Intrusion Prevention System in Web Database and Web Application

B.Kiruthika[1], P. Priyanga[2] and A.Swathi[3]

[1]Assistant Professor, Department of Computer Science and Engineering
SNS College of Engineering, Coimbatore-641107

[2, 3] B.E. Computer Science and Engineering
SNS College of Engineering, Coimbatore-641107

Abstract

A weak point in network-based applications is that they commonly open some known communication port(s), making themselves targets for denial of service (DoS) attacks. Considering adversaries that can eavesdrop and launch directed DoS attacks to the applications' open ports, solutions based on pseudo-random port-hopping. As port-hopping needs that the communicating parties hop in a synchronized manner, these solutions suggest acknowledgment-based protocols between a client-server pair or assume the presence of synchronized clocks. Acknowledgments, if lost, can cause a port to be open for a longer time and thus be vulnerable to DoS attacks; Time servers for synchronizing clocks can become targets to DoS attack them. The proposed system uses BIGWHEEL Algorithm, for servers to communicate with multiple clients in a port-hopping manner, thus enabling support to multi-party applications as well. The algorithm does not rely on the server having a fixed port open in the beginning, neither does it require from the client to get a first-contact port from a third party. We also present an adaptive algorithm, HOPERAA, for hopping in the presence of clock-drift, as well as the analysis and evaluation of the methods. The solutions are simple, based on each client interacting with the server independently of the other clients, without the need of acknowledgments or time server. Provided that one has an estimation of the time it takes for the adversary to detect that a port is open and launch an attack, the method we propose does not make it possible to the eavesdropping adversary to launch an attack directed to the application's open port(s).The proposed system protects 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 an independent ID would not be able to identify.

***Keywords**: Intrusion detection, database, attacks.*

1. Introduction

Web-delivered services and applications have increased in both popularity and complexity over the past few years. Daily tasks, such as banking, travel, and social networking, are all done via the web. Such services typically employ a web server front-end that runs the application user interface logic, as well as a back-end server that consists of a database or file server. Due to their ubiquitous use for personal and/or corporate data, web services have always been the target of attacks. These attacks have recently become more diverse, as attention has shifted from attacking the front-end to exploiting vulnerabilities of the web applications [1],[8] ,in order to corrupt the back-end database system (e.g., SQL injection attacks). A network Intrusion Detection System (IDS) can be classified into two types: anomaly detection and misuse detection. The system proposed composes both web IDS and database IDS to achieve more accurate detection, and it also uses a reverse HTTP proxy to maintain a reduced level of service in the presence of false positives. in our Double Guard, we utilized the container ID to separate session traffic as a way of extracting and identifying causal relationships between web server requests and database query events.

2. Related work

Web-based vulnerabilities [2] represent a substantial portion of the security exposures of computer networks. In order to detect known web-based attacks, misuse detection systems are equipped with a large number of signatures. Web-based attacks can take many forms, with effects ranging from mild inconvenience when a Web site no longer loads properly to complete compromise of the user's system [2]. Unfortunately, it is difficult to keep up with the daily disclosure of web-related vulnerabilities, and, in addition, vulnerabilities may be introduced by installation-specific web-based applications. The system correlates [3] the server- side programs referenced by client queries with the parameters contained in these queries. The application-specific characteristics of the parameters allow the system to perform focused analysis and produce a reduced number of false positives.

Intrusion detection systems aim at detecting attacks against computer systems and networks or against information systems in general as it is difficult to provide provably secure information systems and maintain them in such a secure state for their entire lifetime and for every utilization Sometimes legacy or operational con strains do not even allow a fully secure information system to be realized at all. We introduce a taxonomy of intrusion detection systems [5] that highlights the various aspects of this area.

Most Web sites today add dynamic content to a Web page making the experience for the user more enjoyable. Dynamic content is content generated by some server process, which when delivered can behave and display differently to the user depending upon their settings and needs.[4] Cross-site scripting is gaining popularity among attackers as an easy exposure to find in Web sites. Cross-site scripting flaws[6] have now surpassed buffer over flows as the world's most common publicly-reported security vulnerability. In recent years, browser vendors and researchers have tried to develop client-side filters to mitigate these attacks. We have contributed an implementation of our filter design to the Web Kit open source rendering engine, and the filter is now enabled by default in the Google Chrome browser.

Large-scale attacks, such as those launched by worms and zombie farms, pose a serious threat to our network-centric society. Existing approaches such as software patches are simply unable to cope with the volume and speed with which new vulnerabilities are being discovered. The approach [8], called COVERS, uses a forensic analysis of a victim server’s memory to correlate attacks to inputs received over the network, and automatically develop a signature that characterizes inputs that carry attacks.

3. Existing System

The web server acts as the front-end, with the file and database servers as the content storage back-end. All network traffic, from both legitimate users and adversaries, is received intermixed at the same web server. If an attacker compromises the web server, he/she can potentially affect all future sessions,[1]. Anomaly Detection (AD) systems that generate models of network behavior for both web and database network interactions [3].The communication between the web server and the database server is not separated, and hardly understand the relationships among them. Both the wed and the database are vulnerable. Attacks are come from the web clients. Attackers may strike the database server through the web server or, more directly, by submitting SQL queries Information can not be reached to the authorized user from the Sending user Data Base Information cannot be safe in Multitier application. All communication from client to database is separated by a session. Each session assigned by a unique ID. Low Secure authentication and content security. Static model building algorithm is used. In the case of a static website, the nondeterministic mapping does not exist as there are no available input variables or states for static content.

4. Attack Scenario

4.1 SQL Injection Attack

SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. The primary form of SQL injection consists of direct insertion of code into user-input variables that are concatenated with SQL commands and executed. The injection process [1] works by prematurely terminating a text string and appending a new command. Because the inserted command may have additional strings appended to it before it is executed, the malefactor terminates the injected string with a comment mark "--". Subsequent text is ignored at execution time.

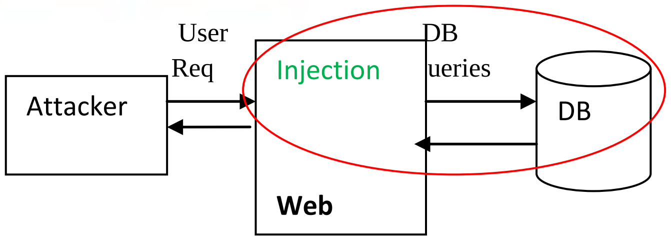

Fig: 1 SQL Injection Attack

Many web pages take input from users, such as search terms, feedback comments or username and password, and use them to build a SQL query which is passed to the database. If these inputs are not validated, there is nothing to stop an attacker inputting malicious code[9].The problem of SQL injection is with vulnerable web application not the web server or services running in the operating system. This vulnerability allows the attacker to gain complete access to underlying database.

4.2 Privilege Escalation Attack

A privilege escalation attack is a type of network intrusion that takes advantage of programming errors or design flaws to grant the attacker elevated access to the network and its associated data and applications. Not every system hack will initially provide an unauthorized user with full access to the targeted system. In those circumstances privilege escalation is required. There are two kinds of privilege escalation: vertical and horizontal [10]. Privilege escalation means a user receives privileges they are not entitled to. These privileges can be used to delete files, view private information, or install unwanted programs such as viruses. It usually occurs when a system has a bug that allows security to be bypassed. The result is that an application with more privileges than intended by the application developer or system administrator can perform unauthorized actions.

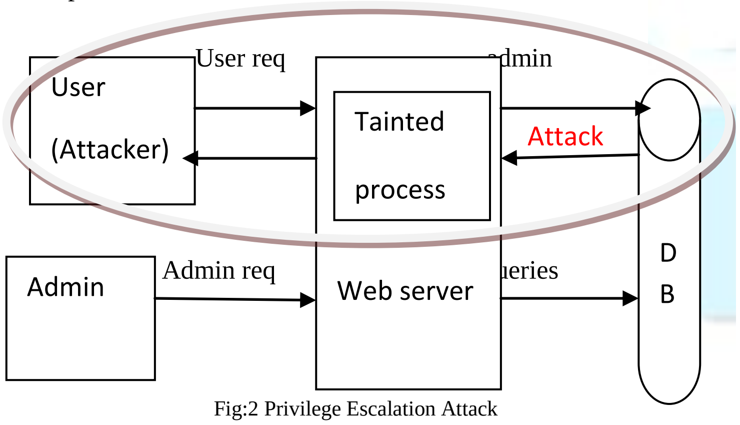

Fig:2 Privilege Escalation Attack

The degree of escalation depends on which privileges the attacker is authorized to possess, and which privileges can be obtained in a successful exploit. For example, a programming error that allows a user to gain extra privilege after successful authentication limits the degree of escalation, because the user is already authorized to hold some privilege. Likewise, a remote attacker gaining super user privilege without any authentication presents a greater degree of escalation.

4.3 Hijack Session Attack

Session hijacking is the exploitation of a valid computer session—sometimes also called a *session key*—to gain unauthorized access to information or services in a computer system. In particular, it is used to refer to the theft of a magic cookie used to authenticate a user to a remote server. It has particular relevance to web developers, as the HTTP cookies used to maintain a session on many web sites can be easily stolen by an attacker using an intermediary computer or with access to the saved cookies on the victim's compute. The Session Hijacking attack compromises the session token by stealing or predicting a valid session token to gain unauthorized access to the Web Server. The most important information in session attack is to obtain a valid valid session identifier (SID). There are three common methods used to obtain a valid session identifier [11].

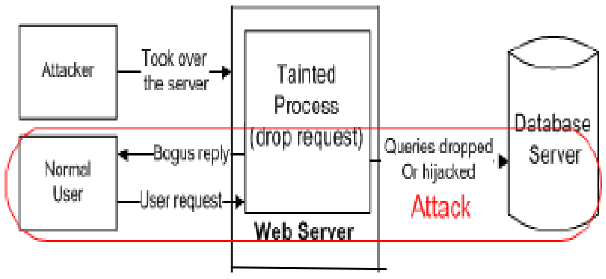

Fig: 3 Hijack Session Attack

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. For instance, by hijacking other user sessions, the attacker can eavesdrop, send spoofed replies, and/or drop user requests. A session hijacking attack can be further categorized as a Spoofing/Man-in-the-Middle attack, an Ex-filtration Attack, a Denial-of-Service/Packet Drop attack, or a Replay attack. According to the mapping model [1], the web request should invoke some database queries.

4.4 Direct DB Attack

An attacker could also have already taken over the web server and be submitting such queries from the web server without sending web requests. Without matched web requests for such queries, a web server IDS could detect neither. Furthermore, if these DB queries were within the set of allowed queries, then the database IDS itself would not detect it either. However, this type of attack can be caught with our approach since we cannot match any web requests with these queries.

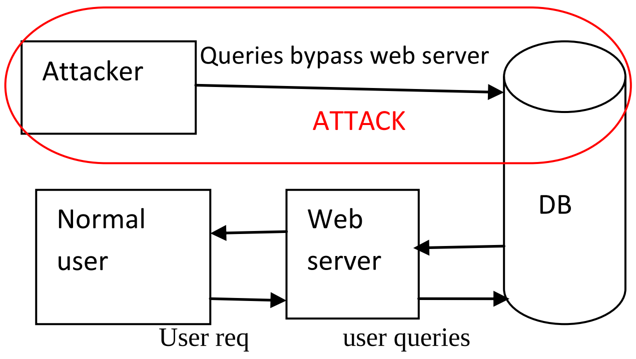

Fig: 4 Direct DB Attack

Vulnerabilities in database protocols may allow unauthorized data access, corruption or availability. For example, the SQL Slammer worm took advantage of a Microsoft SQL Server protocol vulnerability to execute attack code on target database servers. Protocol attacks can be defeated by parsing and validating SQL communications to make sure they are not malformed.

5 Proposed System

Some approaches have detected intrusions or vulnerabilities by statically analyzing the source code and executables. Others dynamically track the information flow to understand propagations and detect intrusions. In double guard the new container based web server architecture [1] enables us to separate the different information flows by each session by using lightweight virtualization. The proposed system is well correlated model that provides an effective mechanism to detect different types of attack, the proposed system implements two algorithms. HOPERAA (Hopping-Period -Align and -Adjust Algorithm) which is an adaptive algorithm, executed by each client when its hopping period length and alignment drift apart from the server's. For enabling multi-party communication with port-hopping, we present the BIGWHEEL algorithm for a server to support hopping with many clients, without the server needing to keep state for each client individually. Detects the attacks in multi tier web services. It protects both the front end and back end. The solutions are simple, based on each client interacting with the server independently of the other clients, without the need of acknowledgments or time server(s). Further, they do not rely on the application having a fixed port open in the beginning; neither do they require the clients to get a "first-contact" port from a third party. We show analytically the properties of the algorithms and also study experimentally their success rates, confirm the relation with the analytical bounds provided that one has an estimation of the time it takes for the adversary to detect that a port is open and launch an attack, the method we propose does not make it possible to the eavesdropping adversary to launch an attack directed to the application open ports. Using Intrusion Detection Technique Easily can find out the attacks in multitier web services Detects the attacks in multi tier web services. It protects both the front end and back end

6 Experimental Evaluation

This section presents results from several this systems working scenarios which can illustrate further some important properties of this project. The first experiment shows the contact-initiation phase. We choose Clients 8, and time intervals 140, which translates to 140 intervals in the port space and each one having 1024 ports that can be used. Then we vary the strength of the adversary from Q = 10000 to Q = 50000, and for each value of Q we let the client perform 50 repetitions of the contact-initiation phase, and then record the number of trials of each contact-initiation phase. We computed the average number of trials that a client has to perform over all these contact-initiation phases. Figure 1 shows both the experimental outcome and the upper bound of the expectation computed in corollary 1. The average number of trials grows with Q, but we can see that even for Q = 50000, the average number of trials during the contact-initiation phase is still not high (7.2 trials).

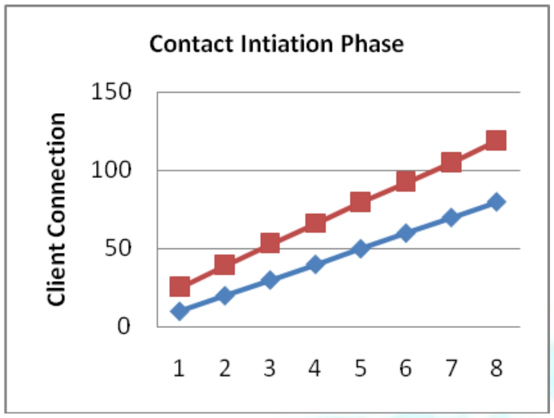

Fig:1Shows the Client connection and contacts initialized for the client with averages ports

In the second experiment we study how the HOPERAA execution interval grows (i.e. the protocol overhead decreases) as a function of the number of executions of the HOPERAA algorithm under different values of clock-rate drift. As shown in Figure 2 and Figure 3,

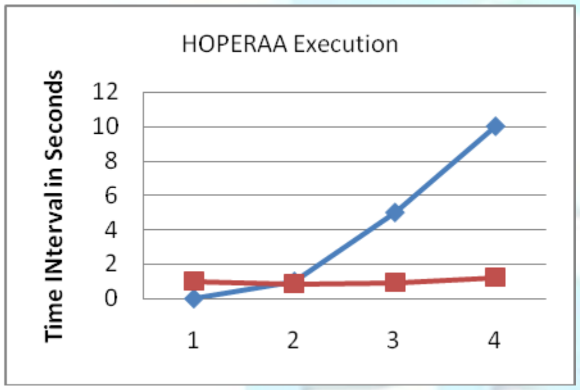

Fig: 2 Shows the time interval for each client

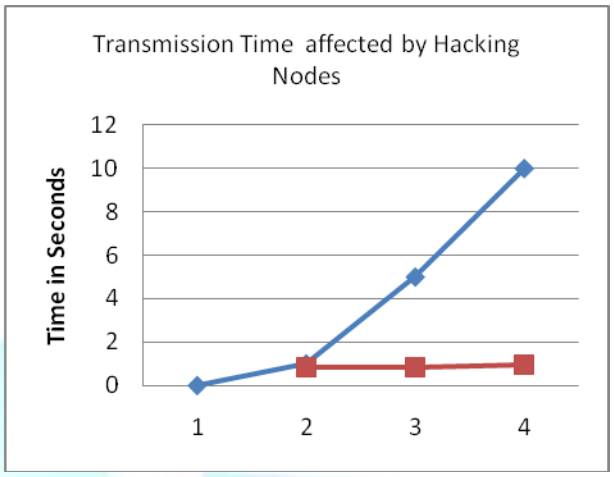

Fig: 3 Shows the transmission effected when the hack client appears

7 Conclusion

Double Guard presented an intrusion detection system that builds models of normal behavior web application from both front end web request and back end database queries. Double Guard forms a container based IDS with multiple input stream to produce alerts. The attacks discussed throughout are representative of growing and increasingly problematic class of vulnerabilities. The connectivity between the Internet and traditional networks introduces new avenues for exploit: once confined to exploiting only inert hosts, remote adversaries can debilitate the services we depend on to carry on our daily lives. In a broader sense, the ability to control the physical world via the Internet is inherently dangerous, and more so when the affected components are part of critical infrastructure. This work provides some preliminary solutions and analysis for these vulnerabilities. Essential future work will seek more general solutions that address these vulnerabilities in current and next generation networks.

Acknowledgement

The authors would like to thank Director cum secretary, Correspondent and Principal of SNS College of Engineering, Coimbatore for their motivation and constant encouragement. The authors would like to thank the faculty Members of

Department of Computer and Engineering for the critical review of the manuscript and for their valuable input and fruitful discussions. Also we take privilege in extending gratitude to their family members and friends who rendered their support throughout this research work.

References

1. Angelos Stavrou, Brent ByungHoon, Meixing Le "DoubleGuard: Detecting Intrusions in Multi-tier Web Applications".

2. Web-based vulnerabilities http://www.computerworld.com.au/article/213510/web-based_vulnerabilities_worsening

3. 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. 2003.ACM Press.

4. Cross site script "http:// www. ibm. com/developerworks/tivoli/library/s-csscript"

5. H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusiondetection systems.Computer Networks, 31(8), 1999.

6. XSS flaws" https:// www.owasp.org /index.php/Cross_Site_Scripting_Flaw

7. D. Bates, A. Barth, and C. Jackson. Regular expressions considered harmful in client-side xss filters. In Proceedings of the 19th international conference on World wide web, 2010.

8. Security controls "http:// www.sans .org/critical-security-controls/"

9. SQL injection dectecion tools computer weekly tech target" http://www.computerweekly.com/tip/SQL-injection-detection-tools-and-prevention-strategies"

10. Tech target "http:// searchsecurity .techtarget.com/definition/privilege-escalation-attack .

11. Hundreg-usefulinformation"http:// hungred.com/useful-information/solutions-session-attacks/#sthash.PQQJ4WWM.dpuf.