Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 04, April-2021

# Preventing Interference Detection in Multi Tier Cyber Security Applications

Y. Bhaskar Reddy[1]
[1] Assistant Professor,
Department of Computer Science Engineering, K.S.R.M. College of Engineering (Autonomous), Kadapa, Andhra Pradesh, India-516003

A. Ram Prakash Reddy [2]
[2] Assistant Professor,
Department of Computer Science Engineering, K.S.R.M. College of Engineering (Autonomous), Kadapa, Andhra Pradesh, India-516003

Dr. B. Sudarshan[3]
[3] Associate Professor,
Department of Mechanical Engineering, K.S.R.M. College of Engineering (Autonomous), Kadapa, Andhra Pradesh, India-516003

***Abstract*:- Both the web and database servers are vulnerable with this reference. Attacks are network-based and originate from web clients; they may use application-layer attacks to compromise the web servers to which they bind. Attackers can get around the web server and go straight to the database server. Delivered through the internet over the last few years, the popularity and sophistication of services and applications has increased. Banking, travel, and social networking are all done on the internet these days. A web server front end, which runs the application user interface logic, and a back-end server, which is usually a database or file server, are used in such services. Web services have long been a focus of attacks due to their widespread use for personal and/or corporate data. As focus has changed from targeting the front end to leveraging vulnerabilities in web applications in order to corrupt the back-end database system, these attacks have become more diverse (e.g., SQL injection attacks). In both the web server and the database system, a variety of Interface Detection Systems (IDSs) currently inspect network packets individually.**

***Key Words: Interface detection, Deep learning, Deep Q Model, Security, internet***

## 1. INTRODUCTION:

Green SQL, for example, serves as a reverse proxy for database connections. Web apps can bind to a network firewall first, rather than a database server. SQL queries are examined and, if found to be secure, forwarded to the back-end database server. To achieve more accurate identification, the method proposed in combines web IDS and database IDS, as well as a reverse HTTP proxy to retain a reduced level of service in the presence of false positives. However, we discovered that certain forms of attacks use regular traffic and are undetectable by either the network or database IDS. There will be no warnings to correlate with such situations. Previously, static analysis of source code or executables was used to identify interfaces or vulnerabilities. Others use complex information flow tracking to understand taint propagation and identify interfaces. The latest container-based web server architecture in Double Guard allows us to isolate the different information flows by session. For each session, this allows you to monitor the information flow from the web server to the database server. We also don't need to evaluate the source code or understand the application logic with our approach[6]. Our Double Guard approach does not require application logic to construct a model for a static web page. . While we don't need the full application logic for dynamic web services, we do need to know the basic user operations to model normal behavior, as we'll see.

## 2. LITERATURE REVIEW:

The research published in exposed flaws in many previous works involving the classification of network attacks using an out-of-date data set such as the KDDCup'99 for performance evaluation [1]. Such related studies normally achieve high classification rates, but this is partially due to flaws in the KDDCup'99 dataset. This hypothesis was put to the test by comparing the results of [2]. With the exception of a practical and modern dataset, accuracy by typical ML algorithms with the same ML approaches: [3] NGIDS-DS. In comparison, the same machine learning algorithms were used, but the authors' approach was not comprehensive. The authors of [4] proposed IDS architecture as a solution to emerging problems in real-time handling of large amounts of data. The efficiency of the attack classification was tested using well-known ML algorithms on the old-fashioned data set KDDCup'99 to assess the process. In addition, a thorough examination of the impact of various selection techniques on classification efficiency is given. Finally, the best results in terms of modelling and detection time were obtained using a suggested FS method combined with the analysis of manual exploratory features and two techniques (FSR and BER). The writers made no mention of FE, DP, or HS, making it difficult to reproduce and test the recorded assumptions [5]. Provided a practical network dataset of current attacks and normal traffic dynamics from an emulated network environment. They tested the dataset's feasibility before using up to seven traditional ML algorithms to identify network-based attacks. For this purpose, any network traffic-based features of a single tool have been extracted. They then used the Random Forest Regress technique or weighted specific characteristics based on the attack. Neither the HS nor the DP techniques, however, have been proven.

## 3. PROPOSED MODEL:

By statically examining the source code or execute tables, some previous methods were able to detect interfaces or vulnerabilities. Others use complex information flow tracking to understand taint propagation and identify interfaces. The new container-based web server architecture in Double Guard allows us to isolate different information flows by session. For each session, this allows you to monitor the information flow from the web server to the database server. We also don't need to study the source code or understand the application logic with our approach. Our Double Guard approach does not require application logic to construct a model for a static web page. While we do not need the full application logic for dynamic web services, we do need to understand the basic user operations in order to model normal behaviour, as we will discuss.

### 3.1. Flow of Work:

1. In a typical multitier app, only the client and the Web Server maintain a session. Between the Web Server and the database or file server, no session is maintained. As a result, the intruder has access to the database or file server.

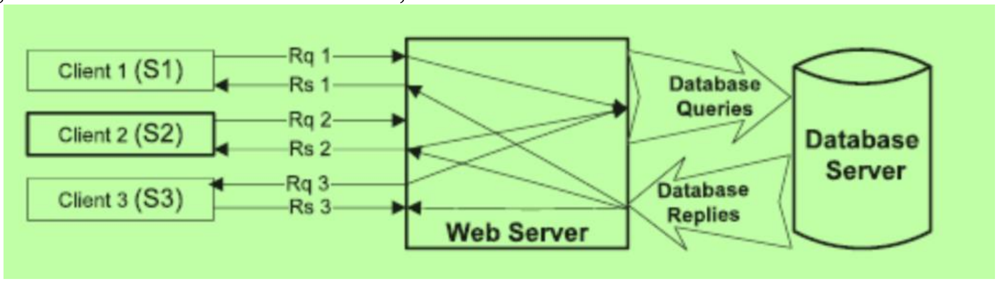

Figure 1: In normal multitier app, session maintained between only client and Web Server

1. So by providing double guard security, (means session for both web and db or file server) we can block intruder

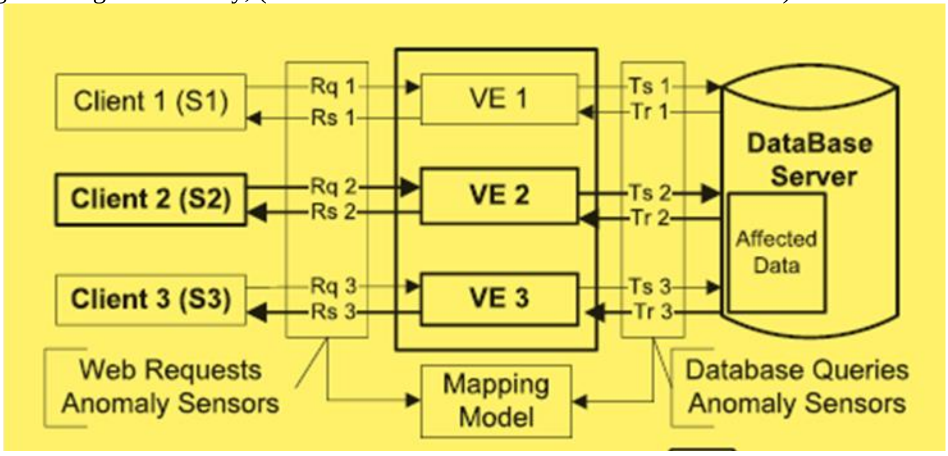

Figure 2: Double Guard Security

**The Proposed System's Features:**

- Provides multi-layer protection.
- The web server is secured by a multi-layer security system that detects multiple operations at different times.
- We suggest a radically new approach to multilayer security protection.

### 3.2 MODULES

Assume the website covers both frequent users and administrators. 3.2.1. Privilege Escalation Attack: The web request ru will execute the set of SQL queries Qu for a daily user, while the request ra will execute the set of admin level queries Qa for an administrator. Assume that an intruder logs in as a regular user, upgrades his or her privileges, and runs admin queries in order to access an administrator's data.

#### 3.2.2. Future Session Hijacking:

This type of attack primarily targets the web server. An attacker typically takes control of the web server and uses it to initiate attacks on all subsequent legitimate user sessions. The intruder, for example, can eavesdrop, send spoofed responses, and/or drop user requests by hijacking other user sessions. Spoofing/Man-in-the-Middle attacks, Exhilaration attacks, Denial-of-Service/Packet Drop attacks, and Replay attacks are all forms of session hijacking attacks. The web request can, according to the mapping model, invoke certain database queries (e.g. Deterministic Mapping), after which the irregular condition can be identified [7].A traditional web server IDS or a database IDS, on the other hand, would not be able to detect such an assault on their own. Fortunately, our container-based web server architecture's isolation property can also avoid this type of assault. An intruder can never break into other users' sessions because each user's web requests are segregated into their own container.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 04, April-2021

### 3.2.3. Injection Attacks:

SQL injection attacks do not necessitate breaching the web server. Attackers can insert data or string content containing exploits into the web server logic, then use the web server to transmit these exploits to the back-end database. Even if the exploits are acknowledged by the web server, the relayed contents to the DB server will not be able to take on the expected structure for the provided web server request[8] because our approach offers a two-tier detection. Since the SQL injection attack alters the structure of SQL queries, even though the inserted data passes through the web server, it will produce SQL queries with a different structure that could be identified as a deviation from the SQL query structure that would usually accompany such a web request.

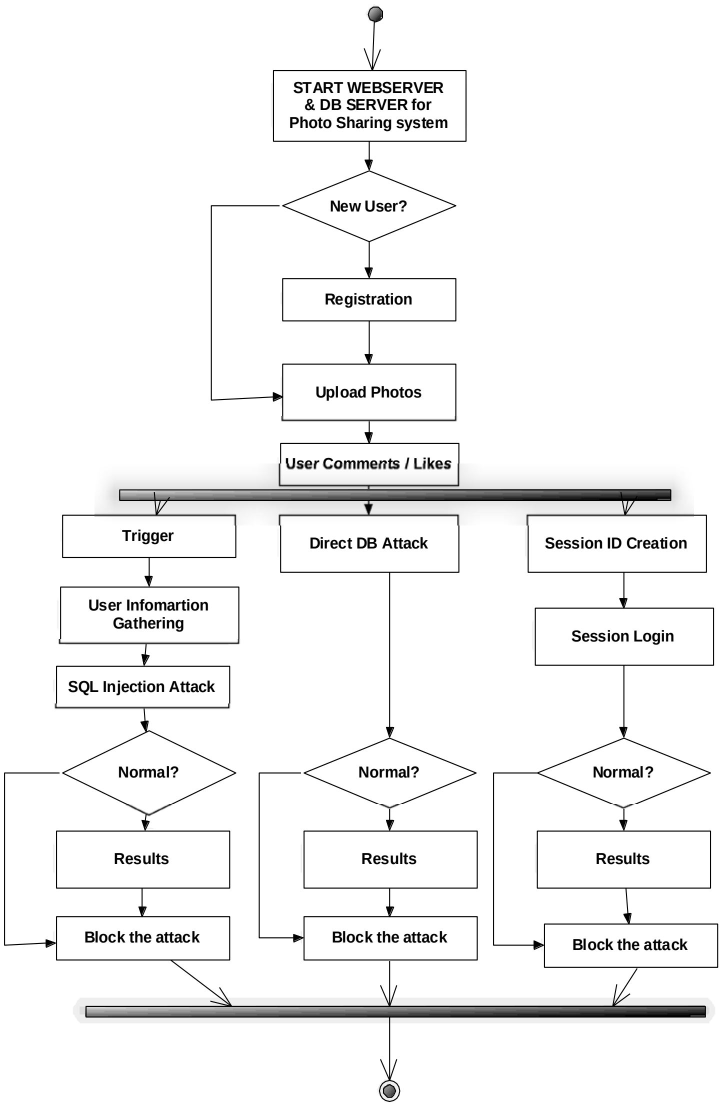

Figure 3: Flow chart of Hijack Future Session Attack

However, very little research has been done on multi tier Anomaly Detection (AD) systems that produce network behaviour models for both web and database network interactions. The back-end database server is often shielded behind a firewall in such multi tier architectures, while the web servers are often accessible remotely over the Internet. Unfortunately, although the back-end systems are shielded from direct remote attacks, they are vulnerable to attacks that use web requests to exploit the back end. Interface detection systems have been commonly used to detect known attacks by matching misused traffic patterns or signatures to protect multi tier web services. Unknown attacks can be detected using a form of IDS that uses machine learning to recognize irregular network traffic that differs from the so-called "standard" behaviour profiled during the IDS training process.

## 4. CONCLUDING THOUGHTS

We presented an interface detection framework that uses front-end web (HTTP) requests and back-end database (SQL) queries to create models of normal behavior for multi tier web applications. Double Guard creates container-based IDS with multiple input streams to produce alerts, unlike previous approaches that correlated or summarized alerts produced by independent IDSs.

Since the interface sensor has a more accurate normality model that identifies a wider range of hazards, we have shown that such correlation of input streams provides a better characterization of the device for anomaly detection. We did this by using lightweight virtualization to isolate the flow of information from each web server session. Furthermore, we tested our approach's detection accuracy by modelling static and dynamic web requests using the back-end file system and database queries. We created a well-correlated model for static websites, which proved to be successful in detecting various types of attacks in our tests.We also demonstrated that this was valid for dynamic queries, in which the web server front end is used for both retrieving information and updating the back-end database. Double Guard was able to detect a wide range of attacks with minimal false positives when we deployed our prototype on a device that included an Apache web server, a blog programme , and a My SQL back end. The number of false positives was, as predicted, proportional to the size and scope of the training sessions we used. Finally, we reduced false positives in complex web applications to 0.6 percent.

### 4.1Future Aims:

Double Guard is an intrusion detection system that models user session network behaviour in both the front-end web server and the back-end database. We can detect attacks that individual IDS wouldn't be able to detect by tracking both site and subsequent database requests. We also measure any multi tier IDS' weaknesses in terms of training sessions and feature coverage. We used an Apache web server with SQL and lightweight virtualization to implement Double Guard. We then received and analysed real-world traffic in both dynamic and static web applications over a 15-day span of device implementation. Finally, we were able to reveal a wide variety of attacks with 100% accuracy using Double Guard, with 0% false positives for static web services and 0.60% false positives for dynamic web services.
Anomaly detection and misuse detection are the two types of network interface detection systems (IDS). To detect abnormal changes or anomalous behaviours, the IDS must first define and describe the system's correct and appropriate static type and dynamic behaviour, which can then be used to detect abnormal changes or anomalous behaviours. The distinction between permissible and abnormal types of stored code and data is debatable.

## 5. REFERENCES:

[1] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of interface detection systems. Computer Networks, 31(8), 2011.
[2] V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna. Toward Automated Detection of Logic Vulnerabilities in Web Applications. In Proceedings of the USENIX Security Symposium, 2010.
[3] Y. Hu and B. Panda. A data mining approach for database interface detection. In H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock, editors, SAC. ACM, 2010
[4] Y. Huang, A. Stavrou, A. K. Ghosh, and S. Jajodia. Efficiently tracking application interactions using lightweight virtualization. In Proceedings of the 1st ACM workshop on Virtual machine security, 2013.
[5] H.-A. Kim and B. Karp. Autograph: Toward automated, distributed worm signature detection. In USENIX Security Symposium, 2010.
[6] C.Kruegel and G.Vigna. Anomaly detection of web-based attacks. In Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS '03), Washington, DC, Oct. 2011. ACM Press.
[7] Lee, Low, and Wong. Learning fingerprints for a database interface detection system. In ESORICS: European Symposium on Research in Computer Security. LNCS, Springer-Verlag, 2010.
[8] Liang and Sekar. Fast and automated generation of attack signatures: A basis for building self-protecting servers. In SIGSAC: 12th ACM Conference on Computer and Communications Security, 2013.
[9] J. Newsome, B. Karp, and D. X. Song. Polygraph: Automatically generating signatures for polymorphic worms. In IEEE Symposium on Security and Privacy. IEEE Computer Society, 2014.
[10] B. Parno, J.M. Mc Cune, D. Wendlandt, D. G. Andersen, and A. Perrig. CLAMP: Practical prevention of large-scale data leaks. In IEEE Symposium on Security and Privacy. IEEE Computer Society, 2009. [36] T. Pietraszek and C. V. Berghe. Defending against injection attacks through context-sensitive string evaluation. In RAID 2010.