Double Guard Detecting Intrusion Multi-Tier In Web Application

Rajeshwari.P[1], Swetha.A[2]
[1, 2] Dept of Software System
[1, 2] Sri Krishna Arts and Science College

Abstract- *One of the challenges facing computer systems is resisting attack and compromise in a networked environment. Today's computing environment is fairly homogeneous, due to a relatively small number of operating systems and application functions running on the vast majority of computers. This environment allows attackers to focus their efforts on the few types of systems deployed. Once an exploit is found, the exploit is effective against a very large number of systems running the same software. The large number of attack methods available on hacker Web sites demonstrates the ease with which attackers can exploit this homogeneous environment. This paper examines several widespread computer attacks to understand the effect of diversity on maintaining the integrity, and hence survivability, of information systems*

Keywords- Double guard, Multi-Tier web application, SQL Injection, Security

I. INTRODUCTION

In the contemporary digital landscape, safeguarding web applications stands as an imperative for organizations amidst the pervasive threat of cyberattacks. Multi-tier web applications, distinguished by their layered structure encompassing presentation, application logic, and data storage tiers, are particularly vulnerable to a spectrum of attacks due to their intricate architecture. To effectively fortify against such threats, organizations implement a multi-faceted security paradigm, with intrusion detection systems (IDS) assuming a pivotal role.

IDSs scrutinize network and system activities to identify malevolent behavior or deviations from established policies, alerting administrators or taking preemptive measures to repel attacks. Within the realm of multi-tier web applications, IDS integration emerges as a cornerstone defense mechanism against an array of threats, encompassing SQL injection, cross-site scripting (XSS), and distributed denial-of-service (DDoS) attacks.

Yet, conventional IDS setups may encounter constraints in accurately discerning and mitigating sophisticated attacks, particularly within dynamic and rapidly evolving threat landscapes. Enter the concept of double guard intrusion detection, wherein two complementary IDS systems are deployed each focusing on distinct security facets and employing disparate detection methodologies.

By amalgamating diverse IDS solutions, organizations augment their threat detection prowess, curtailing the probability of false positives and negatives. For instance, while one IDS specializes in signature-based detection—identifying known attack patterns—the other may leverage anomaly- based detection to pinpoint deviations from normal behavior.

This symbiosis between disparate approaches furnishes a more exhaustive and resilient defense against both recognized and unidentified threats. Furthermore, double guard intrusion detection empowers organizations with heightened adaptability and resilience against the ever-evolving threat panorama.

In this exposition, we will delve into the intricacies of multi-tier web application architecture and the operational dynamics of double guard intrusion detection, elucidating its pivotal role in bolstering organizational security postures amidst the contemporary digital milieu. Through a blend of meticulous analysis and pragmatic illustrations, our objective is to endow readers with the insights and acumen requisite for safeguarding web applications against malicious incursions, ensuring the sanctity and confidentiality of their data.

II. PROBLEM STATEMENT

EXISTING SYSTEM

This traditional model has shown vulnerabilities to data injection attacks. Our strategy involves the development of a dual- memory framework, which categorically divides the storage of code and data into separate spaces. Through this architectural innovation, any malicious code that is injected into the data compartment is effectively neutralized. This is because our system is designed to execute commands solely from the dedicated code compartment, preventing the

execution of any unauthorized code that might be inserted into the data space. This dual-memory approach aims to offer a robust defense against data injection threats, ensuring a safer computing environment by leveraging the division of memory spaces to enhance system security.

PROPOSED SYSTEM

This project outlines a novel system design aimed at enhancing security through the implementation of a dual-memory architecture. Traditional systems, which store executable code and data within a single memory space, are vulnerable to data injection attacks.

Our methodology introduces a concept of partitioned virtual memory, distinctly allocating code and data into separate segments. In the event of a data injection attack, the infiltrated malicious code, now residing in the data segment, is rendered ineffective. This is due to our system's design principle, which dictates that only the code segment is accessed for executing instructions, effectively isolating and neutralizing any malicious code injected into the data segment. Through this approach, the project seeks to fortify system defenses against data injection vulnerabilities by leveraging the separation of memory spaces for code and data.

III. METHODLOGY

Firewalls: Utilize firewalls to control incoming and outgoing traffic, employing stateful inspection for active connection monitoring and prevention of unauthorized access.

Intrusion Detection System (IDS): Deploy IDS to monitor network traffic for anomalies or suspicious patterns, using signature-based or behavior-based detection methods.

Virtual Private Network (VPN): Encourage VPN usage, especially for remote access, to encrypt data transmission and enhance overall network security.

Application-level Security:

Web Application Firewall (WAF): Implement WAF to protect against common web-based attacks like SQL injection, XSS, and CSRF, while also offering logging and monitoring functionalities.

Secure Coding Practices: Emphasize secure coding practices such as input validation, output encoding, and parameterized queries to mitigate vulnerabilities within the application code.

Regular Security Audits: Conduct routine security audits and penetration testing to identify and remediate application vulnerabilities.

Logging and Monitoring:

Centralized Logging: Set up centralized logging to aggregate logs from all application tiers, facilitating regular analysis for detection of suspicious activities.

Real-time Monitoring: Employ real-time monitoring solutions for anomaly detection, user behavior analytics, and SIEM tools to promptly identify and respond to security incidents.

Access Control and Authentication:

Role-based Access Control (RBAC): Implement RBAC to enforce least privilege access and restrict user permissions based on their roles.

Strong Authentication: Utilize strong authentication mechanisms like multi-factor authentication (MFA) to verify user identities and prevent unauthorized access. **Incident Response Plan:**

Develop and maintain an incident response plan outlining procedures for identifying, responding to, and mitigating security incidents effectively.

Conduct regular exercises and simulations to test the incident response plan and ensure personnel readiness during security incidents.

This comprehensive approach integrates network and application security measures to create a robust defense against intrusions in a multi-tier web application environment.

IV. RELATED WORK

Detecting intrusions in a multi-tier web application involves implementing robust security measures, with double-guard strategies being particularly effective. These strategies typically encompass various types of intrusion detection systems (IDS) and mechanisms to fortify the application's defense. Below are some related works and types of double-guard approaches used in detecting intrusions in multi-tier web applications:

Network-Based IDS (NIDS) and Host-Based IDS (HIDS): NIDS monitors network traffic for suspicious activities such as port scans or unusual data patterns, while HIDS focuses on individual hosts, analyzing system logs and configurations for

signs of intrusion. Integrating both NIDS and HIDS provides a comprehensive view of potential threats across the network and application layers.

Anomaly Detection Systems: These systems establish a baseline of normal behavior and flag any deviations as potential intrusions. Techniques such as machine learning algorithms can be employed to continually refine the baseline and adapt to evolving threats, enhancing the accuracy of anomaly

Signature-Based Detection: Signature- based compare incoming data packets or system activities against a database of know attack signatures. While effective against recognized threats, they may struggle with detecting novel or zero-day attacks. Complementing signature-based detection with behavior-based techniques improves the system's ability to identify emerging threats.

Application Layer Firewalls: Deploying firewalls at the application layer adds an additional layer of defense by inspecting and filtering HTTP/HTTPS traffic based on predefined rules. Combined with intrusion detection mechanisms, application layer firewalls can thwart attacks targeting specific web application vulnerabilities.

Log Analysis and Correlation: Analyzing logs generated by various components within the multi-tier architecture enables the detection of suspicious patterns or sequences of events indicative of an intrusion. Correlating logs from different layers enhances the accuracy of intrusion detection and aids in incident response.

Real-time Monitoring and Response: Utilizing real-time monitoring tools allows for immediate detection and response to potential intrusions. Automated response mechanisms, such as blocking suspicious IP addresses or triggering alerts for further investigation, bolster the overall security posture of the web application.

By integrating these double-guard strategies and leveraging advanced detection techniques,

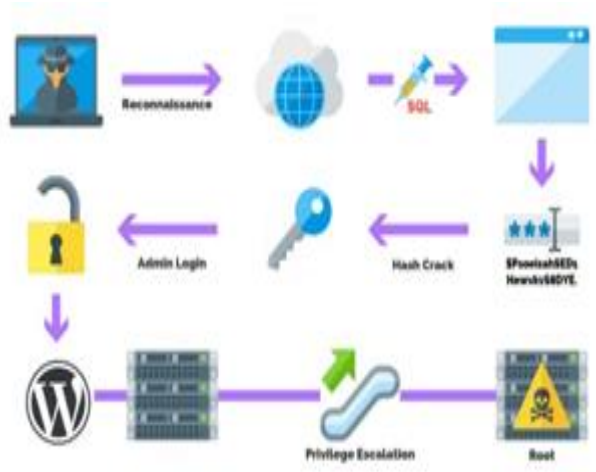
V. SYSTEM ARCHITECTURE

ATTACKSCENARIOS

PRIVILEGE ESCALATION ATTACK

A privilege escalation attack is a network intrusion tactic exploiting programming errors or design weaknesses to grant unauthorized users elevated access to network resources, including sensitive data and applications.

This type of attack typically occurs 6h5 their existing privileges to assume the identity of another user with comparable access levels, thereby gaining additional privileges. For instance, accessing another user's online banking account would exemplify horizontal privilege escalation. During such attacks, the aim is to access user accounts and associated data, potentially leading to further compromise and unauthorized actions within the network



(Fig.1.Privileges escalation attack)

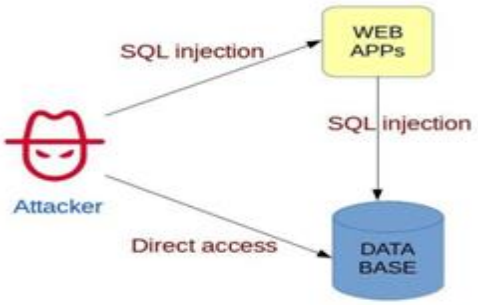
Privilege escalation attacks aim to exploit security flaws and vulnerabilities, enabling unauthorized access to networks, applications, and vital systems. These attacks come in two forms: vertical, where an intruder gains entry into an account to act as the user, and horizontal, where access to limited-permission accounts is obtained and privileges are escalated, typically to an administrator role, to carry out specific actions.

SQL INJECTION ATTACK

SQL injection represents a significant security flaw, enabling attackers to manipulate the database queries an application executes. This vulnerability can expose or alter data not intended for unauthorized access, including other users' information or sensitive internal data. In extreme cases, attackers can leverage this vulnerability to disrupt or take control of the underlying systems or execute denial-of- service attacks

A successful SQL injection exploit can lead to the disclosure of confidential information, such as user credentials, financial details, or personal data. This breach of security has been at the heart of numerous major data leaks, tarnishing the reputation of affected organizations and incurring legal penalties. Furthermore, attackers might secure

a lasting foothold within an organization's network, allowing them to persist undetected for long periods extending



(Fig.2.SQL injection attack)

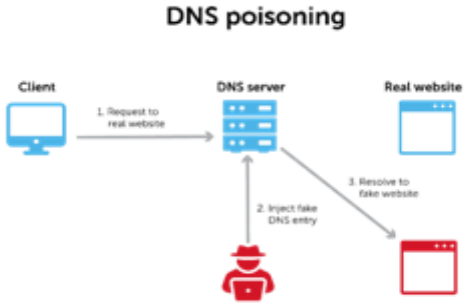
DNS REQUEST ATTACK

DNS filtering involves leveraging the Domain Name System to block access to malicious sites and filter out content deemed harmful or unsuitable, thus securing company data and controlling employee access on corporate networks. This method is a critical component of broader access control measures.

The Domain Name System (DNS) is pivotal for translating human-friendly domain names (e.g., cloudflare.com) into numerical IP addresses (e.g., 192.0.2.24), making it easier for users to navigate the internet without memorizing complex number sequences, similar to how phone contacts replace the need to remember everyone's number.

Accessing a website or web application begins with the device's search for the correct IP address through these steps:

The user inputs a domain name into their browser, prompting the device to initiate a DNS query, which it sends to a DNS resolver. The DNS resolver corresponding IP address by querying other DNS servers or referring to its cache. After identifying the matching IP address, the DNS resolver informs the user's device, a process known as "resolving" the domain. With the IP address received, the user's device connects to the server located at that address, starting the content loading process.



(Fig.3.DNS request attack)

DNS cache poisoning occurs when a malicious actor intercepts and responds to a DNS query with false information. This false information is then stored in the DNS resolver's cache for future use. It's worth noting that many DNS resolvers operate as caching resolvers, meaning they store DNS information temporarily to speed up subsequent queries and reduce network traffic. When a legitimate user subsequently requests the same domain name, the poisoned cache provides the false information, leading the user to a malicious website or server instead of the intended destination. This can result in various security risks, including phishing attacks, data theft, and malware distribution

VI. CONCLUSION

We 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. Unlike previous approaches that correlated or summarized alerts generated by independent IDSs, Double Guard forms a 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 to 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 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 system that employed Internet Information Services web server, a blog application, and a MSSQL backend, DoubleGuard 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. Finally, for dynamic web applications, we reduced the false positives.

REFERENCES

[1] http://www.sans.org/top-cyber-security-risks/.
[2] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-4332.
[3] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-4333.
[4] autobench. http://www.xenoclast.org/autobench/.
[5] Common vulnerabilities and exposures. http://www.cve.mitre.org/.
[6] Five common web application vulnerabilities. http://www.symantec. com/connect/articles/five-common-web-application-vulnerabilities.
[7] greensql. http://www.greensql.net/.
[8] C. Anley. Advanced sql injection in sql server applications. Technical report, Next Generation Security Software, Ltd, 2002.
[9] K. Bai, H. Wang, and P. Liu. Towards database firewalls. In DBSec2005.
[10] B. I. A. Barry and H. A. Chan. Syntax, and semantics-based signature database for hybrid intrusion detection systems. Security and Communication Networks, 2(6),