DETECTING SQL INJECTION ATTACK IN MULTITIER WEB APPLICATIONS

S.V. Evangelin Sonia[1] & D. Sathiya[2]
[1,2]Assistant Professor, Sri Shakthi Institute of Engineering and Technology

Received: August 23, 2018 Accepted: October 14, 2018

ABSTRACT *Web-delivered services and applications have increase in both popularity and complexity over the past few years. All network traffic, from both legitimate users and adversaries, is received intermixed at the same web server. Due to their ubiquitous use for personal and/or corporate data, web services have always been the target of attacks one such attack is SQL injection attack. In this paper, we present Double Guard, an IDS system that protects the database from SQL injection attack. Here we use containers for storing the data instead of storing the data randomly in the database. Each container is given a container-id and container-IDS is used to give alert messages to the system administrator, here normal IDS is replaced by container-IDS. Double Guard provides high security from direct database attacks that is, even a legitimate user can access only if he knows the username, password and the container-id. We implemented Double Guard using an Apache web-server with MySQL and lightweight virtualization.*

1. 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 webserver 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 webserver 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 webservers 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, we found that these IDSs cannot detect cases wherein normal traffic is used to attack the webserver and the database server. For example, if an attacker with non-admin privileges can log in to a webserver using normal-user access credentials, he/she can find a way to issue a privileged database query by exploiting vulnerabilities in the webserver. 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 webserver is not associated with user-privileged access. Unfortunately, within the current multithreaded webserver architecture, it is not feasible to detect or profile such causal mapping between webserver traffic and DB server traffic since traffic cannot be clearly attributed to user sessions.

2. EXISTING SYSTEM

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. 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 or by using rule-based approaches to specify behavior patterns. An anomaly detector then compares actual usage

patterns against established models to identify abnormal events. Our detection approach belongs to anomaly detection, and we depend on a training phase to build the correct model. As some legitimate updates may cause model drift, there are a number of approaches that are trying to solve this problem. Our detection may run into the same problem; in such a case, our model should be retrained for each shift.

Intrusion alerts correlation provides a collection of components that transform intrusion detection sensor alerts into succinct intrusion reports in order to reduce the number of replicated alerts, false positives, and non-relevant positives. 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. It focuses primarily on abstracting the low-level sensor alerts and providing compound, logical, high-level alert events to the users.

2.1 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. Now suppose that an attacker logs into the webserver 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 webserver 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. Figure shows how a normal user may use admin queries to obtain privileged information.

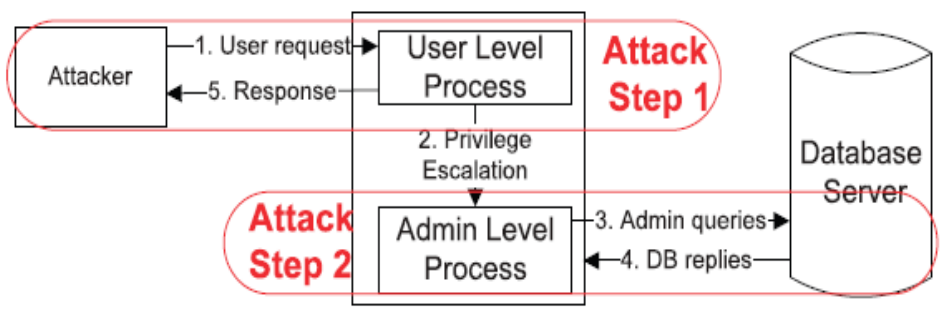

Fig 1. Privilege escalation attack.

2.2 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 the DB server 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. Figure illustrates the scenario of a SQL injection attack.

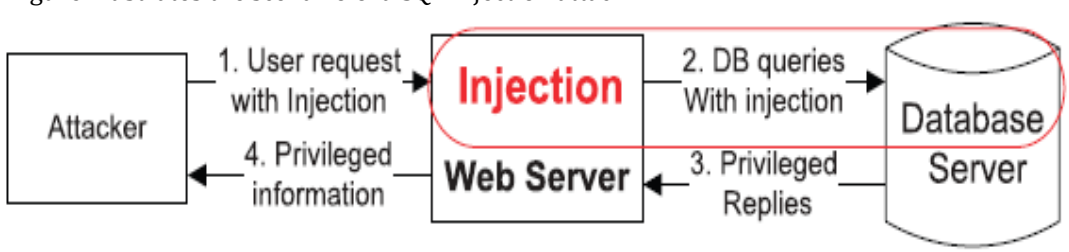

Fig 2. Injection attack.

2.3 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. Figure, illustrates the scenario wherein an attacker bypasses the webserver to directly query the database.

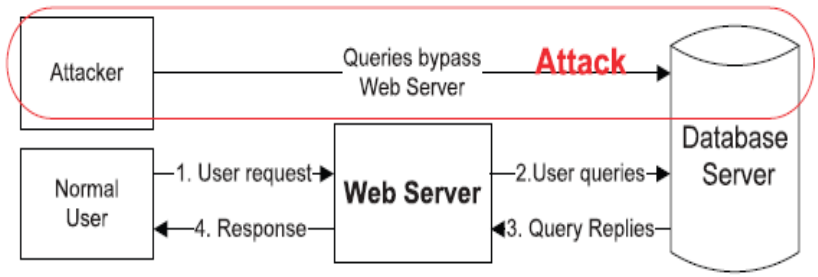

Fig 3. DB Query without causing web requests.

3. PROPOSED SYSTEM

We present Double Guard, a system used to detect attacks in multitier web services. To achieve this, 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.

We have implemented our Double Guard container architecture using Open VZ, and performance testing shows that it has reasonable performance overhead and is practical for most web applications. When the request rate is moderate (e.g., under 110 requests per second), there is almost no overhead.

The container-based web architecture prevents future session-hijacking attacks. 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 ephemeral containers can be easily instantiated and destroyed, we assigned each client session a dedicated container so that, even when an attacker may be able to compromise a single session, the damage is confined to the compromised session; other user sessions remain unaffected by it. The advantage of using this concept is, it provides more security that is, and even a legitimate user can login only if he knows the user name, password, and the container-id so SQL injection attack is not possible in our approach as we do not allow any of the SQL queries to enter in.

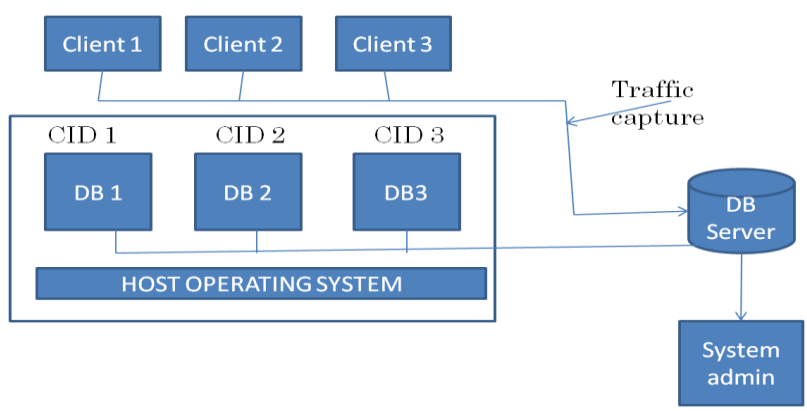

Fig 4. The overall architecture of our prototype.

The above fig illustrates that every client request first reaches the database server where it is processed and the response is obtained. Here each user data is stored separately in a container an each container is given a container id called CID so, this approach will overcome the above mentioned database attacks and if any attacker tries to attack the database the IDS itself will give alert messages to the system administrator.

4. REFERENCES

1. Meixing, Le, Angelos Stavrou, Brent Byung "DoubleGuard: Detecting Intrusions in Multitier Web Applications", IEEE transactions on dependable and secure computing, vol. 9, no. 4, July/august 2012.
2. S. Potter and J. Nieh, "Apiary: Easy-to-Use Desktop Application Fault Containment on Commodity Operating Systems," Proc. USENIX Ann. Technical Conf., 2010.

3. 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.
4. R. Sekar, "An Efficient Black-Box Technique for Defeating Web Application Attacks," Proc. Network and Distributed System Security Symp. (NDSS), 2009.
5. G. Vigna, F. Valeur, D. Balzarotti, W.K. Robertson, C. Kruegel, and E. Kirda, "Reducing Errors in the Anomaly-Based Detection of Web-Based Attacks through the Combined Analysis of Web Requests and SQL Queries," J.
6. 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.
7. V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna, "Toward Automated Detection of Logic Vulnerabilities in Web Applications," Proc. USENIX Security Symp. 2010.
8. B.I.A. Barry and H.A. Chan, "Syntax, and Semantics-Based Signature Database for Hybrid Intrusion Detection Systems," Security and Comm. Networks, vol. 2, no. 6, pp. 457-475, 2009.
9. B. Parno, J.M. McCune, D. Wendlandt, D.G. Andersen, and A. Perrig, "CLAMP: Practical Prevention of Large-Scale Data Leaks," Proc. IEEE Symp. Security and Privacy, 2009.
10. Y. Huang, A. Stavrou, A.K. Ghosh, and S. Jajodia, "Efficiently Tracking Application Interactions Using Lightweight Virtualization," Proc. First ACM Workshop Virtual Machine Security, 2008.
11. M. Cova, D. Balzarotti, V. Felmetsger, and G. Vigna, "Swaddler: An Approach for the Anomaly-Based Detection of State Violations in Web Applications," Proc. Int'l Symp. Recent Advances in Intrusion Detection (RAID '07), 2007.
12. D. Wagner and D. Dean, "Intrusion Detection via Static Analysis," Proc. Symp. Security and Privacy (SSP '01), May 2001.