ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 2 Issue Special 1 Jan-March 2014
International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014)

Survey on Detecting SQL Injection and Cross Site Scripting in Multitier Web Applications

[I]R.Stalinbabu, [II]P. Chellammal
[I]M.E. (Dept. of CSE), J.J. College of Engg. and Tech., Tiruchirappalli, Tamil Nadu, India
[II]Assistant Professor, J.J. College of Engg. and Tech., Tiruchirappalli, Tamil Nadu, India

Abstract
This is the method to detect intrusion in multitier web applications. Multitier web application include two ends that is front end web server as well as back end of the applications database server. The front end web server which can responsible to run the application and gives that output to back end i.e. file server. This strategy is useful to identify the intrusion at both front end and back end of web application. It is used to monitor the behavior across front end web server and back end database server or file server using IDS. We will also able to detect intrusion in static and dynamic web application. IDS having maximum accuracy and is mainly responsible to identify intrusion.

Keywords
Intrusion Detection System, Multitier, Intrusion, Web Server, File Server

I. Introduction

Web-Delivered services and applications have increased in both popularity and complexity over the past few years. Daily tasks, such as banking, travel, and social networking, are all done via the web. Such services typically employ a web server front end that runs the application user interface logic, as well as a back-end server that consists of a database or file server. Due to their ubiquitous use for personal and/or corporate data, web services have always been the target of attacks. These attacks have recently become more diverse, as attention has shifted from attacking the front end to exploiting vulnerabilities of the web applications in order to corrupt the back-end database system (e.g., SQL injection attacks). A plethora of Intrusion Detection Systems (IDSs) currently examine network packets individually within both the web server and the database system. However, there is very little work being performed on multitiered Anomaly Detection (AD) systems that generate models of network behavior for both web and database network interactions. In such multitiered architectures, the back-end database server is often protected behind a firewall while the web servers are remotely accessible over the Internet. Unfortunately, though they are protected from direct remote attacks, the back-end systems are susceptible to attacks that use web requests as a means to exploit the back end. To protect multitiered web services, Intrusion detection systems have been widely used to detect known attacks by matching misused traffic patterns or signatures. A class of IDS that leverages machine learning can also detect unknown attacks by identifying abnormal network traffic that deviates from the so-called "normal" behavior previously profiled during the IDS training phase. Individually, the web IDS and the database IDS can detect abnormal network traffic sent to either of them. However, 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 readily detected if the database IDS can identify that a privileged request from the web server is not associated with user-privileged access. Unfortunately, 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. This present DoubleGuard, a system used to detect attacks in multitiered web services. The approach can create normality models of isolated user sessions that include both the web front-end (HTTP) and back-end (File or SQL) network transactions. To achieve this, employ a lightweight virtualization technique to assign each user's web session to a dedicated container, an isolated virtual computing environment. Use the container ID to accurately associate the web request with the subsequent DB queries. Thus, DoubleGuard can build a causal mapping profile by taking both the web server and DB traffic into account.

A. Introduction to Multi-Tier Web Application

The three tier model at the database server its very hard to find which transaction request is send by which user its because of no communication separation between web server and database server. That is shown in following Fig. 1.1. It illustrates the classic three-tier model. At the database side, we are unable to tell which transaction corresponds to which client request.

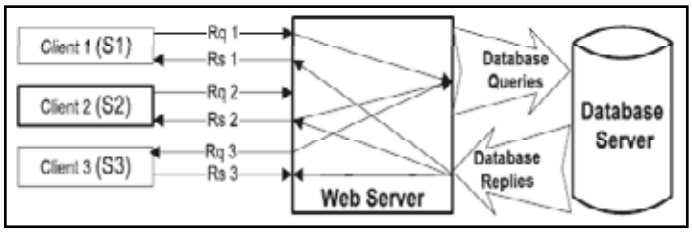

Fig. 1.1: Multitier Architecture

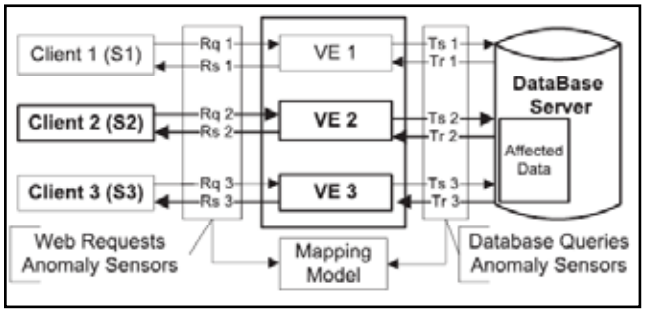

Fig. 1.2: Web Server instances running in containers

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

The communication between the web server and the database server is not separated, and we can hardly understand the relationships among them. Fig. 1.2 depicts how communications are categorized as sessions and how database transactions can be related to a corresponding session. According to Figure 1.1, if Client 2 is malicious and takes over the web server, all subsequent database transactions become suspect, as well as the response to the client. By contrast, according to Figure 1.2, Client 2 will only compromise the VE 2, and the corresponding database transaction set T2 will be the only affected section of data within the database.

Most of the IDS approaches are directed towards signature based detection mechanisms. Well-defined patterns of any well-known attack are used for comparison. To this various pattern matching algorithms can be applied to detect attacks to the system. As per Open Web Application Security Project OWASP [6] four attacks amongst top ten attacks happens due to faulty input validations. Even a simple application logic error gives opportunity to hackers to get into the system. In this paper we are defining some basic terminologies in first section followed by the various works that has been done to this issue and so on up to the conclusion.

B. Types of Attacks

1. Injection Attack

SQL injection vulnerabilities allow attackers to insert SQL commands as a part of user input. When an SQL query is constructed dynamically with maliciously-devised user input containing SQL keywords, attackers can gain access or modify critical information such as a credit card number in a database without proper authorization.

SQL Injection is one of the many web attack mechanisms used by hackers to steal data from organizations. It is perhaps one of the most common application layer attack techniques used today. It is the type of attack that takes advantage of improper coding of your web applications that allows hacker to inject SQL commands into say a login form to allow them to gain access to the data held within your database. In essence, SQL Injection arises because the fields available for user input allow SQL statements to pass through and query the database directly.

2. SQL Injection: An In-depth Explanation

Web applications allow legitimate website visitors to submit and retrieve data to/from a database over the Internet using their preferred web browser. Databases are central to modern websites – they store data needed for websites to deliver specific content to visitors and render information to customers, suppliers, employees and a host of stakeholders. User credentials, financial and payment information, company statistics may all be resident within a database and accessed by legitimate users through off-the-shelf and custom web applications. Web applications and databases allow you to regularly run your business.

SQL Injection is the hacking technique which attempts to pass SQL commands (statements) through a web application, for execution by the backend database. If not sanitized properly, web applications may result in SQL Injection attacks that allow hackers to view information from the database and/or even wipe it out.

Such features as login pages, support and product request forms, feedback forms, search pages, shopping carts and the general delivery of dynamic content, shape modern websites and provide businesses with the means necessary to communicate with prospects and customers. These website features are all examples of web applications which may be either purchased off-the-shelf or developed as bespoke programs.

These website features are all susceptible to SQL Injection attacks which arise because the fields available for user input allow SQL statements to pass through and query the database directly.

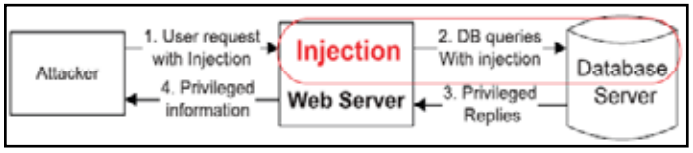

Fig. 1.3: SQL Injection Attack

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 database. Since the approach provides a 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. Figure 1.2 illustrates the scenario of a SQL injection attack.

3. Cross Scripting Attack

Cross-site scripting (XSS) vulnerabilities allow attackers to insert malicious scripts as a part of user input and the script is executed at other user's browser due to the lack of input validation. We provide a simple example of attacks exploiting XSS vulnerabilities. Consider a search engine that returns the search results including the same query given by a user. If the user input includes a script and the returned result page does not encode the script into HTML code, the script in the returned result page will be executed.

Hackers are constantly experimenting with a wide repertoire of hacking techniques to compromise websites and web applications and make off with a treasure trove of sensitive data including credit card numbers, social security numbers and even medical records. Cross Site Scripting (also known as XSS or CSS) is generally believed to be one of the most common application layer hacking techniques.

SQL injection and XSS are the most popular attack. To add to this, many other attack methods, such as Information Disclosures, Content Spoofing and Stolen Credentials could all be side-effects of an XSS attack.

Cross-Site Scripting (also known as XSS) is one of the most common application-layer web attacks. XSS vulnerabilities target scripts embedded in a page which are executed on the client-side (in the user's web browser) rather than on the server-side. XSS in itself is a threat which is brought about by the internet security weaknesses of client-side scripting languages such as HTML and JavaScript. The concept of XSS is to manipulate client-side scripts of a web application to execute in the manner desired by the malicious user. Such a manipulation can embed a script in a page which can be executed every time the page is loaded, or whenever an associated event is performed.XSS is the most common security vulnerability in software today. This should not be the case as XSS is easy to find and easy to fix. XSS vulnerabilities can have consequences such as tampering and sensitive data theft.

II. Intrusion Detection System

An Intrusion Detection System (IDS) is a device or software application that monitors network or system activities for malicious activities or policy violations and produces reports to a Management Station. Some systems may attempt to stop an intrusion attempt but this is neither required nor expected of a monitoring system. Intrusion detection and prevention systems (IDPS) are primarily focused on identifying possible incidents, logging information about them, and reporting attempts. In addition, organizations use IDPSes for other purposes, such as identifying problems with security policies, documenting existing threats and deterring individuals from violating security policies. IDPSes have become a necessary addition to the security infrastructure of nearly every organization. For the purpose of dealing with IT, there are three main types of IDS:

- Network intrusion detection system (NIDS)
- Host-based intrusion detection system (HIDS)
- Stack-based intrusion detection system (SIDS)

IDS are generally categorized in two types viz. Anomaly Detection and Misuse Detection.

A. Anomaly Detection:

In this type of IDS a well-defined behavior of user
Application is known on the first hand. Any random actions are considered as a threat. A predefined pattern is associated with some type of attack. Any unconventionality from the normal conduct is detected as an anomaly to the system. Previously unknown attacks can be detected but there is a high chance of false positive. Anomaly detection is somewhat difficult for dynamic applications.

B. Misuse Detection:

Principally misuse detection defines the attack description and matches them against audit data stream. It searches for any of the known attacks. Verifying against audit data excellently detects the attacks with very less false positive but this scheme is ineffective for undefined attacks. Once intrusion has been detected, there are many ways in which the administrator can be alerted. Depending upon the urgency of the situation an email or a mobile message to the administrator can be sent, or an alarm can be raised to alert the guards. A counterattack against the intruder is also possible in which the attacker can be traced out and attack on one's
system can be mounted but there is a possibility of collateral damage.

III. Container Architecture

Implementation of Intrusion detection System in multitier web application using container architecture as following: Container architecture basically detects intrusion in two sides that is web server side as well as database side. This architecture of Intrusion Detection System is comes under two type of Intrusion detection system so we can also able to say, Implementation of Container Architecture Intrusion detection system is combination of behavioral IDS and Signature based IDS. That means it is Hybrid category of intrusion detection system. This is best approach for Intrusion Detection in multitier web application. Here proposed an efficient system using container architecture that can detect the attacks in multi-tiered web services. This approach can create normality models of isolated user sessions that include both the web front end (HTTP) and back-end (File or SQL) network transactions. To achieve this, we employ a lightweight virtualization technique to assign each user's web session to a dedicated container in an isolated virtual computing environment. Here use the container ID to accurately associate the web request with the subsequent DB queries. Typical flow data particularly relevant to intrusion detection and prevention includes the following:

- Source and destination IP addresses
- Source and destination TCP or UDP ports or ICMP types and codes
- Number of packets and number of bytes transmitted in the session
- Timestamps for the start and end of the session.

In this prototype, chose to assign each user session into a different container; however, this was a design decision. For instance, we can assign a new container per each new IP address of the client. In this implementation, containers were recycled based on events or when sessions time out. This approach is able to use the same session tracking mechanisms as implemented by the Apache server (cookies, mod, user track, etc.) because lightweight virtualization containers do not impose high memory and storage overhead. Thus, we could maintain a large number of parallel-running Apache instances similar to the Apache threads that the server would maintain in the scenario without containers. If a session timed out, the Apache instance was terminated along with its container. Consider, we used a 60-minute timeout due to resource constraints of our test server. However, this was not a limitation and could be removed for a production environment where long-running processes are required. Fig. 1.2 depicts the architecture and session assignment of prototype, where the host web server works as a dispatcher, shows that how communications are categories as sessions and how database transactions can be related to a corresponding sessions.

IV. Conclusion

Here, brief overview of SQL Injection and Cross Site Scripting security techniques have been presented. As per the referred papers false positive rate is high. The success of any IDS depends upon the available attack database and its accuracy. This study is directed towards the attacks over web application and use of different approach to overcome such intrusion.

References

[1] *V. Felmetsger, L. Cavedon, C. Kruegel and G.Vigna, "Toward Automated Detection of Logic Vulnerabilities in Web Applications", Proceeding USENIX Security ymp, 2010.*

[2] *G. Vigna, W.K. Robertson, V. Kher and R.A. Kemmerer, "A Stateful Intrusion Detection System for World-Wide Web Servers", Proceeding Ann. Computer Security Applications Conference (ACSAC '03), October 2003.*

[3] *C. Kruegel and G. Vigna, "Anomaly Detection of Web Based Attacks", Proceeding 10th ACM Conference Computer and Communication Security (CCS '03), October 2003.*

[4] *Liang and Sekar, "Fast and Automated Generation of Attack Signatures: A Basis for Building Self-Protecting Servers," SIGSAC: Proc. 12th ACM Conf. Computer and Comm. Security, 2005.*

[5] *Y. Hu and B. Panda, "A Data Mining Approach for Database Intrusion Detection", Proceeding ACM Symp, Applied Computing (SAC), H.Haddad, A.Omicini, R.L. Wainwright and L.M. Liebrock, eds, 2004.*

[6] *[Online] Available: https://www.owasp.org/index.php/Top_10_2013-Top_10*

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Author's Profile

R. Stalinbabu, Received B.E. (CSE) Degree in 2010 from M.I.E.T Engineering College, Tiruchirappalli, Tamil Nadu, India. Currently Pursuing M.E. (CSE) Degree in J.J. College of Engineering and Technology, Tiruchirappalli, Tamil Nadu, India.

P. Chellammal, Working as a Assistant Professor in J.J. College of Engineering and Technology, Tiruchirappalli, Tamil Nadu, India.