Advanced Intrusion Detection with DoubleGuard in Multitier Web Application Environments

Dr. Samuel Asare*[1] & Dr. Nana Kofi Osei[2]

[1](Senior Lecturer, Department of Mechanical Engineering), Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
[2](Lecturer, Department of Mechanical Engineering), Kwame Nkrumah University of Science and Technology, Kumasi, Ghana

ABSTRACT

Network attacks are increased in number and severity over the past few years, intrusion detection system (IDS) is increasingly becoming a critical component to secure the network. Intrusion detection is the process of monitoring and analyzing the events occurring in a computer system in order to detect signs of security problems. Intrusion Detection Systems has the additional job of triggering alarms toward this security problem and some of it automated in the role of triggering or doing an action on behalf of the network administrator. The goal of intrusion detection system (IDS) is to provide another layer of defense against malicious (or unauthorized) uses of computer systems by sensing a misuse or a breach of a security policy and alerting operators to an ongoing attack.

In this paper, we have illustrated difficulties to implement IDS in multitier architecture. Since it is difficult to implement multiple IDS, We have introduced a new protocol-Double Guard. Double Guard, is an IDS system that models the network behavior of user sessions across both the front-end web server and the back-end database. It monitors both the web and database request and identifies the attacks like SQL injection attack which independent IDS cannot do.

Keywords: Intrusion Detection System, Anomaly Detection, Webserver, Attacks, SQLIA, Classification of SQLIA.

I. INTRODUCTION

With the tremendous growth of internet and interconnections among computer systems, networks security is becoming a major challenge. Security is the process of detecting and preventing to your system or/and computer from unauthorized users. Detection helps you to determine whether or not someone attempted to break into your system and if they were successful what they may have done. Whereas prevention measures help you to stop or block unauthorized users, from accessing any part of your system.

There are various software available for security but they lack some degree of intelligence when it comes to observing, recognizing and identifying attack signatures that may be present in traffic or in case if there is a backdoor or hole in the infrastructure and that's where intrusion detection comes in. IDS categorized into mainly in three types on the basis of kind of activities, system, traffic or behaviour they monitor which is host-based, network-based and application-based. Here we are focusing on network intrusion detection 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 behaviour of the system, which can then be used to detect abnormal changes or anomalous behaviours. The boundary between acceptable and anomalous forms of stored code and data is precisely definable. Behaviour models are built by performing a statistical analysis on historical data or by using rule-based approaches to specify behaviour 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.

II. LITERATURE REVIEW AND RELATED WORK

This paper provides a literature review containing an intrusion detection system and network security related topics that helps in research work and provide the needed conceptual framework for the development of the proposed model.

K. Bai, H. Wang, and P. Liu, "Towards Database Firewalls," Proc. Ann. IFIP WG 11.3 Working Conf. Data and Applications Security (DBSec '05), 2005.

Authentication based access control and integrity constraint are the major approaches applied in commercial database systems to guarantee information and data integrity. However, due to operational mistakes, malicious intent of insiders or identity fraud exploited by outsiders, data secured in a database can still be corrupted. Once attacked, database systems using current survivability technologies cannot continue providing satisfactory services according to differentiated information assurance requirements. In this paper, we present the innovative idea of a database firewall, which can not only serve differentiated information assurance requirements in the face of attacks, but also guarantee the availability and the integrity of data objects based on user requirements. Our approach provides a new strategy of integrity-aware data access based on an on-the-fly iterative estimation of the integrity level of data objects. Accordingly, a policy of transaction filtering will be dynamically enforced to significantly slow down damage propagation with minimum availability loss.

B.I.A. Barry and H.A. Chan, "Syntax, and Semantics-Base Signature Database f or Hybrid Intrusion Detection Systems," Security and Comm. Networks, vol. 2, no. 6, pp. 457-475, 2009.

Signature-based intrusion detection systems (IDSs) have the advantages of producing a lower false alarm rate and using less system resources compared to anomaly based systems. However, they are susceptible to obfuscation used by attackers to introduce new variants of the attacks stored in the database. Some of the disadvantages of signature based IDSs can be attributed to the fact that they are mostly purely syntactic and ignore the semantics of the monitored systems. In this paper, we present the design and implementation of a signature database that assists Specification-based IDS in a converged environment. Our design is novel in terms of considering the semantics of the monitored protocols alongside their syntax. Our protocol semantics awareness is based on the state transition analysis technique which models intrusions at a high level using state transition diagrams. The signature database is hierarchically designed to insure a balance between ease of use and fast retrieval in real time. The database prototype is tested against some implemented attacks and shows promising efficiency.

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.

Cross-site scripting flaws have now surpassed buer over-ows as the world's most common publicly-reported security vulnerability. In recent years, browser vendors and re-searchers have tried to develop client-side filters to mitigate these attacks. We analyze the best existing filters them to be either unacceptably slow easily circumvented. Worse, some of these filters could introduce vulnerabilities into sites that were previously bug-free. We propose a new filter design that achieves both high performance and high precision by blocking scripts after HTML parsing but before execution. Compared to previous approaches, our approach is faster, protects against more vulnerabilities, and is harder for attackers to abuse. We have contributed an implementation of our filter design to the WebKit open source rendering engine, and the filter is now enabled by default in the Google Chrome browser

III. EXISTING SYSTEM

In this chapter we will see how the existing system behaves in multitier web application and how the IDS works in multitier web architecture. In the existing system, Intrusion detection systems have been widely used to detect known attacks by matching misused traffic patterns or signatures in a Multitier web services. Intrusion Detection Systems (IDSs) examine network packets individually within both the web server and the database system. Multitier Anomaly Detection (AD) system that generate model of network behavior for both web and database network interactions is not much efficient. In such multitier architectures, the back-end database server is often protected behind a firewall while the web servers 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. Generally, Web IDS and the database IDS can detect abnormal network traffic sent to either of them. But when attack is done with normal traffic it is difficult to find with such system.

For example if an attacker with non-admin privileged is login to a web server using normal access credentials, an attacker can find a way to issue a privileged database query by exploiting the vulnerabilities in the web server. Neither the web IDS nor database IDS would detect this type of attack. Since web IDS see only normal traffic, according to web IDS username and password it will give access to the user as per its privilege level. Database IDS see traffic only at database and as per database IDS if query is correct and does not contain any malicious code it will them data or information for which it is requested. And hence existing system detect this type because no one is going track the information flow after successful login at the web server and up to the database transaction. This type of attack can be detected if the database IDS can identify that the privileged level of request query is not associated or matched with the privileged level of user-access. And for that it is very necessary mapped database query and response with the client request and response. But with the current multithreaded web server architecture it is not possible to detect/profile such casual mapping between web

server traffic and database traffic since traffic cannot be clearly attributed to user sessions.

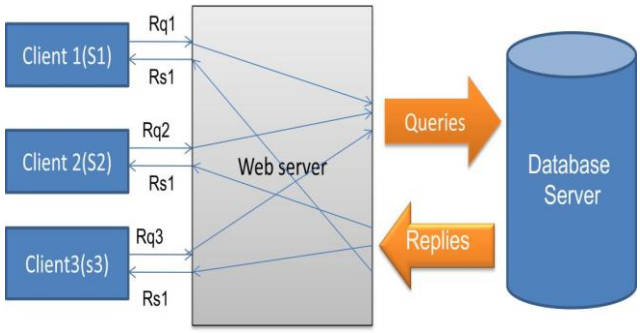


Fig 1: Existing System Architecture in Which Webserver Act as Front End and Database Server as Storage Back End.

As the above figure shows all network traffic from normal user and attacker is received and intermixed at the same web server. At the database side it is unable to tell which transaction corresponds to which client request. As the communication between web server and database are not separated it is difficult to understand the relationships among them. Other approaches have detected intrusions or vulnerabilities by statically analyzing the source code or executables. Other dynamically tracks the information flow to understand taint propagations and detect intrusions.

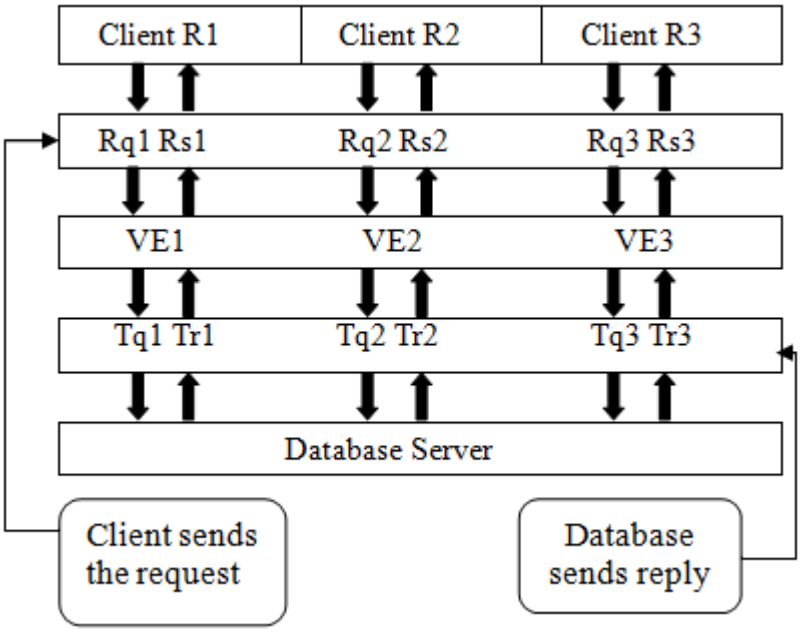
Disadvantages

1. With this existing system, we cannot detect attacks when the traffic is usual. And using multiple IDS to detect anomaly is tedious.
2. The current multithreaded web server architecture, it is not feasible to detect or profile such causal mapping between webs server traffic and DB server traffic since traffic cannot be clearly attributed to user sessions.
3. This system is not efficient to provide security against threats like SQL injection attacks.
4. Other limitation indicates that multitier IDS are not efficient in terms of training sessions and functionality coverage.
5. If an attacker compromises the web server other communication sessions can be hijacked and affect.
6. Difficult to understand the relationships among database server and web server.

IV. PROPOSED SYSTEM

System Architecture

The propose system Double Guard is used to detect attacks in multitier web services. It is used to monitor the network behavior across both the front end webserver and the backend database. For that we are creating normative model of isolated user sessions that include both front end as well as back end network transaction. To achieve this we are using lightweight virtualization technique to assign each users web session to a dedicated container in an isolated virtual computing environment. This new container-based web server architecture enables user to separate the different information flows by each session.



System Architecture Where Webserver Instances Running in Containers and Assigns Each User Sessions to a Dedicated Container.

This provides a means of tracking the information flow from the web server to the database server for each session. This approach also does not require for user to analyze the source code or know the application logic. Double Guard container architecture is based on Open VZ and lightweight virtualization. Virtualization indicates that each client uses its own virtual web server i.e. each client is processed by a different web server. Thus, highly secure system is provided as each client process is taken as separate session.

This system uses lightweight process containers, referred to ephemeral, disposable servers for client sessions. It is possible to initialize thousands of containers on a single physical machine, and these virtualized containers can be discarded, reverted, or quickly reinitialized to serve new sessions. A single physical web server runs many containers, each one an exact copy of the original web server. This approach dynamically generates new containers and recycles used ones. As a result, a single physical server can run continuously and serve all web requests. However, from a logical perspective, each session is assigned to a dedicated web server and isolated from other sessions. A container-ID is used to associate web request with its subsequent DB queries and separate communication at the session level so that single user always deal with the same server which help us to identify suspect behavior by both session and user.

At both the ends we put anomaly sensors whose function is to capture the traffic information from the network. This traffic capture analysis modules are deployed at the host system and cannot be attacked directly since only the virtualized containers are exposed to attackers. Mapping model can be used to detect abnormal behaviors. Both the web requests and the database queries within each session should be in accordance with the model. If there exists any request or query that violates the normality model within a session, then the session will be treated as a possible attack.

V. MODULES DESCRIPTION

Proposed System is implemented by dividing the entire project into the following functional modules:

SQL Attack Module:

In this module, it analyzed the four attacks that generally takes place. These attacks are Hijack Future Session Attack, Privilege Escalation Attack, Injection Attack and Direct DB Attack. In Privilege Escalation Attack, the attacker login as a normal user and triggers admin queries so as to obtain an administrator's data. Hijack Future Session Attack is class of attacks is mainly aimed at the webserver side. An attacker usually takes over the webserver and therefore hijacks all subsequent legitimate user sessions to launch Attacks. SQL injections 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. In a Direct DB attack, an attacker can bypass the webserver or firewalls and connect

directly to the database. Proposed shows how these attacks take place. Initially the attacker passes a query and login. It gets all the data in the database and retrieves it. If the same query he/she types in the backend Sql server, can retrieve all information about the Admin database. So this way, it is shown that how an attack takes place in a system.

Prevention Module:
After the server is activated, each client is initiated to use the service. Each client has its own webserver i.e. multiple virtual webserver is created in a single system using same service. So each client access through a virtual webserver, in this way proposed system can create multiple instance of server. Hence client can access a service through the webserver which indicates the basic concept of Double guard architecture. Once client is initiated, it tries to login to use the service. Here double guard depicts the prevention provided against the attacker. The four attacks has been identified and shown how to overcome it. Here only authorized user can login and use the blog. If an attacker login, he/she is identified and blocked. No further process can be done by them. Due to the use of multiple webserver sometimes attacker get confused about the original server and instance of the server.

Blog Creation Module:
In this module, It shows both Static and dynamic website. Initially the clients logon to his blog. After identifying him as an authenticated user, he can visit the blog. The Home page is a dynamic site as it can be edited and changed. The client can add his profile name or do any changes to his blog. After you click preview, you can see the static site as all contents are static. Changes can not be made in that site. In the blog you can type the content you want to post and do post. It is like all blog pages where user can post his blog. After all work has done the user can logout from the site which guarantee his security.

Traffic Capture Analysis Module:
This module shows the traffic analysis captured between the client and webserver and also between the server and the database. It provides the overall information regarding the total packet sent, length of the packet and time of receiving of packets. It provides details about the destination IP, source IP and captured time of packet. It also gives information about the Ethernet frames, the protocol used like TCP/IP etc and details about HTTP protocol. It also provide graphical display of various OSI layers like Network layer, Application layer etc. That is it provide visual scenario of the usage levels of each layer of OSI layer. Using this intruder can be detected as packet size and its all information is available.

Intrusion Detection Module:
In this module, Intruder is detected and his activities have been noted. Generally, using the information about the capture time of each packet, last sent packet and its length can be identified. So analyzing this overall information an intruder can be detected. Usually an intruder login and then does all his activities. This is stored in the database and can be used to detect the abnormal usage. Subsequent request can be noted and an intruder can be identified. Based on the usage, a Network layer is depicted. It shows the graphical usage pattern.

VI. SYSTEM DEVELOPMENT

In this system we have implemented a prototype**, Double guard** which is used to detect attacks in a multitier architecture. This is container-based web architecture that not only fosters the profiling of causal mapping, but it also provides an isolation that prevents future session-hijacking. This is implemented using light weight virtualization environment that ran many copies of the webserver instances in different containers so that each one was isolated from the rest. Each user's web session is assigned to a dedicated container and an isolated virtual computing environment is created. For websites that do not permit content modification from users, there is a direct causal relationship between the requests received by the front-end webserver and those generated for the database back end.

For modeling a static website, we have used **Static Model Building Algorithm.** This algorithm takes the input of training data set and builds the mapping model for static websites. For each unique HTTP request and database query, the algorithm assigns a hash table entry, the key of the entry is the request or query itself, and the value of the hash entry is AR for the request or AQ for the query, respectively. The algorithm generates the mapping model by considering all three mapping patterns i.e. Deterministic Mapping, Empty Query Set, No Matched Request pattern. This algorithm is implemented in the website traffic analysis of our project.

Static Model
Static website 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 (e.g., blogs, forums) allow regular users with no administrative privileges to update the contents of the server data. This creates tremendous challenges for IDS system training because the HTTP requests can contain variables in the past parameters. For example, instead of one to one mapping, one web request to the web server usually invokes number of SQL queries that can vary depending on type of the request and the state of the system. Some requests will only retrieve data from the web server instead of invoking database queries, meaning that no queries will be generated by these web requests. In other cases, one request will invoke number of database queries. All communications from the clients to the database are separated by a session. Assign each session with a unique session ID. Double Guard normalizes the variable values in both HTTP requests and database queries, preserving the structures of the requests and queries. To achieve this Double Guard substitutes the actual values of the variables with symbolic values.

VII. MAPPING RELATIONS

In Double guard classify the four possible mapping patterns. Since the request is at the origin of the data flows treat each request as the mapping source. In other word, the mappings in the model are always in the form of one request to a query set rm TO Qn.

Deterministic Mapping:
This is the most common and perfectly matched pattern. That is to Say that web request rm appears in all traffic with the SQL queries set Qn. For any session in the testing phase with the request rm, the absence of a query set Qn matching the request indicates a possible intrusion. On the other hand, if Qn is present in the session traffic without the corresponding rm, this may also be the sign of an intrusion. In static websites this type of mapping comprises the majority of cases since the same results should be returned for each time a user visits the same link.

Empty Query Set:
In special cases, the SQL query set may be the empty set. This implies that the web request neither causes nor generates any database queries. For example, when a web request for retrieving an image GIF file from the same web server is made, a mapping relationship does not exist because only the web requests are observed. This type of mapping is called rm assign empty. During the testing phase, we keep these web requests together in the set EQS.

No Matched Request:
In some cases, the web server may periodically submit queries to the database server in order to conduct some scheduled tasks, such as cron jobs for archiving or backup. This is not driven by any web request, similar to the reverse case of the Empty Query Set mapping pattern. These queries cannot match up with any web requests, and we keep these unmatched queries in a set NMR. During the testing phase, any query within set NMR is considered legitimate. The size of NMR depends on web server logic, but it is typically small.

Non Deterministic Mapping:
The same web request may result in different SQL query sets based on input parameters or the status of the webpage at the time the web request is received. In fact, these different SQL query sets do not appear randomly, and there exists a candidate pool of query sets. Each time that the same type of web request arrives, it always matches up with one (and only one) of the query sets in the pool. It is difficult to identify traffic that matches this pattern. This happens only within dynamic websites, such as blogs or forum sites.

Static Model Algorithm:
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 traffic is already separated by session, begin by iterating all of the sessions from 1 to N.

1. Start
2. Input data of HTTP request whether it is query or request.
3. HTTP request i.e. Input is store in the Session.
4. The session entry will be set as input itself.
5. It send query or request to virtual server for validation task.
6. If attack is found then virtual server automatically terminated the STTP request.
7. Else attack is not found the STTP request is forwarded to the original server.
8. It shows the data.
9. Exit.

Some web requests that could appear separately are still present as unit. During the training phase, we treat them as a single instance of web requests bundled together unless we observe a case when either of them appears separately Our next step is to decide the other two mapping patterns by assembling a white list for static file requests, including JPG,GIF, CSS, etc. HTTP requests for static files are placed in the EQS set. The remaining requests are placed in REQ. If we cannot find any matched queries for them, they will also be placed in the EQS set. In addition, all remaining queries in SQL will be considered as No Matched Request cases and placed into NMR. In this algorithm that takes the input of training data set and builds the mapping model for static websites. For each unique HTTP request and database query, the algorithm assigns a hash table entry, the key of the entry is the request or query itself, and the value of the hash entry is AR for the request or AQ for the query, respectively. The algorithm generates the mapping model by considering all three mapping patterns that would happen in static websites.

VIII. PERFORMANCE ANALYSIS

We will implement a prototype of Double Guard using a webserver with a back-end DB. We also sets up two testing websites, one static and the other dynamic. To evaluate the detection results for our system, we analyzed four classes of attacks and measured the false positive rate for each of the two websites.

Software Requirement Specification
Problem Statement:
To develop a system for Detecting Intrusions that is capable of assisting best solutions to problem in multitier web applications.
Input:
Network/traffic information across webserver between database servers.
Output:
Intrusion detection.
Hardware and software tools required for the system:

Content	To Build System	To Run The System
Hardware Required	1.Computer-Pentium III & above 2.Min 256Mb RAM 3.Hard Disk: 40GB	1.Computer-Pentium III & above 2.Min 256Mb RAM 3.Hard Disk: 40GB
Software Required	1.jdk1.6.0_07 2.Net Beans 6.9.1 3.My SQL	1.Java Run Time Environment 2.Apache tomato 3.Wamp Server-win 32-1.7.1 4.Jpcap. 5.Wincap 6.Internet explorer

IX. CONCLUSION

Proposed intrusion detection system that builds models of normal behavior for multitier web applications from both front-end web (HTTP) requests and back-end database (SQL) queries have presented. In the previous approach independent IDS is used to provide alerts unlike that now, Double Guard have used which form container-based IDS with multiple input streams to produce alerts. Such correlation of input 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. This is achieved by isolating the flow of information from each webserver session with a lightweight virtualization. Furthermore, it quantified the detection accuracy of proposed approach when it attempted to model static and dynamic web requests with the back-end file system and database queries.

It builds a well-correlated model for static websites, which proposed experiments proved to be effective at detecting different types of attacks. It also showed that this held true for dynamic requests where both retrieval of information and updates to the back-end database occur using the webserver front end. When proposed

prototype is deployed on a system that employed Apache webserver, a blog application, and a My SQL back end, Double Guard was able to identify a wide range of attacks with minimal false positives which depended on the size and coverage of the training sessions used.

It is my great pleasure in expressing sincere and deep gratitude towards my guide **Asst.Prof.B.K.Patil** for his valuable suggestions, guidance and constant support throughout this work.

Note: If any content found the same and any references aren't mentioned below, for that I am sincerely apologies

X. REFERENCES

[1] K.Bai, H. Wang, and P. Liu, "Towards Database Firewalls," Proc. Ann. IFIP WG 11.3 Working Conf. Data and Applications Security (DBSec ‟05), 2005.

[2] B.I.A. Barry and H.A. Chan, "Syntax, and Semantics-Base Signature Database f or Hybrid Intrusion Detection Systems," Security and Comm. Networks, vol. 2, no. 6, pp. 457-475, 2009.

[3] 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.

[4] M.Christodorescu and S. Jha, "Static Analysis of Executables to Detect Malicious Patterns," Proc. Conf. USENIX Security Symp.,2003.

[5] 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‟lSymp. Recent Advances in Intrusion Detection (RAID ‟07), 2007.

[6] H. Debar, M. Dacier, and A. Wespi, "Towards a Taxonomy of Intrusion- Detection Systems," Computer Networks, vol. 31, no. 9,pp. 805-822, 1999.

[7] Y. Hu and B. Panda, "A Data Mining Approach for Database Intrusion Detection," Proc. ACM Symp.Applied Computing (SAC), H. Haddad, A. Omicini, R.L. Wainwright, and L.M. Liebrock, eds., 2004.

[8] R. Ezumalai, G. Aghila, "Combinatorial Approach for Preventing SQL Injection Attacks", 2009 IEEE International Advance Computing Conference (IACC 2009) Patiala, India, 6-7 March 2009.

[9] Asha. N, M. Varun Kumar, Vaidhyanathan. G of Anomaly Based Character Distribution Models in the, "Preventing SQL Injection Attacks", International Journal of Computer Applications (0975 – 8887) Volume 52– No.13, August 2012 [10] Mehdi Kiani, Andrew Clark and George , "Evaluation e Detection of SQL Injection Attacks".The Third International Conference on Availability, Reliability and Security,0-7695-31024/08, 2008 IEEE.

[10] DoubleGuard: Detecting Intrusions in Multitier Web Applications Meixing Le, Angelos Stavrou, Member, IEEE, and Brent ByungHoon Kang, Member, IEEE.