Nita Prakash Saware, Manish Umale, Nidhi Maheswarkar / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 4, Jul-Aug 2013, pp.2007-2014

Detecting Intrusions in Multitier Web Applications

Nita Prakash Saware[1], Manish Umale[2], Nidhi Maheswarkar[3]
[1, 2] (Department of Computer Engineering Lokmanya Tilak College of Engineering Koparkhairane, Mumbai,
[3](Department of Computer Engineering, Dhyanganga College of Engineering Narhe, Pune

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. 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 web and database request and identifies the attacks like SQL injection attack which an independent IDS cannot do. The limitations of multitier IDS are its training sessions and functionality coverage problem. We have implemented Double Guard using an Apache web server with MySQL and lightweight virtualization. It uses the concept of causal mapping and assigns each client session to container. Each container is associated with an independent container ID and hence it enhances the security. The concept of container is a lightweight virtualization concept that provides a means of tracking the information flow from the web server to the database server for each session. For each client an virtual web server is created.

Keywords **-** Intrusion Detection System, Anomaly Detection, Web Server, Attacks, SQLIA, Classification of SQLIA.

I. INTRODUCTION

With the tremendous growth of internet and interconnections among computer systems, network 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 behavior 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 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.

In this paper, we have showed the difficulties to implement IDS in multitier architecture. Since it is difficult to implement multiple IDS, We have introduced a new protocol-Double guard. A multi-tier architecture (often referred to as n-tier architecture) is a client–server architecture in which presentation, application processing, and data management functions are logically separated. The most widespread use of multi-tier architecture is the three-tier architecture. Three-tier architectures typically comprise

a presentation tier, a business or data access tier, and a data tier

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. The limitation of multitier IDS is its training sessions and functionality coverage problem. We have implemented Double Guard using an Apache web server with MySQL and lightweight virtualization. It uses the concept of causal mapping and assigns each client session to container. Each container is associated with an independent container ID and hence it enhances the security. The concept of container is a lightweight virtualization concept that provides a means of tracking the information flow from the web server to the database server for each session. For each client a virtual web server is created.

This paper emphasizes on various aspects of SQL Inspection. Section II shows prevention techniques and operations in the previous work done in this field. Section III contains proposed solution using tokenization approach as well as conclusion part of this paper and future research directions to prevent SQLIA.

II. BACKGROUND AND RELATED WORK

The attacker's objective for using the injection technique is lies in gaining control over the application database. In a web based application environment, most of the web based applications, social web sites, banking websites, online shopping websites works on the principle of single entry point authentication which requires user identity and password. A user is identified by the system based on his identity.

This process of validation based on user name and password, is referred as authentication. Web architecture illustrated in Fig 1.showes general entry point authentication process. In general client send a HTTP request to the web server and web server in turn send it to the database layer. Database end contains relational tables so queries will be proceeding and result will be send to the web server. So entire process is database driven and each database contains many tables that are why SQLIA can be easily possible at this level.

SQL Injection is a basic attack used for mainly two intentions: first to gain unauthorized access to a database and second to retrieve information from database. Function based SQL Injection attacks are most important to notice because these attacks do not require knowledge of the application and can be easily automated [6].

Oracle has generally aware well against SQL Injection attacks as there is are multiple SQL statements that support (SQL server and Postages SQL), a no. of executive statements (SQL servers) and no. of INTO OUTFILE functions (MYSQL) [7]. Also use of blind variables in Oracle environments for performance reasons provides strong protections against SQL Injection attack.

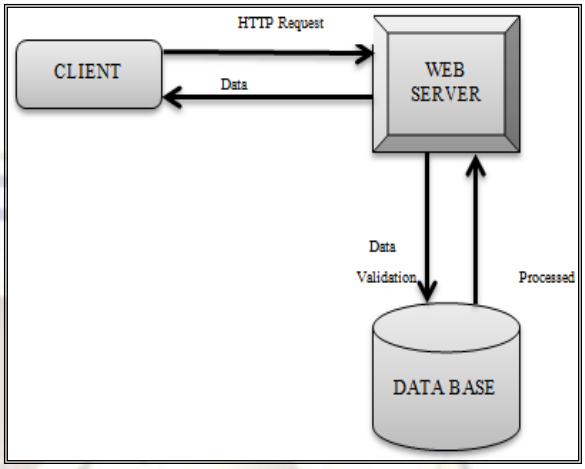


Figure 1.Web Architecture

There are two types of SQLIA detection:

Static approach: This approach is also known as pre-generating approach. Programmers follow some guidelines for SQLIA detection during web application development. An effective validity checking mechanism for the input variable data is also requires for the pre-generated method of detecting SQLIA.

Dynamic Approach: This approach is also known as post-generated approach. Post-generated technique are useful for analysis of dynamic or runtime SQL query, generated with user input data by a web application. Detection techniques under this post-generated category executes before posting a query to the database server [2, 7].

Classification of SQLIA: SQLIA can be classified into five categories:

- Bypass Authentication
- Unauthorized Knowledge of Database
- Unauthorized Remote Execution of Procedure
- Injected Additional Query
- Injected Union Query

Bypass Authentication: It is already discussed in Section I. Researchers have proved that query injection can't be applied without using space, single quotes or double dashes (--). In bypass authentication, intruder passes the query in such a way which is syntactically true and access the unauthorized data [8]. For example:

SELECT SALARY_INFO from employee where username=' or 1=1 - - 'and password='';

This SQL statement will be passed because 1=1is always true and - - which is used for

comments, when used before any statement, the statement is ignored.
So the result of this query allows intruder to access into user with its privileges in the database [3].

Unauthorized Knowledge of Database: In this type of attack, intruder injects a query which causes a syntax, or logical error into the database. The result of incorrect query is shown in the form of error message generated by the database and in many database error messages, it contains some information regarding database and intruder can use these details. This type of SQLIA is as follows:
SELECT SLARY_INFO from employee where username = 'rahul' and password =convert(select host from host);
This query is logically and syntactically incorrect. The error message can display some information regarding database. Even some error messages display the table name also.

Unauthorized Remote Execution of Procedure: SQLIA of this type performs a task and executes the procedures for which they are not authorized. The intruder can access the system and perform remote execution of procedure by injecting queries.
For example:
SELECT SALARY_INFO from employee where username='; SHUTDOWN; and password ='';
In above query, only SHUTDOWN operation is performed which shuts down the database [2].

Injected Additional Query: When an additional query is injected with main query and if main query generates Null value, even though the second query will take place and the additional query will harm the database. For example:
SELECT SALARY_INFO from employee where username='rahul' and password='; drop table user';
First query generates Null because the space is not present between 'and' and password, but the system executes the second query and if the given table present in database, it will be dropped.

Injected Union Query: In this type of attack, the intruder injects a query which contains set operators. In these queries, the main query generates Null value as a result but attached set operators data from database. For example:
SELECT SALARY_INFO from employee where username='' and password='' UNION SELECT SALARY_INFO from employee where emp_id='10125';
In above query, the first part of query generated Null value but it allows the intruder to access the salary information of a user having id 10125.

Major Elements of SQLIA:

It is shown in various research papers that SQLIA can't be performed without using space, single quotes and/or double dashes. These are the major elements of SQLIA. SQLIA is occurred when input from a user includes SQL keywords, so that the dynamically generated SQL query changes the intended function of the SQL query in the application.

When user input types a number, there is no need to use single quotes in the query. In this case SQL Injection is injected by using space. This query can be done on original query.
Original Query: SELECT * from employee where emp_id=10125;
The injection query can be of this form using space:
SELECT * from employee where emp_id=10125 or 1=1;

The injection query shown below is a query which uses single quotes:
SELECT*from employee where emp_name='rahul'or1=1;
In this case if an employee with name rahul is present in database, information is retrieved. But if the name is not present, even then the query is executed becausethestatement1=1 is always true.
The injection query may contain double dashes (--)
SELECT * from employee where emp_name='rahul';--'and SALARY_INFO>25000;

SQLIA is a prominent topic and lots of research work has been done for the detection and prevention of SQLIA. In [3] the author proposes the TransSQL model. In this model author proposes a model for SQLIA prevention.

TransSQL is server side application so, it does not changes legacy of web application. This model uses the idea of database duplication and run time monitoring. The proposed model is fully automated and the result shows the effectiveness of system. TransSQL propose to use two data bases, one is original relational database and another (LDAP) is copy of the first one, But data is arranged in hierarchical form. When a query is paused by the user, the system checks if the query contains the injection or not. The queries inserted in both original database and LDAP. If result of both databases is same, it shows the input query is free from injection, but if results are different, it means, the query contains injection. So the system shows the result as Null.

The major shortcoming of this models that it is not applicable for injection queries which contain instances, alias, UNION ad UNIONALL [11]. In [9], tokenization method is propose, which is efficient but as well as query with injection. It is not possible for all queries that their origin al query is a ready stored. In [2], the author proposes rule-based detection technique, which is based on classification task. For a particular query, rule dictionary is generated and query is replaced with these rules. If another query is present, the rules are applied in new entry and using classification approach, identify that new query contains the SQL injection or not.[2] proposes, two levels of authentication: SQL authentication and XML authentication, and every query is passed

though both systems for checking and preventing against SQLIA.

III. PROPOSED MODEL

In the title propose Double Guard which is used to detect attacks in multitier web services. In Double Guard, the new container-based web server architecture enables user to separate the different information flows by each session. 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 OpenVZ 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 as "containers," as 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.

Components of proposed system:

SQL Attack Module:

In this module, we have analyzed the four attacks that generally takes place. These attacks are Hijack Future Session Attack, Privilege Escalation Attack, and Injection Attack 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 web server side. An attacker usually takes over the web server and therefore hijacks all subsequent legitimate user sessions to launch Attacks. SQL injections do not require compromising the web server. Attackers can use existing vulnerabilities in the web server logic to inject the data or string content that contains the exploits and then use the web server to relay these exploits to attack the back-end database.

In a Direct DB attack, an attacker can bypass the web server or firewalls and connect directly to the database. Here we have shown how the attack takes 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 web server i.e. multiple virtual web servers are created in a single system using same service. So each client access through a virtual web server, in this way we can create multiple instance of server. Hence client can access a service through the web server which indicates the basic concept of Double guard architecture. Once client is initiated, it tries to login to use the service. Here we depict 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 web server sometimes attacker get confused about the original server and instance of the server.

Blog Creation Module:

In this module, we have showed 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 an 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 cannot 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 web server and also between the server and 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 give information about the Ethernet frames, the protocol used i.e. TCPIP 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 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, an Network layer is depicted. It shows the graphical usage pattern.

Advantages:

- Double guard provides high security since the usage of session for each subsequent web request. A session is dedicated to container which refer to the disposable server and a container ID is provided for each client.
- If any one session is attacked by intruder, others remain unaffected. It is very useful to identify attacks like session-hijacking, SQL injection attack etc.
- It not only provides security but also provides isolated information flow.
- It does not depend on time basis and hence provide a complete secure system. It provides an alert system which operates on multiple feeds of input.
- It does not require any input validation as it looks for the structure of request not on the input parameter.

SQLIA is a server type of web vulnerability, which impacts badly on web applications. In this section, a novel model for SQLIA prevention is proposed. As mentioned in previous section, several models are proposed for prevention of SQLIA, but they are not applicable for all type of injection attacks. SQLIA prevention via double authentication through tokenization is an approach to control SQLIA. In this paper we propose a new model names as Double Guard. In this model the system will identify the input. The input may be of two types it may be a request for certain service or information and it may be accepted or rejected by the system, and another one is Query which is generated to find out some specific information if query is syntactically correct it will display information, even then none of that type of information store in database it will display this information also the system store.

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 web server 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 web server and those generated for the database back end.

//Algorithm for Double Guard

Step 1: Identify the input type of HTTP request whether it is a query or a request.

Step 2: Store the input in hash table as per their type AQ for query and for request AR.

Step 3: The key for hash table entry will be set as the input itself.

Step 4: Forward AQ and AR to virtual server to validate.

Step 5: If attack identified then virtual system automatically terminate the HTTP request.

Step 6: Else HTTP request is forwarded to the original server.

Step 7: Display information.

Step 8: Exit.

Figure.2 Algorithm for Double Guard

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. Figure 3 (a) shows the proposed architecture and 3 (b) shows snapshots of Double guard by using above 7 essential steps.

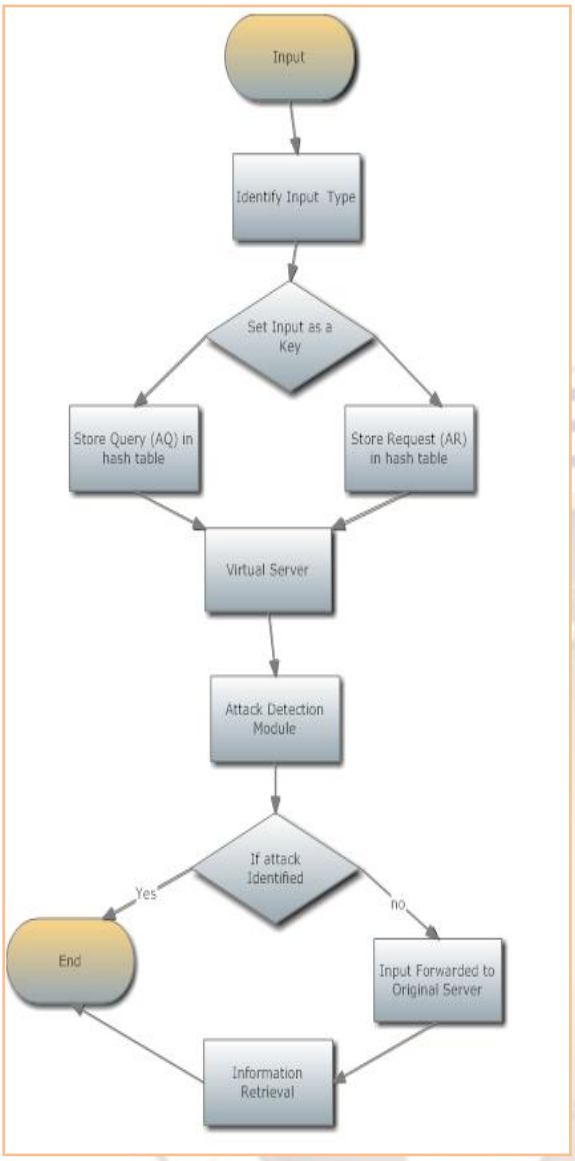
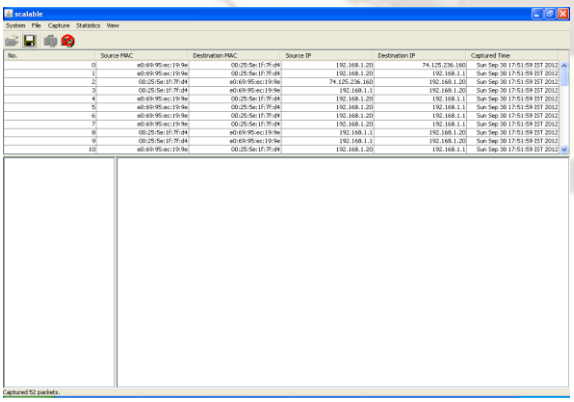
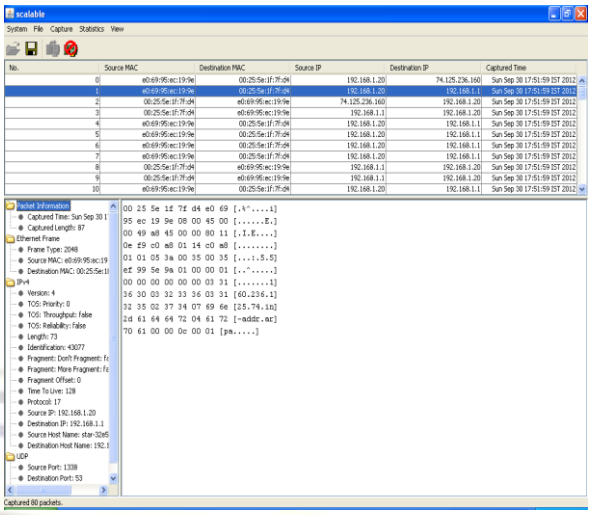
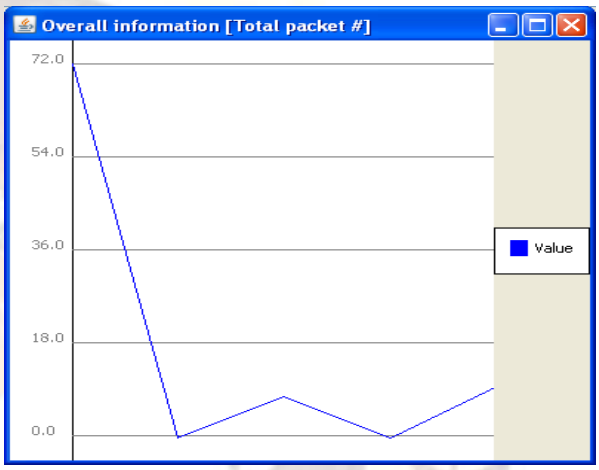


Figure.3 (a) Proposed Architecture







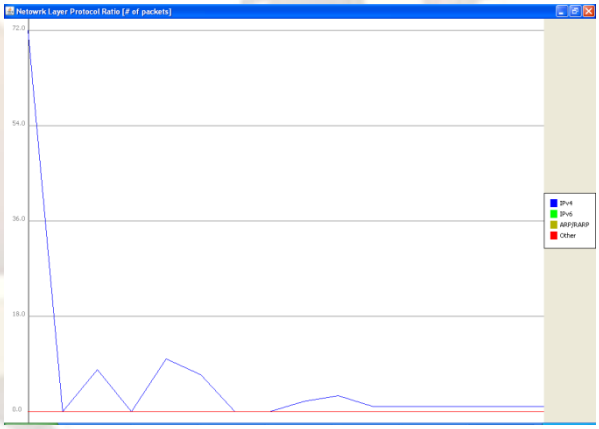


Figure.3 (b) Proposed Model Snapshots

IV. CHALLENGES

Intrusion detection systems aim at detecting attacks against computer systems and networks or in general, against information systems. Building IDS is a complex task of knowledge engineering that requires an elaborate infrastructure. There are various challenges that IDS faced today and which need to

concentrate while building IDS. The most common are:

Denial-of-service attack:

Denial-of-service attacks are common and fashionable these days. In denial-of-service attack, attacker tries to prevent legitimate users from using a service or shutting down a service owing to some implementation vulnerability crashing the machine. But today a new class of denial-of-service attacks has appeared based on the generation of normal traffic. An attacker subverts a number of machine then they install a distributed denial-of-service tool on all these machines, pointing it towards a single target.

All these machines will start sending repeated requests to the target, which often collapses under the load. The problem is that Intrusion-detection cannot able to distinguishes them from real or/and normal traffic.

Identification of the origin of attack:

Then the next problem with the Intrusion-detection product is the identification of the origin of attack. Because there are many software's/toolkits available on the internet which include facilities for disguising the user's identity such as IP spoofing etc and TCP/IP does not allow reliable identification of the source of packet which makes Intrusion detection system very difficult to identify the source of attack.

Domain Name System:

The Domain Name System (DNS) is a protocol used to associate machine names with IP addresses. When a machine wants to reach another one of which it knows only the name, it performs a DNS search by requesting the IP address corresponding to the name. This request goes to a local DNS server. If the local DNS server does not know the association, it in turns asks other servers, "secondary DNS servers", with more general knowledge, up to the 13 root servers that have the entire map of the Internet and can answer DNS queries.

To avoid referring constantly to the root servers, the secondary servers keep a cache of the associations between hostnames and addresses. When a request can be answered by data in the cache, the secondary DNS server does not go further up the chain. The DNS protocol also has a means to update the cache, which can be exploited to poison the cache of a secondary DNS server by associating a different IP address with the same name. This will effectively redirect all requests to a server to another machine under the control of the attacker.

V. CONCLUSION

In this paper, we have presented an 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. In the previous approach we have used independent IDS to provide alerts unlike that now we have used, Double Guard which forms container-based IDS with multiple input streams to produce alerts. We have shown that 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.

We achieved this by isolating the flow of information from each web server session with a lightweight virtualization. Furthermore, we quantified the detection accuracy of our approach when we attempted to model static and dynamic web requests with the back-end file system and database queries. We have built a well-correlated model for static websites, which our 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 web server front end.

When we deployed our prototype on a system that employed Apache web server, a blog application, and a MySQL 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 we used. In this project we use TDT4 method to provide effective summarization methods to extract the core parts of detected topics, as well as graphic representation methods to depict the relationships between the core parts. Applied together, the two techniques, called topic anatomy, can summarize essential information about a topic in a structured manner. In future we can retrieve information from tscan by using our voice instead of typing text. The user can search through voice; the system will recognize this voice and provide essential information about a topic.

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'l Symp. 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 th,"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,0-7695-3102-4/08, 2008 IEEE.

[11] V.Shanmughaneethi, C.EmilinShyni and Dr.S.Swamynathan, "SBSQLID: Securing Web Applications with Service Based SQL Injection Detection" 2009 International Conference on Advancesin Computing, Control, and Telecommunication Technologies, 978-0-7695-3915-7/09, 2009 IEEE

[12] Yuji Kosuga, Kenji Kono, Miyuki Hanaoka, Hiyoshi Kohoku-ku, Yokohama, Miho Hishiyama, Yu Takahama, Kaigan Minato-ku, "Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection" 23rd Annual Computer Security Applications Conference, 2007, 1063-9527/07, 2007 IEEE

[13] Prof (Dr.) Sushila, MadanSupriyaMadan, "Shielding Against SQL Injection Attacks Using ADMIRE Model", 2009 First International Conference on Computational Intelligence, Communication Systems and Networks, 978-0-7695-3743-6/09 2009 IEEE

[14] A S Yeole, B BMeshram, "Analysis of Different Technique for Detection of SQL Injection", International Conference and Workshop on Emerging Trends in Technology (ICWET 2011) – TCET, Mumbai, India, ICWET'11, February 25–26, 2011, Mumbai, Maharashtra, India. 2011 ACM.

[15] Ke Wei, M. Muthuprasanna, Suraj Kothari, "Preventing SQL Injection Attacks in Stored Procedures".Proceedings of the 2006 Australian Software Engineering Conference (ASWEC'06).

[16] Debasish Das, Utpal Sharma, D. K. Bhattacharyya, "Rule based Detection of SQL Injection Attack", International Journal of Computer Applications (0975 – 8887) Volume 43– No.19, April 2012.

[17] NTAGW ABIRA Lambert, KANG Song Lin, "Use of Query Tokenization to detect and prevent SQL Injection Attacks", 978-1-4244-5540-9/10/2010 IEEE.

[18] Kai-Xiang Zhang, Chia-Jun Lin, Shih-Jen Chen, Yanling Hwang, Hao-Lun Huang, and Fu-Hau Hsu, "TransSQL: A Translation and Validation-based Solution for SQL-Injection Attacks", First International Conference on Robot, Vision and Signal Processing, IEEE, 2011.