iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** III **Month of publication:** March 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A Clustering Based Efficient Intrusion Detecting In Multitier Dynamic Web Applications

Kirthiga. N
Under the guidance of Mrs.J.Lekha Msc.,Mphil Assistant professor,SKASC.

Abstract: ***We present DoubleGuard, a system used to detect attacks in multitiered web services and classify through Hierarchal clustering Algorithm. Our approach can create normality models of isolated user sessions that include both the web front-end (HTTP) and back-end (File or SQL) network transactions with respect to Data volumes and Classify them. Implements a lightweight virtualization technique to assign each user's web session to a dedicated container, an isolated virtual computing environment. We use the cluster algorithm to accurately associate the web request with the subsequent DB queries. DoubleGuard can build a causal mapping profile by taking both the webserver and DB traffic into account.***

I. INTRODUCTION

An intrusion detection system (IDS) is a device or software application that monitors network or system activities for malicious activities or policy violations and produces reports to a Management Station. Some systems may attempt to stop an intrusion attempt but this is neither required nor expected of a monitoring system. Intrusion detection and prevention systems (IDPS) are primarily focused on identifying possible incidents, logging information about them, and reporting attempts. In addition, organizations use IDPS for other purposes, such as identifying problems with security policies, documenting existing threats and deterring individuals from violating security policies. IDPSes have become a necessary addition to the security infrastructure of nearly every organization.

A. Introduction To Ids

Internet services and applications have become an inextricable part of daily life, enabling communication and the management of personal information from anywhere. To accommodate this increase in application and data complexity, web services have moved to a multitier design wherein the web server runs the application front-end logic and data are outsourced to a database or file server. In this paper, we present Double Guard, an IDS system that models the network behavior of user sessions across both the front-end web server and the back-end database. By monitoring both web and subsequent database requests, we are able to ferret out attacks that independent IDS would not be able to identify. Furthermore, we quantify the limitations of any multitier IDS in terms of training sessions and functionality coverage. We implemented Double Guard using an Apache web server with MySQL and lightweight virtualization.

To protect multitier web services, Intrusion detection systems have been widely used to detect known attacks by matching misused traffic patterns or signatures. A class of IDS that leverages machine learning can also detect unknown attacks by identifying abnormal network traffic that deviates from the so-called "normal" behavior previously profiled during the IDS training phase.

Individually, the web IDS and the database IDS can detect abnormal network traffic sent to either of them. However, we found that these IDSs cannot detect cases wherein normal traffic is used to attack the web server and the database server.

For example, if an attacker with non admin privileges can log in to a web server using normal-user access credentials, he/she can find a way to issue a privileged database query by exploiting vulnerabilities in the web server.

Neither the web IDS nor the database IDS would detect this type of attack since the web IDS would merely see typical user login traffic and the database IDS would see only the normal traffic of a privileged user. This type of attack can be readily detected if the database IDS can identify that a privileged request from the web server is not associated with user-privileged access.

Unfortunately, within the current multithreaded web server architecture, it is not feasible to detect or profile such causal mapping between web server traffic and DB server traffic since traffic cannot be clearly attributed to user sessions.

II. RELATED WORK

A. A Review of Anomaly based Intrusion Detection Systems.

In **Anomaly-Based** Intrusion Detection System, is a system for detecting computer intrusions and misuse by monitoring system

activity and classifying it as either normal or anomalous. The classification is based on heuristics or rules, rather than patterns or signatures, and will detect any type of misuse that falls out of normal system operation. This is as opposed to signature based systems which can only detect attacks for which a signature has previously been created.

Problem Statement:

Signature-based System (SBS) and Anomaly-based System (ABS). SBS systems rely on pattern recognition techniques where they maintain the database of signatures of previously known attacks and compare them with analyzed data. An alarm is raised when the signatures are matched. On the other hand ABS systems (e.g. PAYL [18]) build a statistical model describing the normal network traffic, and any abnormal behavior that deviates from the model is identified. In contrast to signature-based systems, anomaly-based systems have the advantage that they can detect zero-day attacks, since novel attacks can be detected as soon as they take place. Whereas ABS (unlike SBS) requires a training phase to develop the database of general attacks and a careful setting of threshold level of detection makes it complex.

Methodology

Anomaly detection is based on a host or network. Many distinct techniques are used based on type of processing related to behavioral model. They are: Statistical based, Operational or threshold metric model, Markov Process or Marker Model, Statistical Moments or mean and standard deviation model, Univariate Model, Multivariate Model, Time series Model, Cognition based, Finite State Machine Model, Description script Model, Adept System Model, Machine Learning based, Baysian Model, Genetic Algorithm model, Neural Network Model, Fuzzy Logic Model, Outlier Detection Model, Computer Immunology based, User Intention based. The most significant open issues regarding Anomaly based Network Intrusion Detection systems are identified, among which assessment is given particular emphasis. The presented information constitutes an important point to start for addressing Research & Development in the field of IDS. Counter measures which are faster and more effective are needed to cope up with the attacks ever-growing.

Drawback:

The two approaches have the potential problem of being integrated in one comprehensive solution, where the server that receives requests deemed anomalous by our anomaly detection system is modified to make it more resilient to certain types of attacks. This is certainly an inefficiency of the current research direction and will be the setback for the entire system.

The implementation of the system is limited and is not designed to dealing with buffer overflow/underflow attacks, while our system is able to mitigate also non-control-data attacks. This issue leads to that might eventually to the corruption of the entire database. The attacks that can be rolled-back are limited to attacks against the web server itself, while our focus includes also all server-side components such as CGI programs, server-side scripts, and back-end databases.

While the approach presented does not make any distinction in the type of information managed by the protected web-based system

The creation of a “shadow” server requires the modification of the source code of the web server, and, if extended to server-side applications, to the application code.

B. Intrusion Detection via Static Analysis

This paper focus on some security problems are directly attributable to faulty application logic, such as programs that fail to check authentication information before proceeding and one limitation of our intrusion detection system is that it does not detect attacks that exploit logic errors. Application logic bugs, however, are dwarfed in practice by buffer overflow problems and other vulnerabilities that allow for execution of arbitrary machine code of the attacker’s choice, and it is the latter type of vulnerability.

1) *Problem Statement:* Every program should come with a specification of its intended behavior. This, of course, has been the dream of the formal methods community for 25 years, and is as yet unrealized. We believe it is likely to remain unrealized for some time to come. Although Ko *et al.*’s specification language is simple and admits relatively compact specifications, we believe that the need for manually written specifications will dramatically limit the impact of this work1. We philosophically agree with the direction of Ko *et al.*’s work, but we propose to side-step its main drawback by automatically deriving the specification from the program.
2) *Drawback:* Limitations of these signature engines are that they only detect attacks whose signatures are previously stored in database; a signature must be created for every attack; and novel attacks cannot be detected. This technique can be easily deceived because they are only based on regular expressions and string matching. These mechanisms only look for strings within packets transmitting over wire.

More over signatures work well against only the fixed behavioral pattern, they fail to deal with attacks created by human or a worm with self-modifying behavioral characteristics.

Signature based detection does not work well when the user uses advanced technologies like nop generators, payload encoders

and encrypted data channels.

The efficiency of the signature based systems is greatly decreased, as it has to create a new signature for every variation. As the signatures keep on increasing, the system engine performance decreases.

C. A Comprehensive Approach to Intrusion Detection Alert Correlation

This paper address the issue, researchers and vendors have proposed alert correlation, an analysis process that takes the alerts produced by intrusion detection systems and produces compact reports on the security status of the network under surveillance. Although a number of correlation approaches have been suggested, there is no consensus on what this process is or how it should be implemented and evaluated. In particular, existing correlation approaches operate on only a few aspects of the correlation process, such as the fusion of alerts that are generated by different intrusion detection systems in response to a single attack, or the identification of multistep attacks that represent a sequence of actions performed by the same attacker.

1) *Problem Statement:* Correlation tools that do cover multiple aspects of the correlation process are evaluated "as a whole," without an assessment of the effectiveness of each component of the analysis process. As a result, it is not clear if and how the different parts of the correlation process contribute to the overall goals of correlation.
2) *Methodology*: Correlation process: The main objective of the correlation process is to produce a succinct overview of security-related activity on the network. This process consists of a collection of components that transform intrusion detection sensor1 alerts into intrusion reports. Because alerts can refer to different kinds of attacks at different levels of granularity, the correlation process cannot treat all alerts equally. Instead, it is necessary to provide a set of components that focus on different aspects of the overall correlation task.
3) *Drawback:* The presence/absence and the parameter order model can be evaded without much effort by an adversary that has sufficient knowledge of the structure of a legitimate query. Even after including these models into our IDS the system remains inefficient, especially because of the low number of false alarms they produce.

A limitation of the system is its reliance on web access logs. Attacks that compromise the security of a web server before the logging is performed may not be detected.

The approach advocates the direct instrumentation of web servers in order to perform timely detection of attacks, even before a query is processed. This approach may introduce some unwanted delay in certain cases, but if this delay is acceptable then the system described here could be easily modified to fit that model.

D. Intrusion Recovery for Database-backed Web Applications

In this paper Users or administrators must manually inspect the application for signs of an attack that exploited the vulnerability, and if an attack is found, they must track down the attacker's actions and repair the damage by hand. When an administrator learns of security vulnerability in a web application, he or she can use WARP to check whether that vulnerability was recently exploited, and to recover from any resulting intrusions

1) *Problem Statement:* An attack can affect users' browsers, making it difficult to track down the extent of the intrusion purely on the server. For example attack, when Alice (or any other user) visits an infected Wiki page, the web server cannot tell if a subsequent page edit request from Alice's browser was caused by Alice or by the malicious JavaScript code. Yet an ideal system should revert all effects of the malicious code while preserving any edits that Alice made from the same page in her browser.

WRAP: WARP's workflow begins with the administrator deciding that he or she wants to make a retroactive fix to the system, such as applying a security patch or changing permission in the past.

At a high level, WARP then rolls back the system to a checkpoint before the intended time of the fix, applies the fix, and re-executes actions that happened since that checkpoint, to construct a new system state. This produces a repaired system state that would have been generated if all of the recorded actions happened on the fixed system in the first place.

If some of the recorded actions exploited a vulnerability that the fix prevents, those actions will no longer have the same effect in the repaired system state, effectively undoing the attack. If the application is non-deterministic, there may be many possible repaired states, and WARP only guarantees to provide one of them, which may not necessarily be the one closest to the pre-repair state.

In other words, non-deterministic changes unrelated to the attack may appear as a result of repair, and non-determinism may increase the number of actions re-executed during repair, but the repaired state is guaranteed to be free of effects of attack actions.

Also, due to changes in system state during repair, some of the original actions may no longer make sense during replay, such as when a user edits a Wiki page created by the attacker and that page no longer exists due to repair. These actions are marked as

conflicts and WARP asks the user for help in resolving them.

WAPR cannot undo disclosures of private data, such as if an adversary steals sensitive information from Wiki pages, or steals a user's password.

However, when private data is leaked, WARP can still help track down affected users. Additionally, in the case of stolen credentials, administrators can use WARP to retroactively change the passwords of affected users (at the risk of undoing legitimate changes), or revert just the attacker's actions, if they can identify the attacker's IP address.

2) *Drawback:* The Proposed system previously explored intrusion recovery for command-line workloads on a single machine; WARP is the first system to repair from such attacks in web applications. Recovering from intrusions such as the example above requires WARP to address three challenges not answered by the current system.

Recovering from an intrusion (e.g., in Retro) typically requires an expert administrator to detect the compromise and to track down the source of the attack, by analyzing database entries and web server logs.

Worse yet, this process must be repeated every time a new security problem is discovered, to determine if any attackers might have exploited the vulnerability

Web applications typically handle data on behalf of many users, only a few of which may have been affected by an attack.

E. Advanced SQL Injection in SQL Server Applications

This document discusses in detail the common "SQL injection" technique, as it applies to the popular Microsoft Internet Information Server/Active Server Pages/SQL Server platform. It discusses the various ways in which SQL can be "injected" into the application and addresses some of the data validation and database lockdown issues that are related to this class of attack. The paper is intended to be read by both developers of Web applications which communicate with databases and by security professionals whose role includes auditing these Web applications.

III. PROPOSED SYSTEM

We present Double Guard, a system used to detect attacks in multitier web services and classify through Hierarchal clustering Algorithm. Our approach can create normality models of isolated user sessions that include both the web front-end (PHP) and back-end (File or SQL) network transactions with respect to Data volumes and Classify them.

Implements a lightweight virtualization technique to assign each user's web session to a dedicated container, an isolated virtual computing environment. We use the cluster algorithm to accurately associate the web request with the subsequent DB queries. Double Guard can build a causal mapping profile by taking both the web server and DB traffic into account.

The system builds a causal mapping profile by taking both the web server and DB traffic into account. The system uses a multi-tier approach which makes web applications retain their simplicity for the user and complexity for the attacker.

Our Proposed system implements prototype, which show that, 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.

In fact, we show that this causality-mapping model can be generated accurately and without prior knowledge of web application functionality.

Our experimental evaluation, using real-world network traffic obtained from the web and database requests of a large center, showed that we were able to extract more than 100 percent of functionality mapping by using as few as multiple sessions in the training phase. Of course, we also showed that this depends on the size and functionality of the web service or application.

However, it does not depend on content changes if those changes can be performed through a controlled environment and retrofitted into the training model. We refer to such sites as "static" because, though they do change over time, they do so in a controlled fashion that allows the changes to propagate to the sites' normality models. In addition to this static website case, there are web services that permit persistent back-end data modifications.

These services, which we call dynamic, allow HTTP requests to include parameters that are variable and depend on user input. Therefore, our ability to model the causal relationship between the front end and back end is not always deterministic and depends primarily upon the application logic. For instance, we observed that the backend queries can vary based on the value of the parameters passed in the HTTP requests and the previous application state.

Sometimes, the same application's primitive functionality (i.e., accessing a table) can be triggered by many different WebPages. Therefore, the resulting mapping between web and database requests can range from one to many, depending on the value of the parameters passed in the web request. To address this challenge while building a mapping model for dynamic web pages, we first generated an individual training model for the basic operations provided by the web services. We demonstrate that this approach works well in practice by using traffic from a live blog where we progressively modeled nine operations.

Our results show that we were able to identify all attacks, covering more than 99 percent of the normal traffic as the training model is refined.

A. Algorithm

1) *Data Preprocessing using K Means:* The algorithm accepts two inputs. The data (container) itself, and "k", the number of clusters. We will talk about the implications of specifying "k" later. The output is k clusters with input data partitioned among them. The aim of K-means (or clustering) is this: We want to group the items into k clusters such that all items in same cluster are as similar to each other as possible. And items not in same cluster are as different as possible. We use the distance measures to calculate similarity and dissimilarity. One of the important concepts in K-means is that of centroid.

Each cluster has a centroid. You can consider it as the point that is most representative of the cluster. Equivalently, centroid is point that is the "center" of a cluster.

B. Algorithm

1) Randomly choose k items and make them as initial centroids.
2) For each point, find the nearest centroid and assign the point to the cluster associated with the nearest centroid.
3) Update the centroid of each cluster based on the items in that cluster. Typically, the new centroid will be the average of all points in the cluster.
4) Repeats steps 2 and 3, till no point switches clusters.

The output of the clustering is given as input to above modules and we get optimized result for detecting intrusion in website.

C. Module Description

1) *Building the Normality Model:* We deployed a static testing website using the Content Management System. We chose to assign each user session into a different container. We can assign a new container per each new IP address of the client. The container will log all the SQL Queries executed by client in database. Deterministic Mapping and the Empty Query Set Mapping patterns are discovered from training sessions.
2) *Finding Deterministic Mapping Queries***:** Deterministic Mapping is the most common and perfectly matched pattern. Web request rm appears in all traffic with the SQL queries set Qn. If Qn is present in the session traffic without the corresponding rm is classified as intrusion
3) *Finding 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. Ex 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->O ;. During the testing phase, we keep these web requests together in the set EQS.

D. Hijack Future Session Attack And Injection Attack

An attacker usually takes over the web server and therefore hijacks all subsequent legitimate user sessions to launch attacks. We find hijacking other user sessions, the attacker can eavesdrop, send spoofed replies, and/or drop user requests.

1) *Finding Injection Attack:* 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.

This SQL injection attack changes the structure of the SQL queries, even if the injected data were to go through the web server side, it would generate SQL queries in a different structure that could be detected as a deviation from the SQL query structure that would normally follow such a web request.

This module contains the unique idea that compares SQL query strings and blocks suspicious sql-query and passes original sql-query. The classification of Suspecious query is done by analyzing the datasets of Original query and suspicious query. Classifies learns the dataset and according to learning procedure, it classifies the queries.

2) *Nondeterministic 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. These different SQL query sets do not appear randomly, and there exists a candidate pool of query sets Qn. This Query is classified as intrusion using non deterministic mapping.

IV. EXPERIMENTAL RESULTS

The user sessions are tracked by the ip address, date and time of visit in container. The full details about the users can be viewed by the administrator. The tracking of the details except ip address is general. The ip address is tracked by using the following one line code: $ip=$_SERVER ['REMOTE_ADDR']. This code is placed in the code file to track the users who are all visiting the web site.

The container is stored in the web server and is available in the admin end. The container file consists of the information about the query, ip address, date and time of visit. It consists of all records i.e., the information about all the clients who are all visited the web site with their database query. From 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.

To evaluate the detection results for the system, analyze classes of attacks like deterministic mapping, sql injection, empty query list and etc. When deployed a prototype on a system that employed Apache web server, a cms application and a MySQL back end. The Proposed system was able to identify a wide range of attacks with minimal efforts.

We implement real spatial datasets by developing online portal website for real-estate agencies. We cleaned the datasets by discarding records without features and distance. The object dataset D contains 150 dataset. The feature dataset contains 300 features records for a location. We compare the cost of the algorithms with respect for queries with influence scores.

V. RESULTS AND DISCUSSION

In this mapping model and we found that only the Deterministic Mapping and the Empty Query Set Mapping patterns appear in the training sessions. We expected that the No Matched Request pattern would appear if the web application had a cron job that contacts back-end database server; however, our testing website did not have such a cron job.

We first collected 338 real user sessions for a training data set before making the website public so that there was no attack during the training phase. 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 , 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 20 times and obtained the average false positive rate (since there was no attack in the training data set) 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.

In another experiment, we used sqlmap to attack the websites. This tool tried out all possible SQL injection combinations as a URL and generated numerous abnormal queries that were detected by Double Guard. Green SQL was also effective at detecting these attacks, which shows its ability to detect SQL injection attacks.

Regarding Snort, although it is possible to write user-defined rules to detect SQL injection attack attempts, our experiments did not result in Snort reporting any SQL injection alerts. SQL injection attacks can be mitigated by input validation.

However, SQL injection can still be successful because attackers usually exploit the vulnerability of incorrect input validation implementation, often caused by inexperienced or careless programmers or imprecise input model definitions.

We establish the mappings between HTTP requests and database queries, clearly defining which requests should trigger which queries. For an SQL injection attack to be successful, it must change the structure (or the semantics) of the query, which our approach can readily detect.

VI. CONCLUSION

We presented an intrusion detection system that builds models of normal behavior for multitier web applications from both front-end web (PHP) requests and back-end database (MYSQL) queries. Unlike previous approaches that correlated or summarized alerts generated by independent IDSs, Double Guard 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.

VII. FUTURE ENHANCEMENT

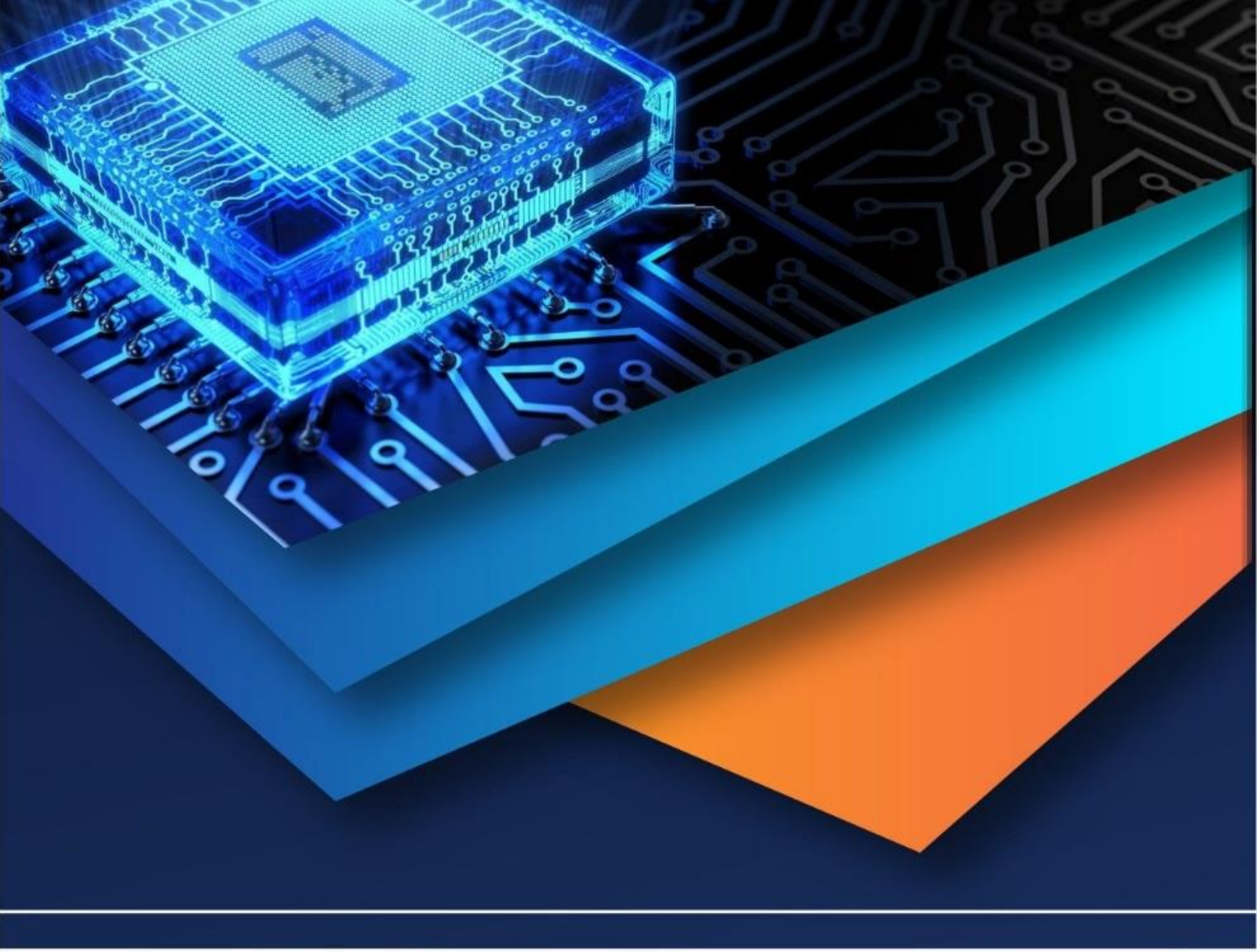
The future work of the detection is to improve accuracy of our approach when we attempted to model static and dynamic web requests with the back-end file system and database queries. For static websites, we built a well-correlated model, which our experiments proved to be effective at detecting different types of attacks. Moreover, we 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. As expected, the number of false positives depended on the size and coverage of the training sessions we used.

VII. REFERENCES

[1] A Review of Anomaly based Intrusion Detection Systems, Authors: V. Jyothsna ,V. V. Rama Prasad , K. Munivara Prasad
[2] David Wagner, Drew Dean "Intrusion Detection via Static Analysis".proc. Symp.security and privacy(SSP '01), May 2001.
[3] Ramesh Chandra, Taesoo Kim, Meelap Shah, Neha Narula, Nickolai Zeldovich "Intrusion Recovery for Database-backed Web Application"
[4] C. Anley, "Advanced Sql Injection in Sql Server Applications," technical report, Next Generation Security Software, Ltd., 2002.
[5] 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.
[6] H. Debar, M. Dacier, and A. Wespi, "Towards a Taxonomy ofIntrusion-Detection Systems," Computer Networks, vol. 31, no. 9,pp. 805-822, 1999.
[7]Y. Huang, A. Stavrou, A.K. Ghosh, and S. Jajodia, "Efficiently Tracking Application Interactions Using Lightweight Virtualization,"Proc. First ACM Workshop Virtual Machine Security,2008.
[8] T. Verwoerd and R. Hunt, "Intrusion Detection Techniques and Approaches," Computer Comm., vol. 25, no. 15, pp. 1356-1365, 2002.
[9] Fredrik Valeur , Giovanni Vigna, Christopher Kruegel, Richard A. Kemmerer "A Comprehensive Approach to Intrusion Detection Alert Correlation" IEEE Trans. Dependable and Secure Computing, vol 1, no.3,pp.146-169,july-sept.2004.
[10] B.I.A. Barry and H.A. Chan, "Syntax, and Semantics-Based Signature Database for Hybrid Intrusion Detection Systems,"Security and Comm. Networks, vol. 2, no. 6, pp. 457-475, 2009.
[11] G. Vigna, F. Valeur, D. Balzarotti, W.K. Robertson, C. Kruegel, andE. Kirda, "Reducing Errors in the Anomaly-Based Detection of Web-Based Attacks through the Combined Analysis of Web Requests and SQL Queries," J. Computer Security, vol. 17, no. 3,pp. 305-329, 2009
[12] 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.
[13]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.
[14] M. Christodorescu and S. Jha, "Static Analysis of Executables to Detect Malicious Patterns," Proc. Conf. USENIX Security Symp.,2003.
[15]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.

AUTHOR

KIRTHIGA. N completed Bsc Computer Science and Msc Computer Science in Vlb Janakiammal Arts And Science College and pursuing Mphil(Cs) in Sri Krishna Arts And Science College under the guidance of Mrs.J.Lekha Msc.,Mphil.,,Assistant Professor.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY