Dual Protecting Mechanism For Multitier Web Application

C. Preethi
Assistant Professor,Department of CSE
Veerammal Engineering College
Dindigul,Tamilnadu

R. Ramesh
Assistant Professor,Department of CSE
Veerammal Engineering College
Dindigul,Tamilnadu

Abstract*—Web application is an application that is accessed over a network such as the Internet. They are increasingly used for critical services, In order to adopt with increase in demand and data complexity web application are moved to multitier Design. As web servers must be publicly available around the clock the server is an easy target for outside intruders. Thus web applications are become a popular and valuable target for security attacks. These attacks have recently become more diverse and attention of an attacker have been shifted from attacking the front-end and exploiting vulnerabilities of the web applications in order to corrupt the back-end database system. In order to penetrate their targets, attackers may exploit well known service vulnerabilities. To protect multitier web applications several intrusion detection systems has been proposed. An intrusion detection system (IDS) is used to detect potential violations in database security. In every database, some of the attributes are considered more sensitive to malicious modifications compared to others.

***Keywords*— Light weight virtualization, Anomaly Intrusion detection, Container**

1. Introduction

Intrusion detection plays one of the key roles in computer system security techniques .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 alerts. There are two general approaches to intrusion detection: anomaly detection and misuse detection. A signature based IDS works similar to anti-virus software. It employs a signature database of well-known attacks, and a successful match with current input raises an alert. Similarly to anti-virus software, which fails to identify unknown viruses a signature-based IDS fails to detect unknown attacks. To overcome this limitation, researchers have been developing anomaly-based IDSs.

An Anomaly-Based Intrusion Detection System is a system for detecting computer intrusions and misuse by monitoring system activity and classifying it as either normal or anomalous. Double guard detection achieved by we employ a lightweight virtualization technique to assign each user’s web session to a dedicated container, an isolated virtual computing environment. We use the container ID to accurately associate the web request with the subsequent DB queries. Thus, Double Guard can build a causal mapping profile by taking both the web server and DB traffic into account. Full virtualization and Para virtualization are not the only approaches being taken, however. An alternative is lightweight virtualization, generally based on some sort of container concept. With containers, a group of processes still appears to have its own dedicated system, but it is really running in a specially isolated environment. All containers run on top of the same kernel. With containers, the ability to run different operating systems is lost, as is the strong separation between virtual systems. Double guard might not want to give root access to processes running within a container environment. On the other hand, containers can have considerable performance advantages, enabling large numbers of them to run on the same physical host.

2. Basic Survey

Research on SQL injection attacks can be broadly classified into two basic categories: vulnerability identification approaches and attack prevention approaches. The former category consists of techniques that identify vulnerable locations in a web application that may lead to SQL injection attacks. In order to avoid SQL injection attacks, a programmer often subjects all inputs to input validation and filtering routines that either detect attempts to inject SQL commands or render the input benign [5, 7]. The techniques presented in [8, 9] represent the prominent static analysis techniques for vulnerability identification, where code is analyzed to ensure that every piece of input is subject to an input validation check before being incorporated into a query (blocks of code that validate input are manually annotated by the user). While these static analysis approaches scale well and detect vulnerabilities, their use in addressing the SQL injection problem is limited to merely identifying

3. Related Work

A network intrusion detection system can be classified into two types: signature detection and anomaly detection .Anomaly detection first requires the IDS to define the characterize the correct and acceptable static from dynamic behavior of the system .It is used to detect the abnormal behavior of the system. We first define the normal behavior of the system and create profile of the user. In early IDS system that use the independent IDS used. Double guard use dependent IDS used. Double guard use the container ID using this ID each user session is assigned to each id.

Our approach does not require input validation, source code validation and know the application logic. Double guard uses the light weight virtualization to create and destroy the container by using the tool open Vs. identify the causal relationship between web server request and database request. Our approach dynamically generates new containers and recycle the used ones.

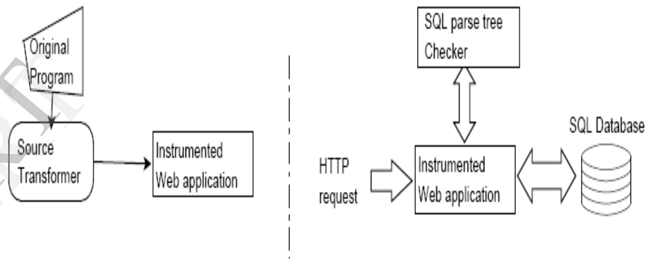
This paper makes the following contributions:

• The dynamic candidate evaluation approach for mining the structures of programmer-intended queries.

• A formal basis for this dynamic approach using the notion of symbolic queries.

A fully automated, program transformation mechanism for Java programs that employs this technique, with a discussion of practical issues and resilience to various artifacts of Web applications.

• A comprehensive evaluation of the effectiveness of attack detection and performance overheads.



4.Threat Model And System Architecture

We initially set up our threat model to include our assumptions and the types of attacks we are aiming to protect against. We assume that both the web and the database servers are vulnerable. 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.

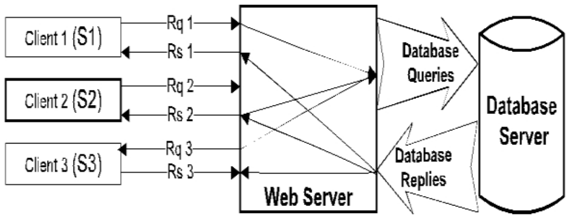


Fig1 Three tier architecture

Fig. 1 illustrates the classic three-tier model. At the database side, we are unable to tell which transaction corresponds to which client request. The communication between the web server and the database server is not separated, and we can hardly understand the relationships among them. According to Fig. 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.

5. Building The Normality Model

This container-based and session separated web server architecture not only enhances the security performances but also provides us with the isolated information flows that are separated in each container session. It allows us to identify the mapping between the web server requests and the subsequent DB queries, and to utilize such a mapping model to detect abnormal behaviors on a session/client level. In typical three-tiered web server architecture, the web server receives HTTP requests from user clients and then issues SQL queries to the database server to retrieve and update data. These SQL queries are causally dependent on the web request hitting the web server. We want to model such causal mapping relationships of all legitimate

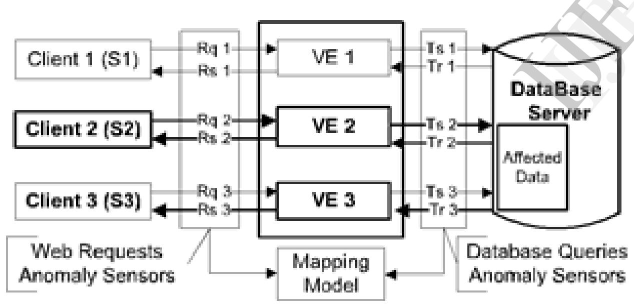


Fig 2.Double Guard model

Fig. 2 depicts how communications are categorized as sessions and how database transactions can be related to a corresponding session. Fig. 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.

It is impossible for a database server to determine which SQL queries are the results of which web requests, much less to find out the relationship between them. Even if we knew the application logic of the web server and were to build a correct model, it would be impossible to use such a model to detect attacks within huge amounts of concurrent real traffic unless we had a mechanism to identify the pair of the HTTP request and SQL queries that are causally generated by the HTTP request. However, within our container-based web servers, it is a straightforward matter to identify the causal pairs of web requests and resulting SQL queries in a given session. Moreover, as traffic can easily be separated by session, it becomes possible for us to compare and analyze the request and queries across different sessions. Section 4 further discusses how to build the mapping by profiling session traffics.

To that end, we put sensors at both sides of the servers. At the web server, our sensors are deployed on the host system and cannot be attacked directly since only the
Virtualized containers are exposed to attackers. Our sensors will not be attacked at the database server either, as we assume that the attacker cannot completely take control of the database server. In fact, we assume that our sensors cannot be attacked and can always capture correct traffic information at both ends. Fig. 2 shows the locations of our sensors.

Once we build the mapping model, it can be used to detect abnormal behaviors. Both the web request and the database queries within each session should be in accordance with the model. If there exists any request or query that violates the normality model within a session, then the session will be treated as a possible attack.

6. Modeling Deterministic mapping And Patterns

Due to their diverse functionality, different web applications exhibit different characteristics. Many websites serve only static content, which is updated and often managed by a Content Management System (CMS). For a static website, we can 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.

6.1 Inferring Mapping Reactions

Mapping relation explain about how the request and corresponding query are matched, causal relationship between rm to {qn,qp).Here qn,qp are mention the different database query. The possible mapping patterns as follows.

6.2 Deterministic mapping

This is the most common and perfectly matched pattern. That is to say that web request rm appears in all traffic with the SQL queries set Qn. The mapping pattern is then rm to Qn. In static websites, this type of mapping comprises the majority of cases since the same results should be returned for each time a user visits the same link.

6.3 Empty Query set

In special cases, the SQL query set may be the empty set. This implies that the web request neither causes nor generates any database queries.

6.4 No Matched Request

In some cases, the webserver may periodically submit queries to the database server in order to conduct some scheduled tasks, such as cron jobs for archiving or backup. This is not driven by any web request, similar tothe reverse case of the Empty Query Set mapping pattern. These queries cannot match up with any web requests, and we keep these unmatched queries in a set NMR. During the testing phase, any query within set NMR is considered legitimate. The size of NMR depends on webserver logic, but it is typically small.

6.5 Nondeterministic Mapping

The same web request may result in different SQL query sets based on input parameters or the status of the webpage at the time the web request is received. In fact, these different SQL query sets do not appear randomly, and there exists a candidate pool of query sets (e.g., {Qn;Qp;Qq . . .}). Each time that the same type of web request arrives, it always matches up with one (and only one) of the query sets in the pool. The mapping pattern is therefore, it is difficult to identify traffic that matches this pattern. This happens only within dynamic websites, such as blogs or forum sites.

7.Attack Detection

Once the model is built, it can be used to detect malicious Sessions. Double Guard detects the following types of attack.

7.1 Privilege Escalation Attack

Let’s assume that the website serves both regular users and administrators. For a regular user, the web request ru will trigger the set of SQL queries Qu; for an administrator, the request ra will trigger the set of admin level queries Qa.

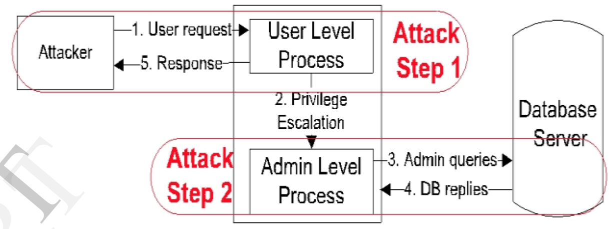


Fig3.Privilage Escalation Attack

Now suppose that an attacker logs into the web server as a normal user, upgrades his/her privileges, and triggers admin queries so as to obtain an administrator’s data. This attack can never be detected by either the web server IDS or the database IDS since both ru and Qa are legitimate requests and queries. Our approach, however, can detect this type of attack since the DB query Qa does not match the request ru, according to our mapping model.

7.2 Hijack Future Session Attack

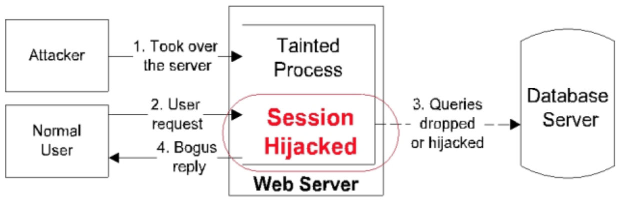


Fig4.Hijack future session attack

This class of attack is mainly based on web server side. An attacker takes over web server by hijack the other user sessions by sending spoofed replies. In double guard it is detected by causal mapping a request without query it is not accepted. 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 users' sessions.

7.3 Injection Attack

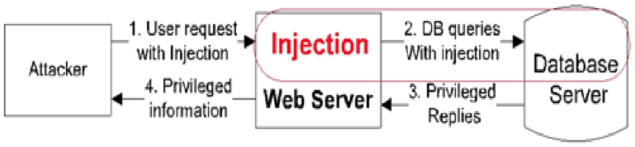

Attacker
1. User request with Injection
4. Privileged information
Injection
Web Server
2. DB queries With injection
3. Privileged Replies
Database Server

Fig5.Injection Attack

Attacks such as SQL injection do not require compromising the webserver. Attackers can use existing vulnerabilities in the webserver logic to inject the data or string content that contains the exploits and then use the webserver to relay these exploits to attack the back-end database. Since our approach provides a two-tier detection, even if the exploits are accepted by the webserver, the relayed contents to theDBserver would not be able to take on the expected structure for the given webserver 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 webserver 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.

7.4 Direct DB Attack

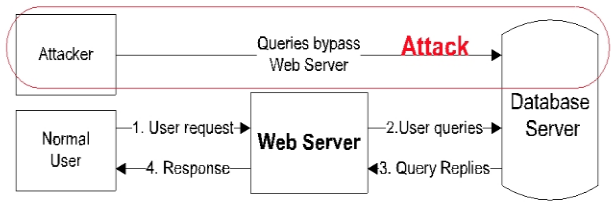


Fig5.Direct DB attack

It is possible for an attacker to bypass the webserver or firewalls and connect directly to the database. An attacker could also have already taken over the webserver and be submitting such queries from the webserver without sending web requests. Without matched web requests for such queries, a webserver IDS could detect neither. Furthermore, if these DB queries were within the set of allowed queries, then the database IDS itself would not detect it either. However, this type of attack can be caught with our approach since we cannot match any web requests with these queries

8. Conclusion

We have presented a novel technique to dynamically de-duce the programmer intended structure of SQL queries and used it to effectively transform applications so that they guard themselves against SQL injection attacks. We have also shown strong evidence that our technique will scale to most web applications. At a more abstract level, the idea of computing the symbolic query on sample inputs in order to deduce the intentions of the programmer seems a powerful idea that probably has more applications in systems security. There are many approaches in the literature on mining intentions of programmers from code as such intentions can be used as specifications for code, and detection of departure from intentions can be used to infer software vulnerabilities and errors [4, 3,10]. The idea of using candidate inputs to mine programmer intentions is intriguing and holds much promise.

9.References

[1]Yi Xie and Shun-Zheng Yu A Large-Scale Hidden Semi-MarkovModel for Anomaly Detection onuser Browsing Behaviors IEEE/ACM transactions on networking, vol. 17, no. 1, February 2009.

[2] Meixing Le, Angelos Stavrou, Brent Byung Hoon Kang, "DoubleGuard: Detecting Intrusions In Multi-tier WebApplications"2012

[3] National Vulnerability Database, "Vulnerability SummaryforCVE-2010-4333," http://web.nvd.nist.gov/view/vuln/detail? vulnId=CVE-2010-4333, 2011.

[4]Autobench, http://www.xenoclast.org/autobench/, 2011.

[5]"Common Vulnerabilities and Exposures," http://www.cve.mitre. org/, 2011.

[6]G. Vigna, F. Valeur, D. Balzarotti, W. K. Robertson, C. Kruegel, and E. Kirda. Reducing errors in the anomaly-based detection of web-basedattacks through the combined analysis of web requests and SQL queries.Journal of Computer Security, 2009.

[7]WordpressBug, ”http://core.trac.wordpress.org/ticket/5487, 2011.524 IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 9, NO. 4, JULY/AUGUST 2012

[8]Y. Hu and B. Panda, “A Data Mining Approach for Database Intrusion Detection,” Proc. ACM Symp. Applied Computing (SAC),

H. Haddad, A. Omicini, R.L. Wainwright, and L.M. Liebrock, eds.,2004.

[9] Seleniumhq, http://seleniumhq.org/, 2011.

[10] V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna, “Toward Automated Detection of Logic Vulnerabilities in Web Applications,” Proc. USENIX Security Symp., 2010.