Volume 5, Issue 3, March 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

IDS and IPS System in Multi-Tier Web Applications

Jayant Kulkarni, Namrata Walunj, Chetan Jagtap, Prof. Anuja Divekar
G. S. M.C.O.E, Pune, Maharashtra,
India

Abstract— ***In the age of information technology the facets of work and availability of everything on the internet services so the internet service and applications have become an inextricable part of day today life, enabling communication and the management of personal information from anywhere. To accommodate this increase in application and data complexity, web and its various services have moved to a multi-tiered system wherein the web server runs the application front-end logic and data is outsourced to a database or file server. In this paper, we present IDS–IPS SYSTEM, an IDS system that models the network behavior of user sessions across both the front-end web server and database present at the back end. By analyzing both web and related database requests, we are able to find out attacks that independent IDS would not be able to identify. Furthermore, we quantify the limitations of any multitier IDS in terms of training sessions and functionality coverage. We implemented IDS–IPS SYSTEM using an Apache web server with MySQL and lightweight virtualization. The real-world data was collected and processed also traffic over a 15-day period of system deployment in both dynamic and static web applications. Finally, using IDS–IPS SYSTEM, we were able to expose a wide range of attacks with 100% accuracy while maintaining 0% false positives for static web services and 0.6% false positives for dynamic web services.***

Keywords—IDS-IPS system, Multi-tier, Web based attack, SQL Injection, Vulnerable.

I. INTRODUCTION

Web-delivered services and applications have increased in both popularity and complexity over the recent years. Daily tasks, such as banking, traveling, and e-commerce, social networking, are all done with the help of web. Such services typically deploy a web server front-end that gives functionality to the application user interface logic, as well as a back-end server that include a database or file server. Due to their present or actual use for personal and/or corporate data, Which is the most vulnerable for attacks.These attacks are various in nature, these attacks are transferred to exploiting vulnerabilities of the web applications [6], [5], [1] in order to corrupt the back-end database system [20] (e.g., SQL injection attacks [20], [30]). A plethora of Intrusion Detection Systems (IDS) currently examine network packets individually within both the web server and the database system but there is very little work being done on multi-tiered Anomaly Detection (AD) systems that generate models of network behavior for both web and database network interactions. In such multi-tiered architecture systems, the back-end DB(database) server is often protected behind a firewall while the web servers are remotely accessible over the Internet. Unfortunately, they are protected from direct remote attacks ,but the back-end systems are vulnerable to to attacks that use web requests as a means to exploit the back-end. To protect multi-tiered web services, Intrusion detection systems (IDS) have been widely used to detect known attacks by matching misused traffic patterns or signatures [34], [30], [13], [22]. 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 Intrusion detection training phase. Individually, the web IDS and the DB IDS can detect abnormal network traffic sent to one out of them. However, we found that these IDS cannot detect cases wherein normal traffic is used to attack the web server and the database server. For example, if an attacker with unauthorized privileges can log in to a web server using authorized user access credentials, he/she can access to issue a privileged database query by finding vulnerabilities in the web server. Both the web IDS and the database IDS would not detect this type of attack since the web IDS would hardly see typical user login traffic and the database IDS would see only the normal traffic of a privileged user. Within a lightweight virtualization environment, we ran many copies of the web server instances in different containers so that each one was isolated from the rest. As ad-hoc containers can be easily instantiated and demolished, we assigned each client session a individual container so that, even when an attacker may be able to access a single session, the damage is limited to the compromised session; other user sessions remain unaffected by it.

Using our prototype, we show that, for websites that do not grant the content modification from users, there is a direct relationship between the requests received by the frontend web server and those generated for the database backend. In fact, we show that this causality-mapping model can be generated accurately and without prior knowledge of web application functionality.

II. LITERATURE SURVEY

I initially set up our threat model to include our assumptions and the attacks we want to protect against. We assume that both the web and the DB servers are vulnerable. Attacks are network-based and come from the web clients; they can launch application-layer attacks to compromise the web servers they are connecting to. The attackers can move the web server to directly attack the DB server. We assume that these attacks cannot be detected or prevented by the current web server Intrusion Detection System, that can be taken over by attacker and the web server after the attack, and they can obtain full control over the web server to launch related attacks.this can be explained , the attackers could modify the application logic related to the web applications, he can secretly access or hijack other users web requests, he can intercept or modify the DB queries to steal sensitive data beyond their privileges.

III. THREAT MODEL & SYSTEM ARCHITECTURE

We initially created our threat model to include our assumptions and the types of attacks we are aiming to protect against. We assume that ,the web and the database servers both are vulnerable. Attacks are network-based and come from the web clients; they can launch application-layer attacks to compromise the web servers they are connecting to.[10] The attackers can bypass the web server to directly attack the database server. We assume that the attacks cannot be detected or prevented by the current web server IDS, those attackers may move to the web server after the attack, and later they can obtain full control of the web server to launch related attacks. Attackers may attack the database server by the web server or directly by submitting SQL queries in it., they may obtain the sensitive data within the database. These assumptions are right since, in many cases, the DB server is not exposed to the public and is therefore difficult for attackers to completely take over.

A. Architecture & Confinement

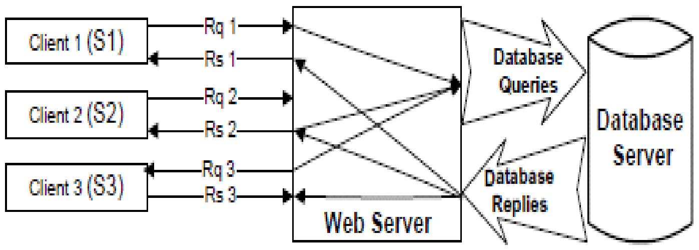

Fig. 1 Classic three-tier Model.

The web server which acts as the front-end, with the file and DB as the content storage back-end.

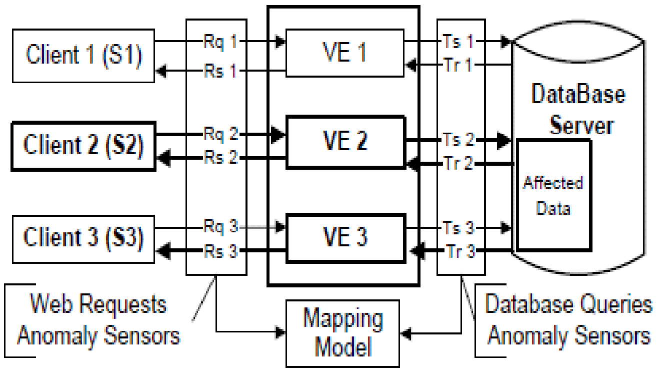

Fig. 2 Web server instances running in containers.

All network traffic, from both authorized users and attackers, is received intermixed at the same web server. If an attacker compromises the web server then he can potentially affect all future sessions (i.e., session hijacking). Assigning each session to a dedicated web server is a unrealistic option, as it will reduce the web server resources. To achieve similar confinement while maintaining a low performance and resource overhead, we use lightweight virtualization. In our design, we make use of these process containers, referred to as containers for client sessions. It is possible to initialize many of containers on a single machine and these virtualized containers can be removed, or quickly initialized to serve new sessions. A single web server runs many containers with having same copy of the original web server. Our approach generates new containers and recycles used containers. As a result a single physical server can run and serve all web requests. Sessions can represent different users to some extent, and also we can expect the communication of a single user to go to the same dedicated web server, allowing us to identify abnormal behavior by both session and user. If we detect suspected behavior in a session then we will treat all traffic within this session as tainted.

B. 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 to identify the mapping between the web server requests with the related database queries and to utilize such a mapping model to detect behaviors on a session or client level which are abnormal. In typical 3-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 generally dependent on the web request hitting the web server.[18] We want to model such causal mapping relationships of all legitimate traffic so as to detect abnormal/attack traffic. In practice, we are unable to build such mapping under a classic 3-tier setup. Although the web server can distinguish sessions from different clients also the SQL queries are mixed and all from the same web server. It is impossible for a DBserver to determine which SQL queries are the results of which web requests, also much less to find out the relationship between them.[12] Even if we knew the application logic of the web server and were to build a correct model, it would not be possible 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 generated by the HTTP request. But, within our container-based web servers, it is a matter to identify the causal pairs of web requests and resulting SQL queries in a session. Moreover, it becomes possible for us to compare and analyze the request and queries across different sessions. Because traffic can easily be separated by session [14]

C. Attack Scenarios

Our system can capture the following types of attacks: 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). Now suppose that an attacker logs into the web server as a normal user upgrades his privileges and also 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 DB IDS since both RA and Aare legal requests and queries. Our approach can detect this type of attack since the data-base query (QA) does not match the request RU, according to our mapping model.

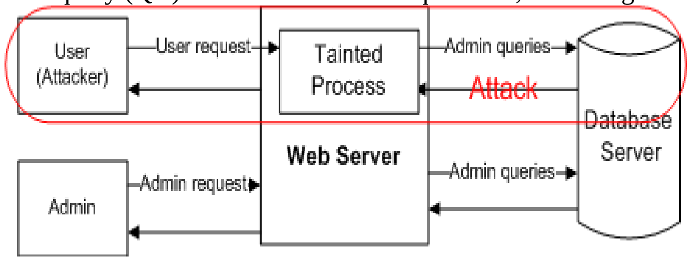

Fig. 3. Privilege Escalation Attack.

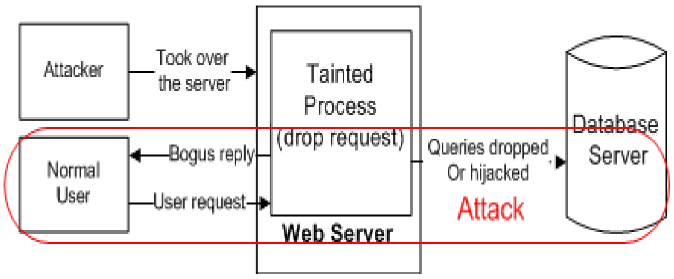

Fig. 4. Hijack Future Session Attack.

Figure 3 shows how a normal user may use admin queries to obtain privileged information. Hijack Future Session Attack: This class of attacks is mainly aimed at the side of web server. An attacker generally takes over the web server and hence hijacks all subsequent legitimate user sessions to launch attacks. For example by hijacking other user sessions, the attacker can spy or send spoofed replies, and drop user requests. [11]

Fortunately, the isolation property of our container-based web server architecture can also prevent this type of attack because each user's web requests are isolated into a separate container and an attacker can never break into other user's sessions. Injection Attack: Attacks such as SQL injection do not require compromising the web server. Attackers can use vulnerabilitiespresent 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 database. Since our approach provides a 2-tier detection even if the exploits are accepted by the web server, the relayed contents to the database server would be unable to take on the expected structure for the given web server request. For example, 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 seek such a web request. Such queries from the web server without sending web requests.[8] Fig.5 shows the SQL Injection attack.

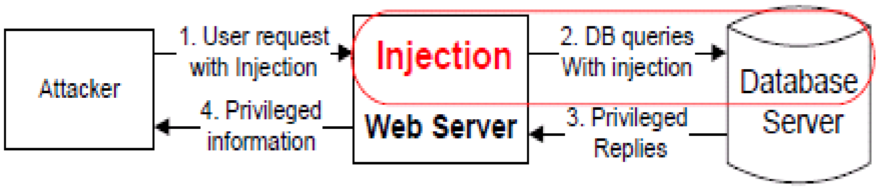
Attacker
1. User request with Injection
4. Privileged information
Injection
Web Server
2. DB queries With injection
3. Privileged Replies
Database Server

Fig.5 SQL Injection Attack

Without matched web direct database attack, It is possible for an attacker to bypass the web server or firewalls and connect directly to the database. An attacker could also have already taken over the web server and be submitting requests for such queries, a web server IDS could detect neither. Also, if these database queries were within the set of allowed queries, then the database IDS it would not detect it either. However, this type of attack can be caught with our approach since we are unable match any web requests with these queries.[11] Fig.6 shows the scenario wherein an attacker bypasses the web server to directly query the database.

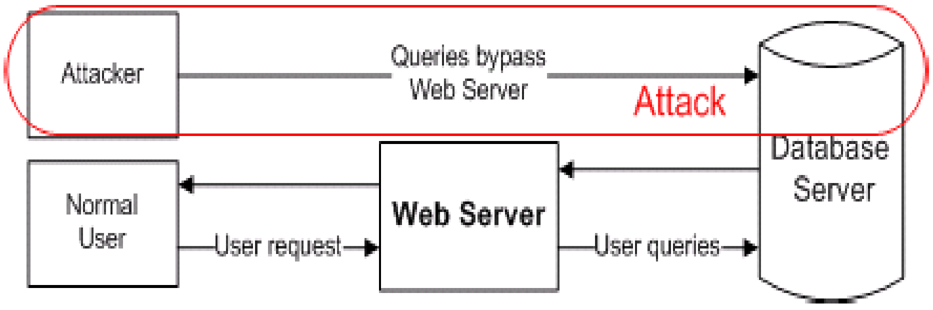

Fig. 6 Database Queries without web request

IV. IDS AND IPS SYSTEM LIMITATIONS

In this section, we discuss the operational and detection limitations of IDS and IPS system. Vulnerabilities Due to Improper Input Processing: Cross Site Scripting (XSS) is a typical attack method where the attackers embed malicious client scripts via valid user inputs. In IDS and IPS system, all of the user input values are normalized so as to build a mapping model based on the structures of HTTP requests and DB queries. Once the malicious user inputs are normalized, IDS and IPS system cannot detect attacks hidden in the values. These attacks can occur even without the databases. IDS and IPS system offers a complementary approach to those research approaches of detecting web attacks based on the characterization of input values [28]. Possibility of Evading IDS and IPS system: Our assumption is that an attacker can obtain “full control” of the web server thread that she connects to. That is, the attacker can only take over the web server instance running in its isolated container. Our architecture ensures that every client be defined by the IP address and port container pair, which is unique for each session.[24] Therefore, hijacking an existing container is not possible because traffic for other sessions is never directed to an occupied container. If this were not the case, our architecture would have been similar to the conventional one where a single web server runs many different processes. Moreover, if the DB authenticates the sessions from the web server, then each container connects to the DB using either admin user account or non- admin user account and the connection is authenticated by the database.[16] In such case an attacker will authenticate using a non-admin account and will not be allowed to issue the admin level queries. In other words the HTTP traffic defines the privileges of the session which can be extended to the backend DB, and a non-admin user session cannot appear to be an admin session when it comes to back-end traffic.
Within the same session that the attacker connects to it is allowed for the attacker to launch mimicry attacks. It is possible for an attacker to discover the mapping patterns by doing code analysis or reverse engineering, and issue expected web requests prior to performing malicious DBqueries. However, this significantly increases the efforts for the attackers to launch successful attacks. In addition, users with non-admin permissions can cause minimal (and sometimes zero) damage to the rest of the system and therefore they have limited incentives to launch such attacks.

V. MODELING DETERMINISTIC MAPPING AND PATTERNS

Due to their diverse functionality, various web applications exhibit different characteristics. Most websites serve only static content, which is updated and frequently 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. However, some websites (for example blogs, forums) allow regular users with non-administrative privileges to update the contents of the served data. This creates tremendous challenges for intrrusion detection system system training because the HTTP requests can contain variables in the passed parameters.

V. PERFORMANCE EVALUATION

We implemented a prototype of IDS-IPS using a web server with a back-end database. We also set up two testing websites, static and the dynamic. To find out the detection results for our system, we analyzed four types of attacks, as discussed in Section III, and measured the false positive rate for each of the two websites.

A. Implementation

We obtained 320 real user traffic sessions from the blog under daily workloads. During this seven-day phase, we made our website available only to internal users to ensure that no attacks would occur then generated 20 attack traffic sessions mixed with these legitimate sessions, and the mixed traffic was used forwe model nine basic operations, we can reach 100 percentSensitivity with six percent False Positive rate. In the case of 23 basic operations, we achieve the False Positive rate of 0.6 percent. By Using Double guard approach we also avoid the following attacks.

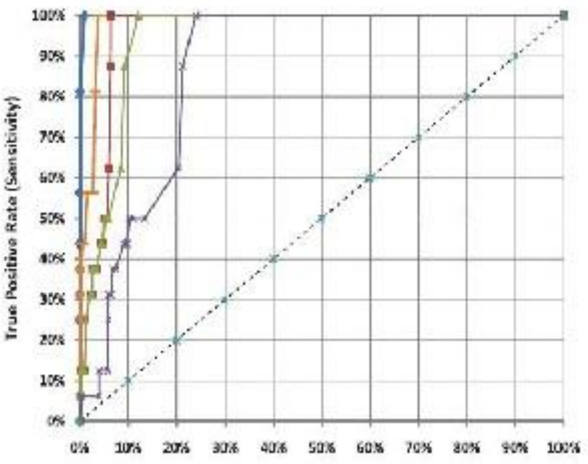

Fig. 7 ROC curve for Dynamic models

B. Container Overhead

One of the primary concerns for a security system is its performance overhead in terms of latency. In our case, even more the containers can start within seconds, producing a container on-the-fly to serve a new session will increase the response time heavily. To reduce this, we created a pool of web server containers for the forthcoming sessions akin to what Apache does with its threads.[2] As sessions continued to grow, in our system dynamically new containers instantiated. Upon completion of a session, we recycled containers by reverting them to their initial clean states.

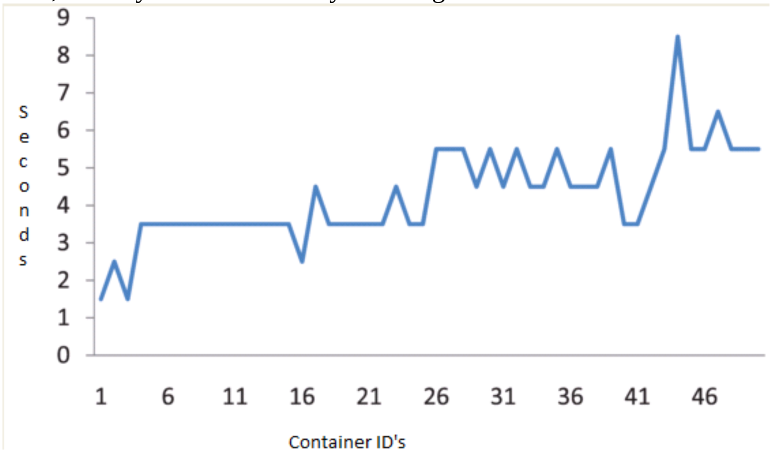

Fig.8 Time for starting container

C. Static website model in training phase

For the static website, we used the algorithm to construct the mapping model, and we discovered that only the Deterministic Mapping and the Empty Query Set Mapping patterns appear in the training sessions. We expected that the No Similar Request pattern would appear if the web application had a cron job that contacts back-end database server; although, our testing website did not have a cron job. We first collected 338 actual user sessions for a training dataset before making the website public so that there was no attack during the training phase

D. Attack Detection

Once the model is built, it can be used to detect malicious sessions. For static website testing, we used production website, which has methodical visits of around 50-100 sessions per day. For the production site we collected regular traffic , which totaled 1172 sessions.[7]

VI. CONCLUSION

We presented an intrusion detection system that builds models of normal behavior for multi-tiered web applications from both front-end web (HTTP) requests and back-end database queries. Unlike previous approaches that summarized alerts generated by independent IDSes, DoubleGuard forms a container-based IDS with multiple input streams to produce alerts. Such correlation of variant type of data streams provides a better characterization of the system for anomaly detection because the intrusion sensor has a more precise normality model that detects a wider range of threats.[6] We achieved this by isolating the flow of information from each web server session with a lightweight virtualization.Moreover, we quantified the detection accuracy of our approach when we attempted to model static and dynamic web requests with the back-end file system and database (SQL) queries. For static websites, we built a wellcorrelated model, which our experiments proved to be more effective at detecting different types of attacks. However, we showed that this held true for dynamic requests where both retrieval of information and updates to the back-end database occur using the web-server front end. [9]

REFERENCES

[1] Meixing Le, Angelos Stavrou, DoubleGuard: Detecting Intrusions in Multitier Web Applications. In IEEE transactions, 2014

[2] K. Kavitha, S.V. Anandhi, Intrusion Detection Using Double guard in Multi-tier Architecture. In IJIRCCE, 2014

[3] K. Bai, H. Wang, and P. Liu. Towards database firewalls. In DBSec 2005.

[4] B. I. A. Barry and H. A. Chan. Syntax, and semantics-based signature database for hybrid intrusion detection systems. Security and Communication Networks, 2(6), 2009.

[5] D. Bates, A. Barth, and C. Jackson. Regular expressions inspect harmful in client side xss filters. In ordering of the 19th international conference on World wide web, 2010.

[6] Y. Hu and B. Panda. A data mining approach for database intrusion detection. In H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock, editors, SAC. ACM, 2004.

[7] Y. Huang, A. Stavrou, A. K. Ghosh, and S. Jajodia. Efficiently tracking application interactions using lightweight virtualization. In Proceedings of the 1st ACM workshop on Virtual security machine , 2008.

[8] H.-A. Kim and B. Karp. Autograph: Toward the automated, classified worm signature detection. In USENIX Security conference , 2004.

[9] J. Newsome, B. Karp, and D. X. Song. Polygraph: Automatically generating signatures for polymorphic worms. In IEEE conference onSecurity and Privacy. Computer Society IEEE, 2005.Symposium on Security and Privacy. IEEE Computer Society, 2009.

[10] P. Krishna Reddy, T. Manjula,D. Srujan Chandra Reddy, Double guard:detecting interruptions in N-Tier web Applications. In IJMER,2013