International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

INTRUSION DETECTION IN MULTITIER WEB APPLICATIONS

Shenbagalakshmi Gunasekaran[1], K.Muneeswaran[2]
[1,2]*Department of Computer Science and Engineering, Mepco Schlenk Engineering College, TamilNadu, India.*
Email: pri.infotech@gmail.com[1], kmuni@mepcoeng.ac.in[2]

Abstract

Intrusion Detection Systems endeavor at detecting attacks against computer systems and networks that offer techniques for modeling and distinguish normal and abusive system behavior. Web Applications are widely used for critical services and sophistication of attacks against these applications has grown as well. In this paper, we designed IDS in multitier architecture that models the network behavior of user sessions across both the front-end web server and the back-end database. The system identifies the attacks like SQL injection attack by monitoring both the web and database request which an independent system cannot do. It also handles the relations between the actions caused by the user for every simple user sessions and is also designed to detect potential violations in database security. Our detection approach belongs to anomaly detection, and we depend on a training phase to build the correct model. The system is implemented using Apache web server with MySQL and lightweight virtualization. For each client a "WebServer Virtual Machine" is created and is associated with an independent holder ID and hence it enhances the security. The concept of holder and the user behavior pattern provides a means of tracking the information flow from the web server to the database server for each session.

***Keywords*-- Multitier Web Application, Anomaly Detection, Virtualization, Web-Based Attacks.**

I. INTRODUCTION

WEB-DELIVERED services and web-based applications have become tremendously popular, and nowadays they are routinely used in security-critical environments, such as travel, medical, military systems and social networking. Web servers are usually accessible through the corporate firewalls, and web-based applications are often developed without following the sound security methodology. Attacks that exploit web servers or server extensions represent a substantial portion of the total number of vulnerabilities. The attention of attacks has shifted from attacking the front end to exploit vulnerabilities of the web applications [3], [6], [1] inorder to corrupt the back-end database. The back-end database server is often protected behind the firewall, though they are protected from direct remote access, the back-end systems are susceptible to attacks that use web requests as a means to exploit the back-end database. To protect the multitired web services, Intrusion detection systems have been widely used to detect known attacks by matching misused traffic patterns or signatures [5], [12], [15], [4]. Individually, the web IDS and the database IDS can detect abnormal network traffic sent to either of them. These IDSs cannot detect cases where the normal traffic is used to attack the webserver and the database server.

In this paper, we present IDS which is used to detect attacks in multitired web services, by employing an isolated Virtual Computing Environment (VCE) and User Behavior.

In our approach, we created the routine models of isolated user sessions which include both the front-end (HTTP) and back-end (SQL) network transactions. A light weight virtualization technique is used to assign each user's web session to the dedicated holder, which is an isolated virtual computing environment. We use the HolderID (HID) and the SessionID (SID) to accurately associate the web request with the subsequent DB queries. By considering the webserver and database traffic into account we are capable of building a mapping pattern. The pattern helps to provide a segregation that prevents the future session hijacking attacks. Many copies of the webserver instances are executed in different holders which help to isolate from the rest. As each client session is assigned to a dedicated holder, the damage is restricted to the compromised session; other sessions remain unaffected by it.

It also includes the representation of the User's behavior that is used by the data mining tools to construct behavior patterns. These are used to decide whether current behavior follows a certain normal pattern or differs from all known User's behavior patterns. The mapping patterns are built based on the working characteristics with the blog or websites. Static websites does not permit content modification from users and some web services permit persistent back-end data modifications, which is the dynamic websites. This allows HTTP requests to include parameters that are variable and it depends on the user input.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

Each user's interaction with the blog consists of different activities that he performs in order to achieve his goals. The user profiles are built basing on different characteristics, such as reading an article, post an article, make comment to article, list articles by categories, visit next page, read RSS feed and these are analyzed using the log files. Sometimes each user performs similar activities which are expressed by repeated sets of actions and which differ on the per-user basis. This gives the possibility to differentiate an intruder from a valid user.

II. Related Work

Anomaly detection relies on models of the intended behavior of users and applications and interprets deviations from this 'normal' behavior as evidence of malicious activities [8], [9], [7], [2]. These models can focus on the users, the applications, or the network. Behavior profiles are built by performing a statistical analysis on historical data [11], [16] or by using rule-based approaches to specify user behavior patterns [18]. Network-based intrusion detection is vulnerable to insertion and evasion attacks [21]. These attacks attempt to desynchronize the view of the intrusion detection system (IDS) with respect to the view of the actual target, that is, web server. In addition, intrusion detection systems do not take into account the application-level logic of the web server, and, therefore, they cannot identify attacks that exploit the organization and configuration of the server application.

Intrusion alert correlation [17] provides a collection of components that that transforms intrusion detection sensor alerts into concise intrusion reports in order to reduce the number of false positives, pretend alerts, and non-relevant positives. It focuses primarily on abstracting the lower-level sensor alerts and providing higher-level events to the users. Our IDS system correlates alerts from different IDSs and operates on multiple feeds on network traffic using a single IDS that looks across sessions to produce an alert without correlating alerts produced by other independent IDSs.

In addition, Database security is also important, since the database stores valuable information of an application. Some data mining techniques have been used for network intrusion detection by considering classification, link analysis and sequential analysis [19] as potential data mining algorithms along with their along with their applicability in the field of intrusion detection. In our approach, we utilized the HolderID to separate session traffic as a way of extracting and identifying relationships between web server requests and database query events.

Validating input is useful to detect or prevent SQL or Cross Site Scripting (XSS) injection attacks [23], [13]. Our approach can detect SQL injection attacks by taking the structures of web requests and database queries without considering the values of the input parameters.

III. Problem Definition

3.1 Goals

Our primary goal is to prevent a web server and database server compromise from the intruders. We achieve this goal by ensuring that sensitive data in the database is released to code running on behalf of the user who has authenticated successfully and has legitimate access to the data. We aim to enable these strong data protection for web applications while at the same time we minimize the false positive rate.

3.2 Assumptions

For the threat model to be built we assume that both the web and the database servers are vulnerable. Attacks are network-borne and come from the web clients, who launch application-layer attacks. The attackers can evade the web server to directly attack the database server. We assume that the attacks can neither be detected nor prevented and the intruder can take full control over the web server after the attack.

Database servers are not exposed to the public and hence we assume that, it is difficult for the attacker to takeover completely. No prior knowledge of the source code or the application logic of web services deployed on the web server. Traffic analysis is done at both the web server and the database server. No attack occurs at the working out phase and model building.

IV. Architecture And Internment

In our design, we make use of the lightweight process referred as "holders", which are disposable servers for client sessions. It is possible to run thousands of holders on a single physical host and these virtualized holders can be reinitialized to serve new sessions. Our approach dynamically generates new holders and it recycles the used ones. Each session is assigned to a dedicated WebServer Virtual Machine and is isolated from other sessions. Single user always deals with the same WebServer Virtual Machine.

4.1 Building the Normal Model

The Holder-based and the Session-separated architecture enhance the security and also provide us with the isolated information flow that is separated in each holder session.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

It also helps to identify the contributory pairs of web requests and the resulting SQL queries in the given session. We deployed sensors at both the ends in the host system, that is, web server front-end and the database server back-end. These cannot be attacked straightforwardly by the attacker since only the virtualized holders are bare to the attackers. Our sensors will not be attacked at the database server too, as we assume that the attacker cannot completely take control of the database server

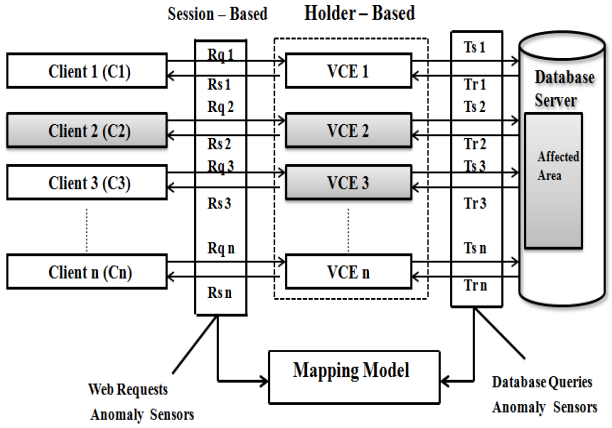

Figure 1: Web server instances running in holders

The above described approach illustrates how the communications are categorized as sessions and how database transactions can be related to the corresponding session. Here, the client 2 will compromise only the VCE 2, and the corresponding database transaction T_2 will be the affected section of the data within the database. Here,

R_q: Request made by the client to the "Web Server VirtualMachine"
R_s: Reply from the VCE to the client.
VCE: Virtual Computing Environment.
T_s: Database Transaction set request.
T_r: Reply made by the database server with the corresponding set of queries.

In fact, we assume our sensors cannot be attacked and we always incarcerate correct traffic information at both the ends. Moreover, as traffic can be easily being separated by session, it is possible and easy for us to compare and analyze the request and queries across different sessions. Once we build the mapping model, it can be thus 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 arises any request or query violating the normal model within the session, then the corresponding session is well thought-out to be an attack.

4.2 Attack Scenarios

Our approach is effectual at capturing the following type of attacks.

4.2.1 SQL Injection Attack

Attacks such as the SQL injection do not require compromising the web server. Attackers use the existing vulnerabilities in web server logic to inject the data content that contains the exploits and then use the web server to relay these exploits to attack the back-end database. Our approach provides a two-tier detection, so the relayed contents to the database server would not be able to take on the expected structure for the given web server request.

4.2.2 Privilege Escalation Attack

The attack takes advantage of the program error or design flaws to grant attacker elevated access to the network. For an administrator, the request R_a will generate a set of admin level queries Q_a and similar for the other regular users. Our approach, can detect this type of attack since the DB query of the administrator does not match with the request made by the normal user, according to the mapping pattern.

4.2.3 Hijack Future Session Attack

The attack is mainly aimed at the web server side. An attacker usually takes over the web server and therefore hijacks all consequent legitimate user sessions to launch attacks. Fortunately, the isolation property of our Holder-based web server architecture helps to prevent this type attack. As each user's web requests are isolated into a separate holder, an attacker can never break into other user's session.

V. MODELING MAPPING PATTERNS

Due to the assorted functionality, different web applications exhibit different characteristics. Some websites allow regular users with the non- administrative privileges to update the contents of the server data. This creates challenge for IDS system because the HTTP requests can contain variables in the passed parameters. Our approach normalizes the variable values in both HTTP requests and database queries, preserving the structures of the requests and queries. Following this step, session i will have a set of requests (R_i), as well as a set of queries (Q_i). If the total number of sessions of the training phase is N, then we have the set of total web requests (REQ) and the set of total SQL queries (SQL) across all sessions. Each single web request $r_m \in$ REQ may also appear several times in different R_i where i = 1, 2 . . . N. The same holds true for $q_n \in$ SQL.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

We classify the four possible mapping patterns [21]. Since the request is at the origin of the data flow, we treat each request as the mapping source. The mappings in the model are always in the form of one request to a query set r_m Q_n. The possible mapping patterns are Deterministic Mapping, Empty Query Set, No Matched Request, and Nondeterministic Mapping.

VI. Modeling For Static And Dynamic Websites

In the case of a static website, the nondeterministic mapping does not exist as there are no available input variables or states for static content. We can easily classify the traffic collected by sensors into three patterns in order to build the mapping model. As the traffic is already separated by session, we begin by iterating all of the sessions from 1 to N. For each $r_m \in REQ$, we maintain a set AR_m to record the IDs of sessions in which r_m appears. The same holds for the database queries.

We search for the AQ_s that equals the AR_m . When $AR_m = AQ_s$, this indicates that every time r_m appears in a session, then q_s will also appear in the same session, and vice versa. Some web requests that could appear separately are still present as a unit.

In contrast to static webpages, dynamic webpages allow users to generate the same web query with different parameters. Additionally, dynamic pages often use POST rather than GET methods to commit user inputs. Based on the webserver's application logic, different inputs would cause different database queries. By placing each r_m , or the set of related requests R_m , in different sessions with many different possible inputs, we obtain as many candidate query sets { Q_n , Q_p , Q_q . . .} as possible. This mapping model includes both deterministic and nondeterministic mappings, and the set EQS is still used to hold static file requests.

VII. Performance Evaluation

The implementation of our prototype involves the web server and the back-end DB. We also used two testing websites, static and dynamic. We analyzed three classes of attacks and measured the false positive rate for each of the two websites. Finally we compared the user behavior for each of the session for a different set of users. The following represents the implementation of our prototype and the attack detection rates.

7.1 Prototype Implementation

In our design, we choose to assign every user session into a different holder which was the security design decision. Each and every new client (IP address) is assigned to a new holder and these holders are cast-off or recycled based on the session time out.

The session time out is considered to be 30-minute. Thus, we are capable of running multiple instances in a single server.

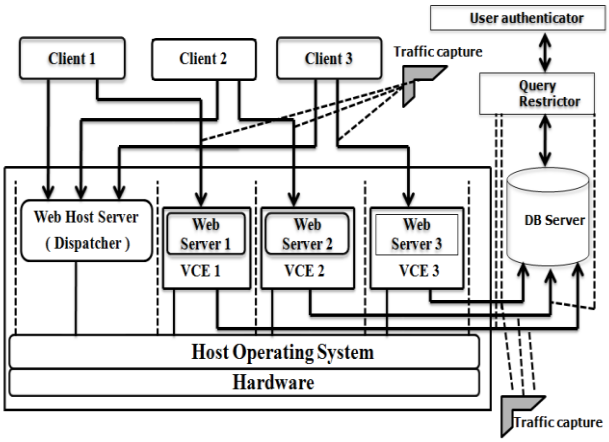

Figure 2: Architecture of the IDS

The above figure depicts the architecture and session management of our prototype, where the host web server acts as the dispatcher. In the case of the static website, we served 15 unique web pages and collected real traffic to this website and obtained 350 user sessions. In the case of the dynamic websites, the site visitors are allowed to read, post and comment on articles.

Query Restrictor: The Query Restrictor (QR) is a trusted module that restricts "Web Server VirtualMachine" access to sensitive database content. In our implementation, the QR is a specialized SQL proxy that interposes on all databases traffic without requiring changes to the Web Server. When Web Server VirtualMachine acting on behalf of a user attempts to connect to the database, the QR instead connects the client to a separate *restricted database* tailored specifically to that user**.** Prevents one user from retrieving another user's data from the database.

User Authenticator: It binds the executing server code to the user's identity. Because the Web Server VirtualMachine is entrusted, our approach provides UA, to authenticate users. For instance, the UA may check that the supplied password matches the user's entry in the database, which the UA accesses via the QR.

7.2 Static Website Model in Training Phase

For the static website, Deterministic Mapping and the Empty Query Set Mapping patterns appear in the training sessions. We first collected 150 real user sessions for a training data set before making the website public so that there was no attack during the training phase.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

We used part of the sessions to train the model and all the remaining sessions to test it. For each number on the x-axis of Fig. 3, we randomly picked the number of sessions from the overall training sessions to build the model using the algorithm, and we used the built model to test the remaining sessions. We repeated each number 10 times and obtained the average false positive rate (since there was no attack in the training data set).

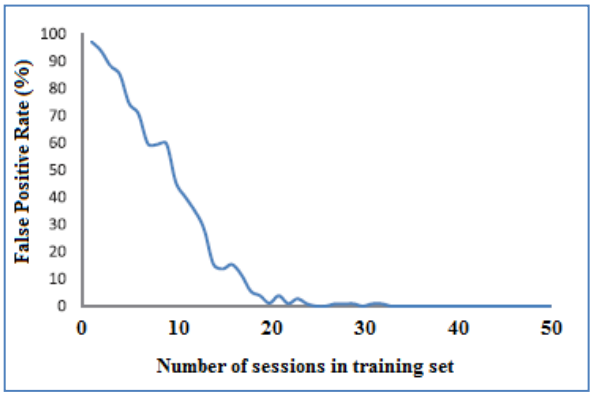

Figure 3: False positives verses training time for static websites

Fig. 3 shows the training process. As the number of sessions used to build the model increased, the false positive rate decreased (i.e., the model became more accurate). From the same figure, we can observe that after taking 35 sessions, the false positive rate decreased and stayed at 0. This implies that for our testing static website, 35 sessions for training would be sufficient to correctly build the entire model. Based on this training process accuracy graph, we can determine a proper time to stop the training.

7.3 Dynamic Model Detection Rates

We also conducted model building experiments for the dynamic blog website. We obtained 215 real user traffic sessions from the blog under daily workloads. During this phase, we made our website available only to internal users to ensure that no attacks would occur. We then generated 10 attack traffic sessions mixed with the normal legitimate user session, hence the mixed traffic is used for the attack detection.

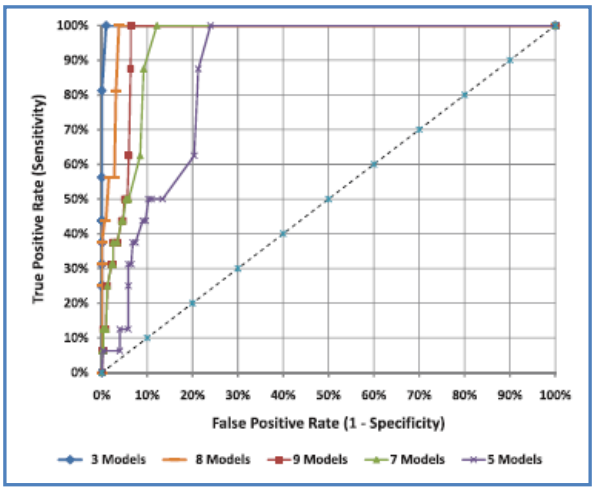

Figure 4: False positive rate for dynamic models

The above figure shows the ROC curves for the testing result. We built our models with different number of operations, and each point on the curves indicates the threshold value. The threshold value is defined as the number of HTTP requests or SQL queries in a session that are not matched with the normality model. The nature of the false positives comes from the fact that our manually extracted basic operations are not sufficient to cover all legitimate user behaviors.

VIII. CONCLUSION

We presented an intrusion detection system that builds models of normal behavior for multitired web applications from both the front-end web (HTTP) requests and back-end database (SQL) queries. Correlation of the input streams provides a better characterization of the system for anomaly detection because the intrusion sensor has a precise normality model that detects a wide range of attacks. We achieved this by isolating the information flow from each web server session with a virtualization technique. For static websites, we built a model which proved to be effective at detecting different types of attacks. Hence, we are able to identify a wide range of attacks with minimal false positives. The False positive rates for the static and dynamic websites are 0 and 0.7 respectively.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

REFERENCES

[1] "Five Common Web Application Vulnerabilities", http://www.symantec.com/connect/articles/five-common-web-application-vulnerabilities, 2011.

[2] C. Ko, M. Ruschitzka, and K. Levitt, "Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-based Approach." In Proceedings of the 1997 IEEE Symposium on Security and Privacy, pages 175to187, May 1997.

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

[4] B.I.ABarry 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.

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

[6] SANS, "The Top Cyber Security Risks," http://www.sans.org/top-cyber-security-risks/,2011

[7] A.K. Ghosh, J. Wanken, and F. Charron. , "Detecting Anomalous and Unknown Intrusions Against Programs". In Proceedings of the Annual Computer Security Applications Conference (ACSAC'98), pages 259to267, Scottsdale, AZ, December 1998.

[8] D.E.Denning.,"An Intrusion Detection Model". IEEE Transactions on Software Engineering, 13(2):222to232,February 1987.

[9] T. Lane and C.E. Brodley. "Temporal sequence learning and data reduction for anomaly detection". InProceedings of the 5th ACM conference on Computer and communications security, pages 150 to158. ACM Press, 1998.

[10] A. Schulman, "Top 10 Database Attacks," http://www.bcs.org/server.php?show=ConWebDoc.8852, 2011.

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

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

[13] T. Pietraszek and C.V. Berghe, "Defending against Injection Attacks through Context-Sensitive String Evaluation," Proc. Int'lSymp. Recent Advances in Intrusion Detection (RAID '05), 2005.

[14] G.E. Suh, J.W. Lee, D. Zhang, and S. Devadas, "Secure Program Execution via Dynamic Information Flow Tracking," ACM SIGPLAN Notices, vol. 39, no. 11, pp. 85-96, Nov. 2004.

[15] T.H. Ptacek and T.N. Newsham. "Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection".Technical report, Secure Networks, January 1998.

[16] G. Vigna, W.K. Robertson, V. Kher, and R.A. Kemmerer, "A Stateful Intrusion Detection System for World-Wide Web Servers,"Proc. Ann. Computer Security Applications Conf. (ACSAC '03),2003.

[17] A. Seleznyov and S. Puuronen, "Anomaly Intrusion Detection Systems: Handling Temporal Relations between Events," Proc Int'l Symp. Recent Advances in Intrusion Detection (RAID '99), 1999.

[18] M. Roesch,"Snort, Intrusion Detection system," www.snort.org/2011.

[19] W. Lee, S.J. Stolfo, "Data Mining Approaches for Intrusion Detection", Proceedings of the USENIX Security Symposium, pp. 79-94 (1998).

[20] Liang and Sekar, "Fast and Automated Generation of Attack Signatures: A Basis for Building self-Protecting Servers," SIGSAC:Proc. 12th ACM Conf. Computer and Comm. Security, 2005.

[21] Meixing Le, AngelosStavrou, BrentByungHoon Kang," DoubleGuard: Detecting Intrusions in Multitier Web Applications", IEEE transactions on dependable and secure computing, vol. 9, no. 4,July/august 2012.

[22] D. Bates, A. Barth, and C. Jackson, "Regular Expressions Considered Harmful in Client-Side XSS Filters," Proc. 19th Int'lConf. World Wide Web, 2010.