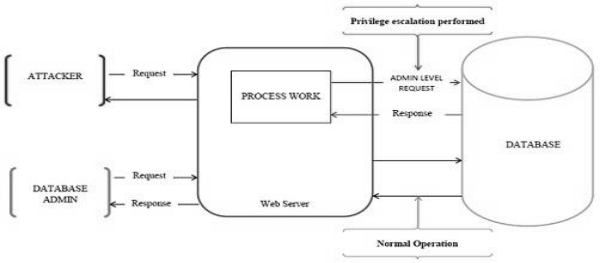


Fig. Privilege escalation attack

For every web application admin level query and regular user query (SQL query) are stored separately in the database. When regular user sends web request, SQL query gets triggered and when admin sends any web request, set of admin level query gets triggered. Suppose any attacker enters as a regular user using regular user's credentials and tries to trigger admin level queries to obtain administrator data. Here attacker tries to upgrade his privilege from regular user to admin. This attack is easily detected using query matching model because regular user's web request does not match with admin level query. In this way using query matching model detect attack and prevent it.

2. *Hijack Future Session Attack*

In computer science, session hijacking is also known as cookie hijacking. This kind of attack is mainly aimed at the web server side. An attacker usually takes over the web server and therefore hijacks all subsequent legitimate user

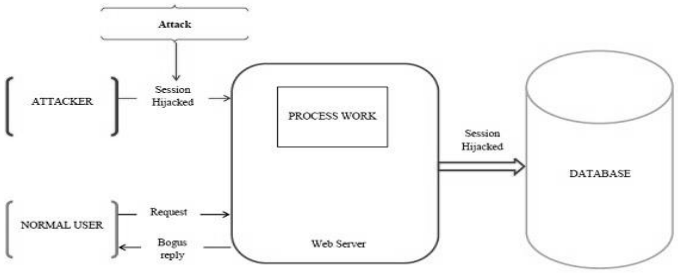
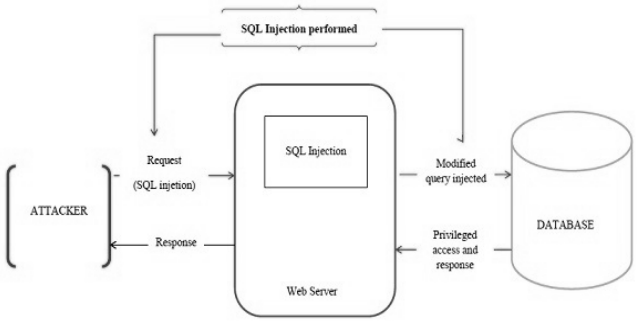


Fig. Hijack Future Session Attack

session to launch attack. For instance, by hijacking other user sessions, the attacker can eavesdrop, send spoofed replies, and/or drop user requests. A session hijacking attack can be further categorized as a spoofing/man-in-the-Middle attack, an Exfiltration attack, a denial-of-Service attack or replay attack. According to the mapping model, the web request should invoke some database queries and then abnormal situation can be detected. However, neither a conventional web server IDS nor a database IDS can detect such an attack by itself. Fortunately, the isolation property of our container based web server architecture can also prevent this type of attack. As each user's web requests are isolated into a separate container, an attacker can never break into other user's sessions.

3. *Injection Attack*

Attacks such as SQL injection do not require compromising the web server. Attackers can use existing vulnerabilities in the web server logic to inject the data or



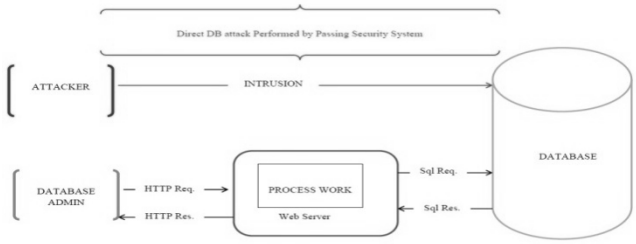
string content that contains the exploits and then use the web server to relay these exploits to attack the back-end

Fig. SQL Query Injection Attack

database. Since our approach provides two-tier detection , even if the exploits are accepted by the web server, the relayed contents to the DB server would not be able to take on the expected structure for the given web server request. For instance, since the SQL injection attack changes the structure of the SQL queries, even if the injected data were to go through the web server side, it would generate SQL queries in a different structure that could be detected as a deviation from the SQL query structure that would normally follow such a web request.

4. Direct DB Attack

This kind of attack mainly happens by passing firewall. It is possible for an attacker to bypass the web server or



firewall and connect directly to the database.

Fig. Direct Database Attack

In direct DB attack request will not come from front end (like JSP or HTML pages). It will come from direct java program which sending request to server. An attacker could also have already taken over the web server and be submitting such queries from the web server without sending web request. Furthermore, if these DB queries were within the set of allowed queries, then the IDS it would not detect its web request. This type of attack can be caught with our approach since we cannot match any web request these queries.

5. Man-In-The-Middle Attack

In this type of attack, even though the attacker tries to make independent connections with those who are vulnerable, it will not be able to authenticate itself to the prove and verifier which has no clue of the fingerprint of the two end nodes. And the intruders not get a chance to know the finger print of the node.

VI. CONCLUSION

We propose an intrusion detection system which builds models of normal behavior for multitier web applications from both front-end web (HTTP) requests and back-end database (SQL) queries. Previous IDSs correlated or summarized alerts, whereas Double Guard forms a container-based IDS with multiple input streams to produce alerts. Such correlation of input streams provides a better characterization of the system for anomaly detection since the intrusion sensor has a more specific normality model that investigates a wider range of attacks

VII. FUTURE SCOPE

It is possible to make some future modifications into the system; which can be make existing system more efficient. The Intrusion detection systems can be installing on wide range of machines having different operating system and platforms. The query processing mechanism can be made simpler by applying natural language processing (NLP); so as to convert simple English sentences into SQL queries.

Since the this system works on the basis of signature; each activity of intrusions is to be memorized by the system previously. New attacks are often unrecognizable by popular IDS. So there is continuous race going in between new attacks and detection systems have been a challenge. Nowadays Intrusion detection systems also work on the wireless networks. The latest wireless devices come with its own set of protocols for communication that break the traditional OSI layer model. So IDS must learn new communication patterns of the latest wireless technology.

REFERENCES

[1] 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.

[2] Meixing Le, Angelos Stavrou, Member, IEEE, and Brent ByungHoon Kang, Member, IEEE," DoubleGuard: Detecting Intrusions in Multitier Web Applications", IEEE Transactions on Dependable and Secure Computing,Vol.9,No.4,July/August 2012.

[3] M. Christodorescu and S. Jha, "Static Analysis of Executables toDetect Malicious Patterns," Proc. Conf. USENIX Security Symp.,2003.

[4] M.Patil, R. More,"Using Container Architecture to Detect Intrusion for Multitier Web Application", *International Journal of Computer Applications (0975 – 8887) Volume 62– No.9, January 2013.*

[5] N.Sakthipriya, K.Palanivel, "Intrusion Detection for Web Application: An Analysis", International Journal of Scientific & Engineering Research, Volume 4, Issue 5, May-2013 ISSN 2229-5518

[6] S.Gore, A. Rangari, V.Bhagat, K. Khalilullah,"Dual Armor: Intrusion Detection and Prevention System in Multitier Web Applications", International Journal of Emerging

Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 1, January 2013).

[7] S.Potter, J.Nieh,"Apiary: Easy-to-Use Desktop *Application Fault Containment on Commodity Operating* Systems", Columbia University Technical Report CUCS-034-09, August 2009.

[8] Tripti Sharma, Khomlal Sinha,"Intrusion Detection Systems Technology", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-1, Issue-2, December 2011.

[9] V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna, "TowardAutomated Detection of Logic Vulnerabilities in Web Applications,"Proc. USENIX Security Symp., 2010.

[10] W. Robertson, F. Maggi, C. Kruegel, and G. Vigna, "Effective Anomaly Detection with Scarce Training Data," Proc. Network and Distributed System Security Symp. (NDSS), 2010.