TWIN PROTECTION TO AVOID INTRUSIONS IN MULTITIER DYNAMIC WEB APPLICATIONS

[1]Sandeep Shinde, [2]Prashant Kumbharkar
Department of Computer Engineering, Siddhant College of engineering, Sudumbare, Pune
[1]smshinde09@gmail.com, [2]pbk.rscoe@gmail.com

Abstract: Today most of the web applications used three tier web applications. With the increased new technologies, on the other side web site hacking, intrusion also becomes big issues in developed environments. For this problem we proposed Twin Protection to multitier dynamic web application. This system used to detect attacks in multi-tiered web services. This approach can create routine copies of lonely user sessions that contain both the HTTP and back end network communications. Websites that do not permit gratified alteration from user; and there is a direct causal connection between the requests acknowledged by the front-end web server and those generated for the database back end. The user doesn‟t know information of the source code or the application logic of web services deployed on the web server. Virtualization is used to detach objects and enhance security routine. Insubstantial containers can have large routine advantages over full virtualization**.**

Keyword: SQL Injection, IDS, XSS, Anomaly Detection, Dynamic Web, Session Hijacking, J2EE.

1. INTRODUCTION

In real world dynamic web applications providing various services to users, unluckily, web applications have been contain many security susceptibilities, due to a grouping of unsafe improvement tools and a historic lack of security awareness among programmers. In adding, the risks are magnified when vulnerable software is deployed in the context of the Web, since applications are typically widely accessible and often have access to sensitive contents [1]. These factors have naturally resulted in web-related vulnerabilities receiving substantial attention. The occurrence of data cracks or online crime and other crimes resulting from the exploitation of web application vulnerabilities continues to rise. It is essential to protect applications and systems connected to the Internet against attacks.

Due to their universal use for personal or corporate data, web services always of attacks. These attacks have recently become more mixed, as attention has shifted from attacking the front end to exploiting liabilities of the web applications in order to corrupt the back end database system. An excess of Intrusion Detection Systems (IDS) currently examine network packets individually within both the webserver and the DBMS [2]. Though, there is little work being achieved on multitier intrusion detection systems that generate models of network behavior of both web and database network connections. In such multitier, the database server is often protected firewall while the webservers are remotely accessible over the web. Unfortunately, yet they are threatened from direct remote attacks, and DBMS systems are vulnerable to attacks that use web requests as a means to exploit the back end.

For illustration, observed that the queries will vary based on the value of the constraints approved in the HTTP requirements and the previous state. At times, the same application original functionality can be triggered by many web pages. So, the ensuing mapping between web and database can range from one to many, contingent on the value of the constraints passed in the web request.

2. RELATED WORK

2.1 Toward Programmed Detection:

Web applications are the common path to make services and data accessible on the web. Unluckily, with the increase in the number and complication of these requests, it has also been an increase in the number and difficulty of exposure. Existing techniques to identify security problems in webapplications have mostly focused on input validation faults, such as XSS or SQL injection attack [3]; much less attention dedicated to application logic attacks. Application logic vulnerability is a crucial class of faults that are the result of faulty application logic. These vulnerabilities are definite to the functionality of specific web

technology.

2.2 Client-Site XSS Filters:

Cross Site Scripting faults have now exceeded buffer as the world most common reported security vulnerability. Recent, browser developers and researchers require tried developing client side sifts to moderate these attacks. Our studies focus planned best existing filters and find them to be either inadmissibly slow or easily sidestepped. Some of these filters could introduce vulnerabilities into sites that were error free. The proposed work is new filter design that achieves both high presentation and high correctness by blocking scripts after HTML parsing but earlier destroying. Compared to existing approaches, our approach is faster, defends against more vulnerability.

A number of client-side XSS filters attempt to moderate XSS vulnerabilities by preventing the attacker's script from leaking complex data to the attacker's servers [4]. Normally, filters monitor the movement of data within the JavaScript environment and aim to block the attacker from exfiltration that information to server systems. The real technical difficulty by means of preventing attacks is that web sites frequently export data to third party web sites. For instance, web site that holds a hyperlink to another site drips some amount of data to that site, modern web sites frequently have gorgeous connections with other web sites (POST Message, OAuth Protocol) [5]. To differentiate between benign, malicious information leaks, the XSS filters regularly employ refined investigation techniques, including fault tracking analysis, by means of the associated false negatives and false positives.

2.3 Effective Anomaly Detection:

Learning based anomaly detection has confirmed to be an effective black box technique for detecting attacks. However, the effectiveness of this technique importantly rest on upon both the quality and the completeness of the training data. In most cases, the traffic to the system protected by an anomaly detector is not evenly distributed. Hence, some works (e.g., payments, authentication, and publishing) potency not is trained enough to an anomaly detection scheme in a sensible time intervals. This type of particular importance in real settings, wherever anomaly detection systems are organized with manual configuration, they are expected to automatically learn the normal behavior of a system to detect or block attacks. In this work, first demonstrate that the structures utilized learning based detector can be semantically assembled, and that features of the same group tend to persuade like models. Our method runs our tests on a real world data set casing over HTTP requests to more than 1000 web application mechanisms. The outcome shows that by using the proposed system, it is possible to achieve actual detection even with occasional training data.

2.4 Intrusion Detection System:

A network Intrusion Detection System can be confidential into two types, anomaly detection and hacking detection. Anomaly is first requires the IDS to define and describe the correct and satisfactory static form and dynamic conduct of the scheme, which can be used for detecting abnormal changes or uncommon concerts. The boundary between suitable and anomalous forms of stored code and data is exactly definable.

IDS, also uses time-based material to detect intrusions however, it does not associate actions on a time basis, which runs the risk of incorrectly seeing liberated but simultaneous actions as connected events [6]. Twin protection doesn‟t have such a restriction as it uses the container ID for each session to causally map the associated actions.

2.5 Easy-to-Use Desktop Application:

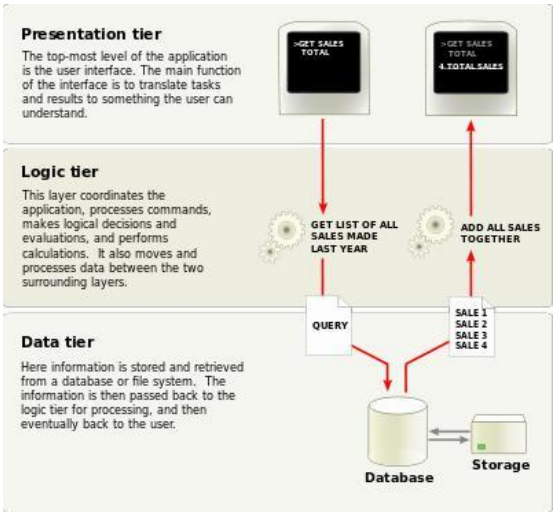
Desktop computers are often cooperated by the communication of untrusted software. To address this problem, this paper presents the technique called Apiary, a system that plainly contains application faults while retaining the usage metaphors of an out dated desktop situation. Apiary completes with three key contrivances [7]. It separates submissions in web containers that incorporate in a controlled way at the display and file system. It introduces container that are quickly instantiated for single request accomplishment, to avoid any activity that ensues from continuing and to guard user privacy. It declares the Layered File System to sort instantiating containers fast and universe efficient [8], and to make dealing many containers no

more composite than a single out dated desktop.

3. SYSTEM ARCHITECTURE

In Software development, multitier architecture often mentioned to as N tiered architecture is a client server architecture in which the demonstration, the application dispensation, the data management are logically separate processes. Fig.1 shows our complete architecture of proposed system. For example, an application that practices middleware to service data requests between a user and a database employs multitier architecture. The most common use of multi-tier architecture is the MVC architecture.

For example, a change of OS in the *presentation tier* would only affect the user interface code. Normally, the user interface turns on a desktop PC or workstation and uses a standard GUI, functional logic may contain of one or more separate components successively on a workplace or application servers and RDBMS on a database server or workstation that clutches the computer storage logic. The intermediate tier strength be multitier itself.



3.1 Multitier Web Applications:

N-tier application architecture provides a model for developers to create a flexible and reusable presentation [9]. By breaking up an application into layers (tiers), developers simply have to change or add a detailed layer, relatively than require rewriting the complete submission over. There must be an exhibition tier, a commercial or data admission tier.

The concepts of layer and tier are often used interchangeably. Yet, one fairly common point of view is that there is indeed a difference and that a cover is a rational structuring utilization for the landscapes that make up the software clarification, although a tier is a corporal constructing appliance for the system substructure.

Three-tier is a client–server architecture in which the user interface, functional process logic ("business rules"), data access and maintained as self-regulating modules, most normally on separate platforms [9]. Apart from the usual advantages of linked software with well-defined boundaries, the three-tier design is planned to permit any of the three tiers to be promoted or substituted autonomously in response to changes in requirements or technology.

3.2 Presentation tier:

This is the topmost level of the application. The presentation tier displays material related to such services as browsing merchandise, purchasing, and shopping cart insides.

It communicates with other tiers by output results to the client tier and all other tiers in the system.

3.3 Application tier:

The logic tier is dragged out since, the presentation tier has own layer; it controls an application functionality by comprehensive handling.

3.4 Data tier:

This tier contains of database servers. Primary task is to storage and retrieval. This layer keeps data neutral and independent from application servers or business logic. Providing data on its own tier also recovers scalability and performance.

4. SYSTEM MODULES

We primarily set up our threat model to include our expectations and the types of occurrences we are aiming to protect alongside. We take on that both the web and the database servers are susceptible. Attacks are network allowed and come from the clients; they launch application layer attacks to cooperation the webservers they are involving to. The attackers can

bypass the webserver to straight attack the database server [10]. We assume that the attacks can neither be detected nor prevented by the current webserver IDS, that attacker may take over the web server after the attack, and that subsequently they can obtain full control of the webserver to launch subsequent attacks. For example, the attackers could modify the application logic of the web applications, snoop or hijack other users‟ web requests and intercept the database queries to snip sensitive data elsewhere their privileges.

4.1 Session Monitoring:

In our prototype, we chose to assign each user session into a different container; however, this was a design decision. For instance, we can assign a new session per each new IP address of the client [11]. In our enactment, sessions were salvaged based on events or when sessions time out. We were able to use the same session tracking mechanisms as implemented by the Apache server because lightweight virtualization containers do not impose high memory and storing overhead. We could preserve a large number of parallel consecutively instances similar to the threads that the server would preserve in the scenario without session containers. If session time out, the instance was concluded along with its container. In our prototype system, we used a one minute timeout due to resource restraints of our test server.

4.2 Analysis of Dataset:

Based on the web server and application logic, diverse inputs would cause different database queries. For instance, to keep a remark to a blog critique, the webserver would first query the database to see the existing comments.

If the user comment differs from previous remarks, then the webserver would mechanically generate a set of new queries to insert the new post into the database. Or else, webserver will reject the input in order to prevent duplicated comments from being posted (i.e., no corresponding SQL query would be issued). In such cases, even assigning the same parameter values would cause different set of queries, depending on the previous state of the website.

Likewise, this nondeterministic mapping case (i.e., one-to-many mapping) happens even after we normalize all parameter values to extract the structures of the web requests and queries. Since the mapping can appear differently in different cases, it becomes difficult to identify all of the one-to-many mapping patterns for each web request. Moreover, when different jobs occasionally overlap at their possible query set, it becomes even tougher for us to abstract the one-to-many mapping for each operation by comparing matched requests and queries across the sessions.

Since the mapping can appear differently in different cases, it becomes difficult to identify all of the one-to-many mapping patterns for each web request [12]. Moreover, when altered operations occasionally overlap at their possible query, it develops even harder for us to extract the mapping for each operation by comparing matched requests and queries across the sessions. Since the algorithm for extracting mapping patterns in static pages no longer worked for the dynamic web, proposed work also created additional preparation technique to build the classical. First, we strained to classify all of the atomic operations on the webpages. All of the operations that appear within one session are permutations of these operations.

4.3 Attack Detection:

The attacker visits the website as a normal user aiming to concession the webserver process or exploit vulnerabilities to bypass authentication. At that point, the attacker issues a set of privileged DB queries to retrieve sensitive data [13]. We capture all the session information and process both legitimate web requests and database queries in the session, here are no mappings amongst them. The twin protection separates the traffic by meetings. If it is a user session, then the requests and queries should all belong to normal users and match physically. By means of the mapping prototypical that we created during the phase, our scheme can capture the unmatched suitcases. And we established the mapping between the HTTP requests and database query, visibly defining which requests should trigger which queries. For an SQL injection attack to be positive, it must change the structure of the query, which our method can willingly perceive. First of all, permitting to our plotting model, Database queries will not have any matching web requests during this type of attack. On the other hand, as this traffic will

not go through any containers, it will be halted as it completes to differ from the genuine traffic that goes through the ampules. The twin protection is intended to mitigate DOS attacks. These attacks can occur in the server architecture without the DBMS.

Attack Scenarios: - Our system is effective at capturing the following types of attacks:

- Privilege Escalation Attack
- Hijack Future Session Attack
- Injection Attack
- Direct DB Attack

4.4 Privilege Escalation Attack:

```
<% InetAddress is=InetAddress.getLocalHost ();
        session.setAttribute ("ip", is);

ip=session1.getAttribute("ipadd").toString();
     session1.setAttribute("ipaddr",ip);
     java.util.Random rnd=new java.util.Random();

          java.util.Random          rnd1=new
java.util.Random();

<meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">

<s: iterator value="list" status="username">
<s: property value="%{ username}"/>

<s: property value="%{ password}"/>
<s: a action="ddos">View DOS 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 legitimate requirements and queries. Our approach, can notice this type of occurrence since the DB query.

4.5 Hijack Future Session Attack:

This class of occurrences is generally expected at the web server side. An attacker usually takes over the webserver and therefore hijacks all successive legitimate user sessions to launch attacks. For instance, by hijacking other user sessions, the attacker can eavesdrop, send spoofed replies, and/or drop user requests.

```
public pojoclass(String type,String time) {
     this.type = type;

     this.time=time;
   }
public void setTime(String time) {
     this.time = time;
   }
```

A session-hijacking attack can be further categorized as a Spoofing an Exfiltration Attack, a DOS and Packet Drop attack, or a Replay attack. According to the mapping model, the web request should invoke some database queries (e.g., a Deterministic Mapping then the abnormal situation can be detected. However, neither a predictable webserver IDS nor a record IDS can detect such an attack by itself. Fortunately, the isolation property of our container based webserver architecture can also prevent this type of attack. As each user‟s web requests are insulated into a separate container, an attacker can never halt into other users‟ session.

4.6 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. Our approach provides two tier detection, even if the activities are recognized by the webserver, the transmitted insides to the DB server would not be able to take on the anticipated structure for the given webserver application.

```
/*Table structure for table `attack`
*/ DROP TABLE IF EXISTS
`attack`; CREATE TABLE `attack` (
  `type` varchar(20) default NULL,
  `time` varchar(50) default NULL )
    ENGINE=InnoDB DEFAULT
```

CHARSET=latin1;

*/*Data for the table `attack` */*

For instance, since the SQL injection attack modifications the structure of the SQL queries, even if the injected data were to go concluded the webserver side, it would generate SQL queries in a different composition that could be perceived as a deviation from the SQL query structure that would typically monitor such a web request.

4.7 Direct DB Attack:

It is likely for an attacker to bypass the webserver or firewalls and join directly to the database. An attacker could also have already taken over the webserver and be acquiescing such inquiries from the webserver without transfer requests. Deprived of matched web requests for these queries, a webserver IDS could perceive neither. Furthermore, if these DB queries were inside the set of allowed queries, then the catalogue IDS it would not distinguish it either.

Though, this type of occurrence can be fixed with our method since we are not matched any web requests with these database queries.

```
/* SQLyog Ultimate v9.02 MySQL-5.0.41-
community-nt:Database- viewdb
----------------------------------------------------------
*/
/*!40101 SET NAMES utf8 */;
/*!40101 SET SQL_MODE=''*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/`viewdb` /*!40100 DEFAULT CHARACTER SET latin1 */;
USE `viewdb`;
/*Table structure for table `login` */
DROP TABLE IF EXISTS `login`;
CREATE TABLE `login` (
`sno` int(10) NOT NULL auto_increment,
`username` varchar(150) default NULL,
`password` varchar(150) default NULL,
PRIMARY KEY (`sno`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*Data for the table `login` */
```

5. RESULT

In our proposed work we applied twin protection for web application. Proposed work has been implemented in by J2EE technology with Apache Tomcat web server and back end as a MySQL Server. Our implementation tested and works efficiently with various attacks. It prevents and fights against the SQL injection attack and Session attack.

6. CONCLUSION

We existing an intrusion detection system that figures models of normal performance for multitier web claims. Container based IDS with multiple input streams to produce alerts. We have shown that such correlation of input torrents offers a better classification of the system for anomaly detection.

REFERENCES

[1] D. Bates, A. Barth, "Regular Expressions Considered Harmful in Client-Side XSS Filters," Proc. 19th Int‟l Conf. World Wide Web, 2010.

[2] M. Christodorescu, S. Jha, „Static Analysis of Executables to Detect Malicious Patterns,‟ Conf. USENIX Security Symp. 03.

[3] M. Cova, D. Balzarotti, V. Felmetsger, "Swaddler: An Approach for the Anomaly-Based Detection of State Violations in Web Applications," Proc. Recent Advances in Intrusion Detection (RAID ‟07).

[4] H. Debar, M. Dacier, A. Wespi, "Towards a Taxonomy of Intrusion-Detection Systems," Computer Networks, vol. 31, no. 9, 99.

[5] V. Felmetsger, L. Cavedon, C. Kruegel, "Toward Automated Detection of Logic Vulnerabilities in Web Applications" USENIX Security Symp.,‟10.

[6] Y. Hu, B. Panda, "A Data Mining Approach for Database Intrusion Detection," Proc. Applied Computing (SAC), H. Haddad, A. Omicini, R.L. Wainwright, and L.M. Liebrock,04. \

[7] Y. Huang, A. Stavrou, A.K. Ghosh, S. Jajodia, „Efficiently Tracking Application Interactions Using Lightweight Virtualization,‟ First ACM Workshop Virtual Machine Security,08.

[8] H.-A. Kim and B. Karp, "Autograph: Toward Automated Distributed Worm Signature Detection," Proc. USENIX Security Symp., 2004.

[9] C. Kruegel, G. Vigna, "Anomaly Detection of Web-Based Attacks," 10th Conf. Computer and Comm. Security (CCS ‟03), 2003.

[10] S.Y. Lee, W.L. Low, P.Y. Wong, "Learning Fingerprints for a Database Intrusion Detection System, Research in Computer Security, 2002.

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

[12] J. Newsome, B. Karp, and D.X. Song, "Polygraph: Automatically Generating Signatures for Polymorphic Worms," Proc. IEEE Symp. Security and Privacy, 2005.

[13] B. Parno, J.M. McCune, D. Wendlandt, D.G. Andersen, A. Perrig, „CLAMP: Practical Prevention of Large-Scale Data Leaks" IEEE Symp. Security and Privacy - 09.

[14] A. Schulman, "Top 10 Database Attacks," http://www.bcs.org/server.php?show=ConWebDoc.8 852 - 2011.

[15] R. Sekar, „An Efficient Black-Box Technique for Defeating Web Application Attacks," Network and Distributed System - 2009.

[16] A. Seleznyov,S. Puuronen, "Anomaly Intrusion Detection Systems- Handling Temporal Relations between Events," Int"l Recent Advances in Intrusion Detection,1999.

[17] Y. Shin, L. Williams, T. Xie, "SQLUnitgen: Test Case Generation for SQL Injection Detection," Dept. of Computer Science, 2006.

[18] A. Srivastava, S. Sural, A.K. Majumdar, „Database Intrusion Detection Using Weighted Sequence Mining," Computers, no. 4, pp. 8-17, 2006.