Enhanced Security Approach for Detecting Intrusions in Multitier Web Applications

Ashwini R Pawar, S.S. Bhardwaj,, Sachin N. Wandre
Sinhgad Institute of Technology,,Lonavala,Pune,Maharashtra,India

Abstract— ***In our daily life, Internet services and applications are playing an important role, such as enabling the personal information management and communication from anywhere. 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, to accommodate this increase in application and data complexity. Replica the network performance of user sessions across both the front-end webserver and the back-end database, an IDS system Double Guard is approached in this paper. We are able to search out attacks that self-governing IDS would not be able to discern, by checking both web and subsequent database requests. Likewise, in stipulations of functionality, reporting and training sessions, we enumerate the limitations of any multitier. With the help of lightweight virtualization and an Apache web server with MySQL, we implemented DoubleGuard.We then framed and processed real-world traffic over a 15-day period of system deployment in both stable and dynamic network applications. Lastly, we were able to depict a broad range of attacks with 100 percent accuracy while maintaining 0 percent false positives for static web services and 0.6 percent false positives for dynamic web services with the help of DoubleGuard.***

Index Terms— Multitier web application, virtualization, anomaly detection.

I. INTRODUCTION

Over the past few years, web delivered services and applications are increasing in both complexity and popularity. Social networking, banking, travel like daily task are done with the help of web.This type of services are generally working 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. Web services are forever being the target of attacks due to their ever-present use of personal and corporate data. In order to corrupt the back-end database system [1], these attacks have freshly become more varied, as concentration has moved from attacking the front end to developing vulnerabilities of the web applications [2], [3], [4]. Recently, overabundances of Intrusion Detection Systems (IDSs) inspect network packets individually within both the web server and the database system. Though, the generative models of network behavior for both web and database network interaction done by performing very small work on multitier Anomaly Detection that is AD systems. When the webservers are remotely accessible over the internet, the back-end database server is frequently protected behind a firewall in such multitier architecture.

By corresponding distorted traffic patterns or signatures [5], [6], [7], [8], Intrusion detection system extensively used to detect known attacks to protect multitier web services. The database IDS and the web IDS also detect the abnormal network traffic individually, which is sent to either of them. During the IDS training phase, 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. But we observe that these IDSs cannot able to detect the cases where in normal traffic is used to attack the webserver and the database server. If the database IDS recognize that a privileged request from the web server which is not connected with the user-privileged access then this type of attack can be willingly detected. Regrettably, within the current multithreaded web server architecture, it is not practicable to detect or profile such causal mapping between web server traffic and DB server traffic since traffic cannot be evidently attributed to user sessions.

Double Guard, a system which is used to detect attacks in multitier web services is presented in this paper. Normality models of isolated user sessions that include both the web front-end (HTTP) and back-end (File or SQL) network transactions are created is approach here. We attain this by utilizing a lightweight virtualization technique to allocate each user's web session to a devoted container, an isolated virtual computing environment. With the help of container ID, we precisely relate the web request with the subsequent DB queries. By taking both the DB and web server traffic into account, DoubleGuard can build a causal mapping profile.

Using OpenVZ, DoubleGuard container is implemented. There is almost no overhead in comparison to an unprotected

Manuscript received Nov, 2014.
Ashwini R. Pawar, Department of Electronics and telecommunication Engineering, University of Pune Sinhgad Institute of Technology,Lonavala,Pune,Maharashtra,India

S.S. Bhardwaj Department of Electronics and telecommunication Engineering, University of Pune Sinhgad Institute of Technology,Lonavala,Pune,Maharashtra,India

Sachin N. Wandre, Department of Computer Engineering, University of Pune Sinhgad Institute of Technology,Lonavala,Pune,Maharashtra,India

vanilla system, when the request rate is moderate. It has reasonable performance overhead and is practical for most web techniques are proved by the performance testing. The container-based web structural design not only promotes the outlining of causal mapping, but it also offers an isolation that averts future session-hijacking attacks. As transient containers can be easily instantiated and destroyed, we allocated each client session a devoted container so that, even when an attacker may be able to compromise a single session, the damage is restricted to the compromised session and other user sessions remain unaffected by it. We ran a lot of copies of the web server examples in dissimilar containers so that each one was remote from the rest, in lightweight virtualization environment.

We demonstrate that, for websites that do not authorize content alteration from users, there is a direct fundamental relationship between the requests received by the front end web server and those produced for the database back end, using the prototype. The causality - mapping model can be generated precisely and without previous knowledge of web application functionality also shown here. Using real-world network traffic obtained from the web and database requests of a large center demonstrate that we were able to extract 100 percent of functional mapping by using as few as 35 sessions in the training phase, are evaluated in our experiment. It does not depend on content changes, but it depends on the size and functionality of web server and applications. These sites are static, they do not change over a time, but they are in forbidden fashion so that they permit the changes to spread to the sites' normality models.

Our ability to model the causal relationship between the front end and back end is not forever deterministic and depends primarily upon the application logic because there are web services that authorize unrelenting back-end data alterations. These services, which we call dynamic, let HTTP requests to comprise parameters that are changeable and depend on user input. We initial y generated an individual training model for the basic processes offered by the web services, to deal with this challenge while structuring a mapping model for dynamic web pages.

In this paper, we discuss about a related work in section II, system architecture that is proposed work, algorithms in section III, performance evaluation in section IV and at the last Conclusion in section V.

I. RELATED WORK

Misuse detection and anomaly detection are the two types of Intrusion Detection System. The IDS to describe and typify the accurate and satisfactory static form and dynamic behavior of the system, which can then be used to detect abnormal changes or irregular behaviors [9], [10],is the initial needs of the Anomaly detection. Behavior models are constructed by performing a statistical analysis on historical data [11], [12],[13] or by using rule-based approaches to identify behavior patterns [14]. The boundary flanked by satisfactory and irregular forms of stored code and data is exactly defined. An anomaly detector then compares real usage patterns against recognized models to recognize abnormal proceedings. We depend on a training phase to build the correct model, and our detection approach belongs to anomaly detection. There are numerous of approaches [15] that are annoying to resolve the difficulty where lawful updates may cause the model float.

A compilation of mechanism that transforms intrusion detection sensor attentive into concise intrusion information in order to decrease the number of replicated alerts, false positives, and non-relevant positives. With the goal of producing a concise impression of security connected with action on the network, it also combines the attentive from dissimilar levels recitation a single attack. It focuses primarily on theoretical the low-level sensor alerts and providing compound, logical, high-level alert proceedings to the users. DoubleGuard functions on manifold nourishes of network traffic using single IDS that seems across sessions to create an alert without associating or abbreviation the alerts produced by other self-governing IDSs.That's why DoubleGuard differs from this type of approach that associates alerts from self-governing IDSs.

To detect intrusions, such IDS also uses the temporal information. DoubleGuard does not have such a restraint as it uses the container ID for each session to causally map the connected events, whether they are simultaneous or not. It does not associate proceedings on a time basis, which runs the risk of incorrectly considering self-governing but simultaneous events as connected events.

The database receives the highest level of protection because it always contains more valuable information. Due this reason most of the research efforts have been made on database IDS and database firewalls .This software like Green SQL [7], work as a reverse proxy for database connections. Web applications will primarily connect to a database firewall, Instead of connecting to a database server. Then SQL queries are analyzed, they are forwarded to the back-end database server if they are deemed safe. The system which is proposed in creates both web IDS and database IDS to attain more precise detection, and it also uses a reverse HTTP proxy to uphold a reduced level of service in the presence of false positives.

By statically analyzing the source code or executable some preceding approaches have detected intrusions or vulnerabilities. Others [11], [14] dynamically track the information flow to appreciate spoil propagations and detect intrusions. The new container-based webserver structural design allows us to divide the dissimilar information flows by each session, in DoubleGuard. To examine the source code or know the application logic in our approach. DoubleGuard approach does not need application logic for structure a model. Also for dynamic web services, we haven't required full application logic and not need to know the basic user processes in order to normal behavior.

To prevent or detect Cross Site Scripting that is XSS injection attacks [13] or SQL injection attacks, a validating input is useful. And we observe that, by taking the arrangements of web requests and database queries without seeming into the values of input parameters, DoubleGuard can detect SQL injection attacks.

To improve security, performance and to isolate the object, a virtualization is used. Lightweight virtualizations, such as OpenVZ [14], Parallels Virtuozzo or Linux-VServer [11] are the alternatives because para-virtualization and full virtualization are not only the approaches being taken .Light weight containers can have substantial presentation advantages over full virtualization or para-virtualization. There are also some desktop systems that use lightweight virtualization to isolate different application instances. In our DoubleGuard, we utilized the container ID to separate session traffic as a way to take out and recognizing causal relationships between webserver requests and database query events.

For preventing data leaks even in the presence of attacks, CLAMP is architecture is used. CLAMP assurances that a user's responsive data can only be accessed by code running on behalf of different users, By dividing code at the webserver layer and datain the database layer by users. DoubleGuard focuses on replicating the mapping patterns between HTTP requests and DB queries to detect malicious user sessions. DoubleGuard uses process isolation, whereas CLAMP requires platform virtualization, and CLAMP offers more coarse-grained isolation than DoubleGuard. However, DoubleGuard would be unproductive at detecting attacks if it were to use the coarse-grained isolation as used in CLAMP.

II. SYSTEM ARCHITECTURE

A. Threat Model

To include the assumptions and types of attacks to protect against with the help of threat model. We assume that both the web and the database servers are susceptible. Attacks are network stand and come from the web clients, they can open application layer attacks to cooperation the webservers they are connecting to. Assume that the attacks can neither be detected nor prevented by the current webserver, IDS, that attacker may take over the webserver after the attack, and that later they can get full control of the webserver to launch subsequent attacks. We also assume that the database server will not be totally taken over by the attackers. Attackers may hit the database server through the webserver or, more directly, by submitting SQL queries, they may get and contaminate responsive data within the database. We assume that no attack would happen during the training phase and model building, no prior knowledge of the source code or the application logic of web services deployed on the web server, analyzing only network traffic that reaches the web server and database.

We design, lightweight process containers for client sessions. Were thousands of containers can be initialized on a single physical machine. Our approach dynamically produces new containers and recycles used ones. As a result, a single physical server can run incessantly and serve all web requests. However, from a logical viewpoint, each session is allocated to a dedicated webserver and isolated from other sessions. We can assure that each session will be served with a clean webserver example, at initialization, we initialize each virtualized container using a read-only spotless template. We choose to separate communications at the session level so that a single user always contrasts with the same webserver. Sessions can stand for different users to some degree, and we wait for the communication of a single user to go to the same devoted webserver, thereby permitting us to recognize suspect behavior by both session and user. We will treat all traffic within this session as tainted, if we detect abnormal behavior in a session. If attacker cooperation's a vanilla webserver, other sessions' communications can also be hijacked. In our system, an attacker can only wait within the webserver containers that he/she is connected to, with no knowledge of the survival of other session communications. We can thus make sure that lawful sessions will not be compromised directly by an attacker.

B. Building the Normality Model

This container-based and session-separated webserver architecture not only improves the security performances, but also gives us with the remote information flows that are alienated in each container session. It lets us recognize the mapping between the webserver requests and the succeeding DB queries, and to exploit such a mapping model to detect abnormal behaviors on a session/client level.

C. Attack Scenarios

Following types of attacks are effectively captured in our system.

- Privilege Escalation Attack
- Hijack Future Session Attack
- Injection Attack
- Direct DB Attack

D. DoubleGuard Limitations

DoubleGuard have a following detection and operational limitations.

1. Vulnerabilities due to improper input processing.
2. Possibility of evading DoubleGuard.
3. Distributed DoS.

E. MODELING DETERMINISTIC MAPPING AND PATTERNS

Here, we classify the four mapping patterns. We treat each request as the mapping source because the request is at the origin of the data flow.

1. Deterministic Mapping
2. Empty Query Set
3. No Matched Request
4. Nondeterministic Mapping.

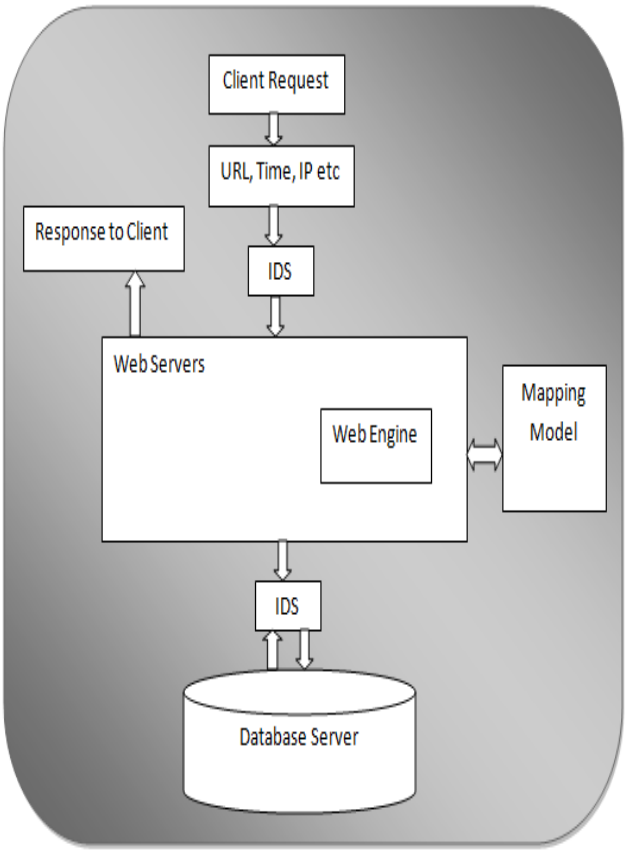

Fig.:The overall architecture of our prototype.

Algorithm
Step1: Start
Step 2: for i=0 to n
Step 3: Analyse the ith request and extract the request attributes Ra i
Step 4: Intrusion Detection System(IDS Level 1)
Step 5: Compare Request Ra i with mapping model database and signature of vaid request
Step 6: if request is valide then
Go to 8
else
Drop the request

Step 8: Extract query Qi from request Req i
Step 9: Intrusion Detection System(IDS Level 2)
Step 10: Compare query Qi with mapping model databes and legal query databes
Step 11: if request is valide then
Go to 12
else
Drop the request
Step 12: Process the Request and generate the response
Step 13: Go to step 2 if new Request is identified
Step 14: End

III. PERFORMANCE EVALUATION

Using a web server with a back-end DB, we implement a prototype of DoubleGuard. By analyzing four classes of attacks, we evaluate the detection result. Our design decision is select to assign each session into a different container, in our prototype. Containers were recycled based on events or when sessions time out, in our implementation. We were able to use the same session tracking mechanisms as implemented by the Apache server because lightweight virtualization containers do not impose high memory and storage overhead. We discuss performance overhead, which is common for both static and dynamic models, in the following section. In our analysis, we did not take into consideration the potential for caching expensive requests to further reduce the end-to-end latency; this we left for future study. We perform on the Static website model in training phase as well as dynamic modeling detection rates. We also perform on the attack detection, which is mentioned in section III.

IV. CONCLUSION

In this paper, we projected an intrusion detection system which is built the models of normal behavior for multitier web applications from both front-end web (HTTP) requests and back-end database (SQL) queries. Different from preceding approaches that connected or summarized alerts produced by self-governing IDSs, DoubleGuard form's container-based IDS with manifold input streams to create alerts. We have exposed that such association of input streams give same improved description of the system for anomaly detection because the intrusion sensor has a more accurate normality model that detects a wider range of intimidation.

We attain this by separating the flow of information from each webserver session with a lightweight virtualization. When our effort to model static and dynamic web requests with the back-end file system and database queries, we enumerate the detection correctness of our approach. For static websites, we built a well-correlated model, which our experiments showed to be effectual at detecting dissimilar types of attacks. For dynamic requests where both recoveries of information and updates to the back-end database happen using the webserver front end. DoubleGuard was able to recognize a wide range of attacks with minimal false positives.

REFERENCES

[1] A. Schulman, "Top 10 Database Attacks," http://www.bcs.org/server.php?show=ConWebDoc.8852, 2011.
[2] "Five Common Web Application Vulnerabilities," http://www.symantec.com/connect/articles/five-common-web-applicationvulnerabilities,2011.
[3] "Common Vulnerabilities and Exposures," http://www.cve. mitre. org/, 2011.
[4] SANS, "The Top Cyber Security Risks," http://www.sans.org/top-cyber-security-risks/, 2011.
[5] J. Newsome, B. Karp, and D.X. Song, "Polygraph: Automatically Generating Signatures for Polymorphic Worms," Proc. IEEE Symp.Security and Privacy, 2005.
[6] H. -A. Kim and B. Karp, "Autograph: Toward AutomatedDistributed Worm Signature Detection," Proc. USENIX SecuritySymp. 2004.
[7] Liang and Sekar, "Fast and Automated Generation of AttackSignatures: A Basis for Building Self-Protecting Servers," SIGSAC: Proc. 12th ACM Conf. Computer and Comm. Security, 2005.

[8] B.I.A. Barry and H.A. Chan, "Syntax, and Semantics-BasedSignature Database for Hybrid Intrusion Detection Systems, "Security and Comm. Networks, vol. 2, no. 6, pp. 457-475, 2009.
[9] H. Debar, M. Dacier, and A. Wespi, "Towards Taxonomy ofIntrusion-Detection Systems," Computer Networks, vol. 31, no. 9, pp. 805-822, 1999.
[10] T. Verwoerd and R. Hunt, "Intrusion Detection Techniques and Approaches," Computer Comm., vol. 25, no. 15, pp. 1356-1365, 2002.
[11] C. Kruegel, and G. Vigna, "Anomaly Detection of Web-Based Attacks," Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03), Oct. 2003.
[12] G. Vigna, W.K. Robertson, V. Kher, and R.A. Kemmerer, "AStateful Intrusion Detection System for World-Wide Web Servers,"Proc. Ann. Computer Security Applications Conf. (ACSAC '03), 2003.
[13] 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.
[14] M. Roesch, "Snort, Intrusion Detection System," http://www. snort.org, 2011.

