A Container Based IDS to Automatically Trace the Web-Application Related Attacks

Shahanaz Begum and G. Geetharamani

BIT Campus, Anna University, Department of Information Technology, Trichy-620024, Trichy, India

BIT Campus, Anna University, Department of Mathematics, G. Geetharamani, Trichy-620024, Trichy, India

A R T I C L E  I N F O

Article history:
Received 12 October 2014
Received in revised form 26 December 2014
Accepted 1 January 2015
Available online 25 February 2015

Keywords:
Intrusion Detection System
Session hijacking attack
SQL Injection attack
e-commerce application

A B S T R A C T

The web has been embraced by millions of businesses as an inexpensive channel to communicate and exchange information with prospects and transactions with customers. This development of the Internet use has unfortunately been accompanied by a growth of malicious activity in the web application. The Session Hijacking attack compromises the session token by stealing or predicting a valid session token to gain unauthorized access to the Web Server. In the earlier works, a single Intrusion Detection System can detect attacks such as Session hijacking attack, SQL Injection attack and Privilege Escalation attack. Using existing system design still these attacks are not completely wiped off. To solve this problem, in this paper, we propose a container based IDS to detect such attacks and to prevent the compromise of a system, and the working environment used is Tomcat server and SQL Yog db server. We experimented and found that we were able to trace the occurrence of Session hijacking attack and sql injection attack and also we were able to prevent its occurrence. Conclusion: By incorporating our technique in the container of all java based web servers many such web application related attacks can be prevented.

INTRODUCTION

The growth of the Internet use has unfortunately been accompanied by a growth of malicious activity in the Internet. The use of the internet for accomplishing important tasks such as transferring a balance from a bank account always comes with a security risk. Today’s websites strive to keep their user’s data confidential and after years of doing secure business online, these companies have become experts in information security.

The database systems behind these secure websites store non-critical data along with sensitive information, in a way that allows the information owners quick access while blocking break-in attempts from unauthorized users. A common break-in strategy is to try to access sensitive information from a database by first generating a query that will cause the database parser to malfunction, followed by applying this query to the desired database. Such an approach to gaining access to private information is called SQL injection. Since databases are everywhere and are accessible from the internet, dealing with SQL injection (Common Vulnerabilities and Exposures, 2011) has become more important than ever.

This attack has become more diverse, as attention has shifted from attacking the front end to exploit the vulnerabilities of the web applications (Five Common Web Application Vulnerabilities, 2011; SANS, 2011) in order to corrupt the back-end database system (Schulman, A., 2011) (e.g., SQL injection attacks (Anley, C., 2002.; Newsome, J., 2005; Shin, Y., 2006).

Although current database systems have little vulnerability, the Computer Security Institute discovered that every year about 50% of databases experience at least one security breach. The loss of revenue associated with such breaches has been estimated to be over four million dollars.

Intrusion Detection Approaches:

An intrusion can be defined as any set of actions that attempt to compromise the integrity, privacy, confidentiality or accessibility of resources of the system. An Intrusion detection system aims to identify an intruder breaking or misusing system resources.
Misuse Detection:

It is based on the knowledge of vulnerabilities and known attack signatures. Misuse detection is concerned with detecting intruders who are attempting to break into a system by using some known vulnerabilities. Signature based IDS store patterns of known attacks. It uses stored behavior patterns to identify and detect attacks. It can detect only known attacks. The main drawback of signature based IDS is that it cannot detect new attacks or previously unseen attacks.

To protect multi-tier web services, Intrusion detection systems have been widely used to detect known attacks by matching misused traffic patterns or signatures ([Newsome, J., 2005; Kim, H., 2004; Liang and Sekar, 2005. Lee, S.Y., 2002; Barry, B.I.A. and H.A. Chan, 2009]).

Anomaly Detection:

Anomaly detection assumes that intrusions always reflect some variation from normal pattern. This type of IDS stores normal behavior of a system (using previously seen behavior). It is used to classify any behavior that violates it as attacks. Anomaly based IDS detects new attacks but it produces false alarm for legitimate but previously unseen system behavior which is termed as false positives. In this paper, we propose a framework, which accurately detects attacks in a web application. The design of this system stops the attacker’s chain of actions in performing various web application related attacks. The goal of the system is to detect the attacker at the container’s level itself.

Container-Based means that the container of the web application (e.g. tomcat, jboss, etc.) performs the authentication, and makes the results of the authentication known to the web application.

Container-based virtualization, also called operating system virtualization, is an approach to virtualization in which the virtualization layer runs as an application within the operating system. In this approach, the operating system’s kernel runs on the hardware node with several isolated client virtual machines installed on top of it. The isolated clients are called containers.

With container-based virtualization, there is no overhead associated with each client running a completely installed operating system. This approach can also improve performance because there is just one operating system taking care of hardware calls. A disadvantage of container-based virtualization, however, is that each client must use the same operating system the host uses.

Typically, corporate environments avoid container-based virtualization, preferring hypervisors and the option of having many operating systems. A container-based virtual environment, however, is an ideal choice for hosting providers who need an efficient and secure way to offer operating systems for customers to run services on.

This container-based IDS is used to gather information about an attacker or intruder into the system who is trying to access the resources.

1. Methodologies:

A network Intrusion Detection System can be classified into two types: anomaly detection and misuse detection. Anomaly detection first requires the IDS to define and characterize the correct and acceptable static form and dynamic behavior of the system, which can then be used to detect abnormal changes or anomalous behaviors [8], [31]. The boundary between acceptable and anomalous forms of stored code and data is precisely definable. Behavior models are built by performing a statistical analysis on historical data (Kruegel, C. and G. Vigna, 2003; Vigna, G., 2009) or by using rule-based approaches to specify behavioral patterns (Roesch, M., 2011). An anomaly detector then compares actual usage patterns against established models to identify abnormal events. It also fuses the alerts from different levels describing a single attack, with the goal of producing a succinct overview of security-related activity on the network.

DoubleGuard correlates alerts from independent IDSs. DoubleGuard operates on multiple feeds of network traffic using a single IDS that looks across sessions to produce an alert without correlating or summarizing the alerts produced by other independent IDSs. It handles efficiently in detecting the attacks which are targeted towards the backend database systems (SANS, 2011).

An IDS such as in (Seleznyov, A. and S. Puuronen, 1999) also uses temporal information to detect intrusions. DoubleGuard, (Meixing Le, 2012), does not correlate events on a time basis, which runs the risk of mistakenly considering independent but concurrent events as correlated events. DoubleGuard does not have such a limitation as it uses the container ID for each session to causally map the related events, whether they be concurrent or not.

Since databases always contain more valuable information, they should receive the highest level of protection.

Therefore, significant research efforts have been made on database IDS (Srivastava, A., 2004; Lee, S.Y., 2002) and database firewalls (Bai, K., 2005; Parno, B., 2009). These software, such as Green SQL, work as a reverse proxy for database connections. Instead of connecting to a database server, web applications will first connect to a database firewall. SQL queries are analyzed; if they’re deemed safe, they are then forwarded to the
back-end database server. The system proposed in (Vogt, P., 2007) 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. AutoBench is a tool which can be used to automatically compare the performance of two websites.

Detecting intrusions or vulnerabilities can be performed by various techniques. One of the techniques that can be done manually is through statically analyzing the source code or executables (Felmetsger, V., 2010; Christodorescu, M. and S. Jha, 2003). Tracking the information flow dynamically is for understanding taint propagations and for detecting intrusions. (Sekar, R., 2009; Kim, H., 2004). DoubleGuard, the container-based webserver architecture enables us to separate the different information flows by each session.

For each session this tracking of information has to be carried out from the webserver to the database server for detecting the attack targeted in each session.

To build a static model for web services, application logic is not required by DoubleGuard approach. But for developing a dynamic webservice model the basic user operations must be known. There are approaches to dynamically track the information flow in order to detect the intrusions. Behavior models are built by performing a statistical analysis on historical data or by using rule-based approaches to specify behaviour (Vigna, G., 2009) patterns.

In addition, validating input is useful to detect or prevent SQL or Cross Site Scripting (XSS) injection attacks (Bates, D., 2010; Pietraszek, T. and C.V. Berghe, 2005). This is orthogonal to the DoubleGuard approach, which can utilize input validation as an additional defense.

DoubleGuard can detect SQL injection attacks by taking the structures of web requests and database queries without looking into the values of input parameters (i.e., no input validation at the webserver).

Virtualization is used to isolate objects and enhance security performance. Full virtualization and para-virtualization are not the only approaches being taken. An alternative is a lightweight virtualization, such as OpenVZ (Hu, G. and B. Panda, 2004), Parallels Virtuozzo (Kruegel, C. and G. Vigna, 2003), or Linux-VServer. In general, these are based on some sort of container concept. With containers, a group of processes still appears to have its own dedicated system, yet it is running in an isolated environment.

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. There are also some desktop systems (Huang, Y., 2008; Potter, S. and J. Nieh, 2010) that use lightweight virtualization to isolate different application instances. Such virtualization techniques are commonly used for isolation and containment of attacks. In DoubleGuard, the container ID is used to separate session traffic as a way of extracting and identifying causal relationships between webserver requests and database query events.

CLAMP [20] is an architecture for preventing data leaks even in the presence of attacks. By isolating code at the webserver 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.

The Container-based IDS discussed in our work can perform well in automatically identifying different types of web application related attacks.

2. Attack Scenarios:
2.1 SQL Injection Attack:

SQL injection is one of the most common type of attack in web connected Databases. Attacker inserts an unauthorized SQL statement through SQL data channel. This attack is caused by non validated input parameters. SQL injection attack is one of the most prominent threats today. SQL injection is a security vulnerability that occurs in the database layer of an application.

![Fig. 1: SQL Injection Attack.](image)

2.2 Hijack future session attack:

This attack is mainly aimed at the Web Server. An attacker takes over the web server and hijacks all the subsequent legitimate user sessions to launch attacks.

2.3 Threat model and system architecture:

The threat model is setup to focus on the different attacks associated with web applications. 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. The attackers can bypass the web server to directly attack the database server. The attackers try to elevate their privileges by impersonating the administrator. They target the server to make services unavailable to the legitimate users. The attackers could modify the web application’s logic, eavesdrop or hijack other users’ web requests, or intercept and modify the database queries to steal sensitive data beyond their privileges.

Fig. 2: Session Hijacking Attack.

The attackers may try to exploit all the vulnerabilities to either take control over the webserver or database server by performing all possible attacks.

On the other hand, at the database end, we assume that the database server will not be completely taken over by the attackers. Attackers may strike the database server through the webserver 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 that attack would occur during the course of e-shopping by the client.

3. Results:

3.1 Implementation:

This framework deals with the container based IDS in detecting Session hijacking attack and SQL Injection attack performed by the attacker against any web application in any web server platform.

The proposed framework was implemented by TOMCAT Web Server and SQL Yog as the DB Server, the framework acts as the container based IDS in detecting the SQL Injection attack and Session Hijacking attack and related attacks of the web application. It gets the input from user and validates the user input parameter before it’s assigned to queries. Usually the intruder uses the blind and illegal/incorrect sql injection techniques to know about the database information from error messages which are provided by the database.
Algorithm 1. Dynamic Threat Model Building Algorithm:

Require: Attacks are defined in the patch file
Ensure: The container based IDS is detecting attacks occurring against the e-commerce website
1: for each session separated traffic Ti the IDS do check 
1.1 if any invalid requests are sent continuously 
1.2 the IDS available with the container invalidates the session 
1.3 the system is prevented from session hijacking attack 
2: for each session Get different HTTP requests r and DB queries q in this session 
3: for each different r do 
4: if r is a request to a dynamic site then 
5: Perform validation and sanitization steps 
6: if any invalid datas are entered in the login field by padding special characters 
6.1 the IDS detects this SQL Injection attacks 
6.2 the malicious user is prevented from accessing the DB Server 

Discussion:
We created a dynamic web application for an e-commerce application dealing with the database created using SQL Yog Db server and the web server environment is created using TOMCAT server. The executable files performing session hijacking attack and SQL Injection attacks are introduced in the container of the Web server assuming that the attacker has maliciously entered the code for performing the above said attacks, at the same time the code for detecting the occurrence of Session Hijacking attack and SQL Injection attacks are also introduced in the same container which helps in automatically tracing the occurrence of such attacks. And it is experimentally proved that these attacks were detected at the time of its occurrence and it provides the countermeasures to prevent the occurrence of these attacks.

This paper deals with the development of dynamic web site for shopping cart application whenever the user performs shopping, the different web application related attacks can be detected by the IDS. The attacker who is introduced into the application as an executable code, tries to send continuously the invalid requests after hijacking the session id of the user The IDS available with the container invalidates the session of the user thinking that there is some malicious user who is trying to perform some mischief. The SQL Injection attack module is also attached as a patch file along with the container, when the user tries to give his username and pwd the attacker tries to pad special characters to the username field, but the IDs detects this sql injection and prevents him from accessing the db server.

Tomcat server, as web server and SQL Yog as db server are used in this framework. This container based IDS is created as a patch file using servlets. This patch file can work in any container of the following java based Webservers such as jboss, BEA Weblogic and glassfish.

Web container can be inbuilt in the web server or it can be separate from the web server. With the Apache tomcat server either jboss or weblogic or glassfish can be linked by implementing API’s of the container in the IDS.

Tomcat container has Servlet API and JSP API known as the references. If the Apache Server connected to PHP Web Container only PHP code can be run.
By Adding the patch file a container becomes a secure container to the web server. This patch file can be used with any JAVA Webservers such as Glassfish, Jboss and BEAWeblogic and Tomcat. When server runs this patch file or executable file is added to the Tomcat container.

Conclusion:
Since millions of users are using this TOMCAT Server holding the GOOGLE information we have chosen TOMCAT Server in our work.
The Tomcat Manager who is monitoring the various activities can prevent any type of Session hijacking, SQL Injection and the other web application related attacks.
We presented an intrusion detection system that builds the threat model of malicious behavior for multiltier web applications from both front-end web (HTTP) requests and back-end database (SQL) queries. Unlike previous approaches this container-based IDS with the in-built patch or executable file the attacks are performed and also they are detected at an earlier stage itself before it is propagated to the DB Server.
In our work we attempted to model dynamic web requests with the back-end SQL Yog DB Server and TOMCAT as the Web server.
Our experiments proved to be effective at detecting different types of attacks for dynamic requests where both retrieval of information and updates to the back-end database occur using the webserver with the requests initiated from the client end involving both normal and malicious users. This technique can be extended for detecting other types of web application attacks such as XSS[34] and CSRF attacks and other related attacks.

REFERENCES


Meixing Le, Angelos Stavrou, Member, IEEE, and Brent ByungHoon Kang, Member, IEEE, July/August 2012., DoubleGuard: Detecting Intrusions in Multitier Web Applications, IEEE Transactions on Dependable and Secure Computing, 9-4.


