ISSN: 2321-7782 (Online)
Volume 2, Issue 12, December 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Review of Double Guard: Detecting and Preventing Intrusions in Multi-tier Web Applications

Ekta Naik[1]
M.E. Computer Engineering
D. Y. Patil School of Engineering and Technology,
Lohagaon, Chroli.
Pune, Maharashtra, India

Ramesh Kagalkar[2]
Assistance Professor.
D. Y. Patil School of Engineering and Technology,
Lohagaon, Chroli.
Pune, Maharashtra, India.

Abstract: ***In today's world we have many issues in internet security and privacy. We use internet in travelling, social media, banking, study etc. But we often face the problems with the privacy of the network system. To accommodate this increase in application and data complexity, web services have moved to a multi-tiered design wherein the web server runs the application front-end logic and data is outsourced to a database or file server. IDS play a key role in computer security technique. But it also has drawbacks of its own. To overcome those drawbacks Double guard technique is introduced.***

I implemented double guard using IIS (internet information and service manager), an in built web server of windows 7 ultimate, with My SQL. This paper presents Double Guard, an IDS system that models the network behaviour of user sessions across both the front-end web server and the back-end database. By monitoring both web and subsequent database requests, it's possible to ferret out attacks that independent IDS would not be able to identify. Furthermore, it quantifies the limitations of any multitier IDS in terms of training sessions and functionality coverage. I am implementing the prevention techniques for attacks. I am also finding IP Address of intruder.

Keywords: IDS; IIS; attacks.

I. INTRODUCTION

To protect multi-tiered web services, Intrusion detection systems (IDS) have been widely used to detect known attacks by matching misused traffic patterns or signatures [1]. In the existing system we require different IDS one for web server and another for database server. Two IDSes required so we need to create two IDSes with different prevention measure first IDS that contains prevention measure related to web server so attack should not happen on web server but some time attack happen on database server by passing web server so for that reason need to create another IDS with prevention measure related to database server attack. We want to avoid creating two IDS so we are creating one DoubleGuard system that act as IDS and prevent both side of attack. Attack may be on web server or database server. DG actually invented by Le, Stavrou and Kung[1].

Most of the IDS examine the attack individually on web server and database server. In order to protect multi-tiered web services an efficient system call Intrusion Detection System is needed to detect attacks by mapping web request and SQL query, there is direct causal relationship between request recived from the front end web server and those generated for the database backend. Le, Stavrou and kung showed that this causality model can be generated accurately and without prior knowledge of web applications functionality.

I implemented the DG using sessions. It will allocate the isolated session for each user it is practical for most of the web applications.

Static web sites has the controlled environment whereas dynamic website not. Dynamic web site allow persistent back end data modification through the HTTP requests to include the parameters that are variable and depend on the user input. Because of which the the mapping between the web and the database rang from one to many as shown in the mapping model.

In the proposed system we are implementing Double Guard that handle both sides of attack. Attack may be from static web site or dynamic web site. No need to create two different IDSes for two different web site. Double Guard can handle both types of attack.

Following tasks should be accomplished by Doubleguard:

» It should prevent the damage that detected intrusion could cause

» it should mitigate the damage that detected intrusion could cause

» to identify the perpetrator

» to discover the new attacks patterns

» In order to fulfil the above tasks double guard must follow some requirements. The systematic overview of the requirements is given in [8]

» Accuracy: It must not identify the legitimate action in system environment as anomaly or misuse like IDS

» Performance : DG performance must be high enough to carry out the real time intrusion detection

» Completeness : It should not fail to detect an intrusion. It is oractically impossible because it is impossible to have a global knowledge about past, present and future attacks.

» Fault tolerance : it should be resistant to attacks and its consequences

» Timeliness : DG should perform the analysis as quickly as possible.

Apart from the functional requirements, DG should satisfy the number of economical requirements, in particular case, cost.

» cost of the product

» cost of additional computer resources needed

» cost of administration

» An importance of all this is oblivious. DG should available not only to large enterprises, but also small enterprises, as well as private person.

II. Review of Intrusion Detection Systems

A network Intrusion Detection System (IDS) can be classified into two types: anomaly detection and misuse detection. Anomaly detection first requires the IDS to define and characterize the correct and acceptable static form and dynamic behavior of the system, which can then be used to detect abnormal changes or anomalous behaviour. The boundary between acceptable and anomalous forms of stored code and data is precisely definable. Behaviour models are built by performing a statistical analysis on historical data or by using rule-based approaches to specify behavior patterns an anomaly detector then compares actual usage patterns against established models to identify abnormal events. Our detection approach belongs to anomaly detection and we depend on a training phase to build the correct model. As some legitimate updates may cause model drift, there are a number of approaches [13] that are trying to solve this problem. Our detection may run into the same problem; in such a case, our model should be retrained for each shift.

An IDS such as also uses temporal information to detect intrusions. On temporal intervals the Allens Algebra is applied to discover the relation between temporal intervals and to store them for further classification. DG, however, does not correlate events on a time basis, which runs the risk of mistakenly considering independent but concurrent events as correlated events. DG does not have such a limitation as it uses the container ID for each session to causally map the related events, whether they be concurrent or not.

Apiary[13] an intrusion detection system for the desktop computers. Like DG Apiary also creates different containers, but it creates it only for the running web applications on a single desktop. In mobile operating system we use Apiary of different web applications running differently. So that we can acces them fast than our regular internet explorer. Whereas DG can create different sessions for every user on multiple systems.

CLAMP(Confidentiality to LAMP)[3] is an sensitive data leakage prevention technique even in the presence of attack. Clamp guarantees that user sensitive data can only be accessed by code running on the behalf of different user. Whereas DG focuses on modelling the mapping patterns between the HTTP request and the database queries to detect the malicious user session. CLAMP requires modification to the existing application code, and the Query Restrictor works as a proxy to mediate all database access requests. Moreover, resource requirements and overhead differ in order of magnitude: DoubleGuard uses process isolation whereas CLAMP requires platform virtualization, and CLAMP provides more coarse-grained isolation than DG. However, DG would be ineffective at detecting attacks if it were to use the coarse-grained isolation as used in CLAMP. Building the mapping model in Dg would require a large number of isolated web stack instances so that mapping patterns would appear across different session instances.

In addition, validating input is useful to detect or prevent SQL or XSS injection attacks [3], [6]. This is orthogonal to the DG approach, which can utilize input validation as an additional defence. However, we have found that Double- Guard can detect SQL injection attacks by taking the structures of web requests and database queries without looking into the values of input parameters (i.e., no input validation at the web server).

III. Algorithm Review

A. Static Model Building Algorithm:

In this algorithm we are getting set of web request and generate SQL query according to web request. If user perform web request and for that web request SQL query is not generated then that web request mark as EQS (Empty Query Set) else generated SQL query and get result. If we got same result as expected up to threshold value then mapping is correct otherwise need more training sessions. In NMR (No match request) [1]SQL query generated without web request from user but according to SQL query action will be performed.

I have used the following static model building algorithm to create the static webpage.

Algorithm 1 Static Model Building Algorithm.

Require: Training Dataset, Threshold t

Ensure: The Mapping Model for static website

1: for each session separated traffic Ti do

2: Get different HTTP requests r and DB queries q in this session

3: for each different r do

4: if r is a request to static file then

5: Add r into set EQS


```
6:      else
7:        if r is not in set REQ then
8:          Add r into REQ
9:          Append session ID i to the set ARr with r as the key
10:   for each different q do
11:     if q is not in set SQL then
12:       Add q into SQL
13:       Append session ID i to the set AQq with q as the key
14: for each distinct HTTP request r in REQ do
15:   for each distinct DB query q in SQL do
16:     Compare the set ARr with the set AQq
17:       if ARr = AQq and Cardinality(ARr) > t then
18:         Found a Deterministic mapping from r to q
19:         Add q into mapping model set MSr of r
20:         Mark q in set SQL
21:       else
22:         Need more training sessions
23:       return False
24: for each DB query q in SQL do
25 :     if q is not marked then
26:       Add q into set NMR
27: for each HTTP request r in REQ do
28:     if r has no deterministic mapping model then
29:       Add r into set EQS
30:     return True   .....
```

B. AES Algorithm:

We are using AES encryption algorithm for encryption and decryption of data. We are already giving security to our application but not for data. By using encryption we can provide security for our data also. AES algorithm having 128 bit size. We are storing data in encrypted format. When user upload file, we get that file perform encryption by using encryption algorithm i.e. AES (Advance Encryption Standard)[4]. After that it give encrypted file and that file stored into database. When use upload file by using encryption method we encrypt that file and store into database in encrypted format but when user click on download file get that encrypted file from database perform decryption method on that file and convert it into original and readable format. Once data stored into database in encrypted format then even if database got hacked by hacker, hacker cannot understand what is that data due to encrypted format.

Explanation: Encryption

1. Input: Attribute Value (Attr).

2. Get Byte [](B1) of that Attr.

3. Generate Key().

4. Perform Encryption on B1.

5. Convert B1 into string(EAttr).

Decryption

1. Input: Encrypted attribute value(EAttr)

2. Convert EAttr into byte [](B2).

3. Generate Key.

4. Perform Decryption on B2.

5. Convert B2 into string(DAttr).

IV. SYSTEM ARCHITECTURE

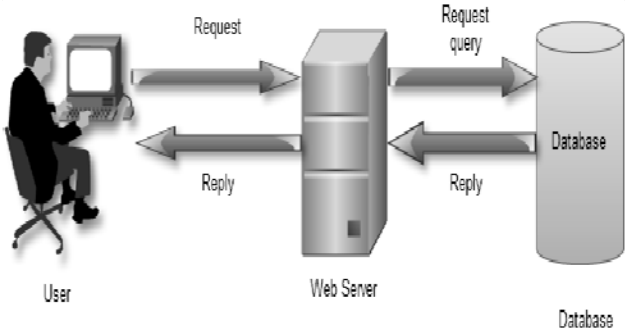

Figure 1: System architecture

The system architecture of the project is shown in figure 1. The architecture of the system shows three modules or three layers the user , the server and the database layer. User uses all the applications which server provides to him/her. In our architecture user has windows operating system . The web server is IIS (Internet Information Services) server which is inbuilt in the windows 7 ultimate operating system. The server has some dynamic and some static web pages which I have created using ASP.NET and C#. and these web pages are working in Google chrome browser. And the database is My SQL which stores the user's information and also protects it. In this architecture I can add multiple users on single web server. All these users can use the same database too.

The static and dynamic web pages are shown with above functions. The functional diagram shows all the possible ups and downs of the project. ADG(advanced double guard) system is created to protect the user's confidential data from outside world or from intrusions. The intrusions can badly damage the user's confidential data or file. To protect from this kind of damage the ADG is build up.

a) Attack Scenario :

1. PRIVILAGE ESCALATION ATTACK :

This attack shows how the attacker can access the Admin's credentials and act as admin in the system. The attacker gets the admin's user id and password and gives command to web server to get the private database of user. Suppose, the attacker login into the web server as a normal user and trigger admin queries to obtain the administration data then this kind of attack can

never be detected by the IDS or normal intrusion detection technique. But as our system is allocating the different sessions for different user we can easily detect this kind of attack.

2) HIJACK FUTURE SESSION ATTACK :

Figure 4 shows how this attack attempted by the middle person/attacker. The third person accesses the username and password of normal user and misuse them. In banking, travelling, personal accounts these kinds of attacks are happened to get the personal information of normal user. But in my double guard technique this type of attack is not possible. As every user is getting his/her personal session which no can access. So using this technique we can prevent this kind of attack.

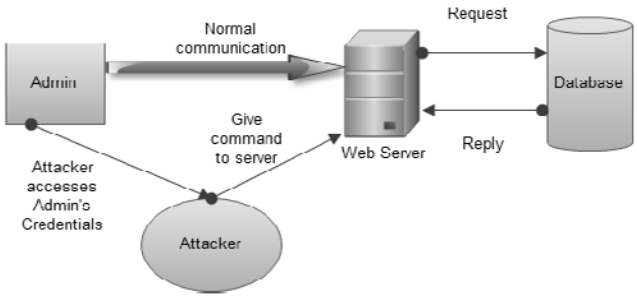

Figure 2: Privilage escalation attack

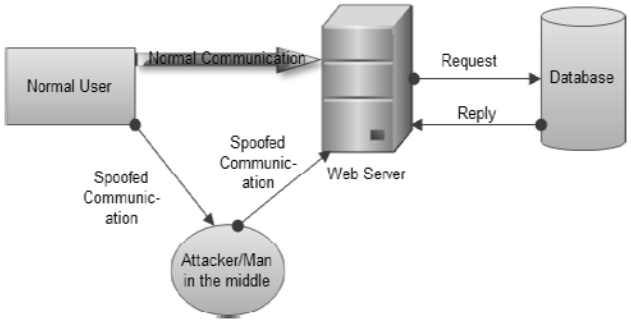

Figure 3: Hijack Future session attack

3) SQL INJECTION ATTACK:

Figure 5 shows how the SQL injection attack happens. In this attack rather communicating with the user and web server, the attacker directly communicate with web server to access the username and password pair of any normal user to attack the back end database. So this will not be detected by the DG as who is accessing the credentials.

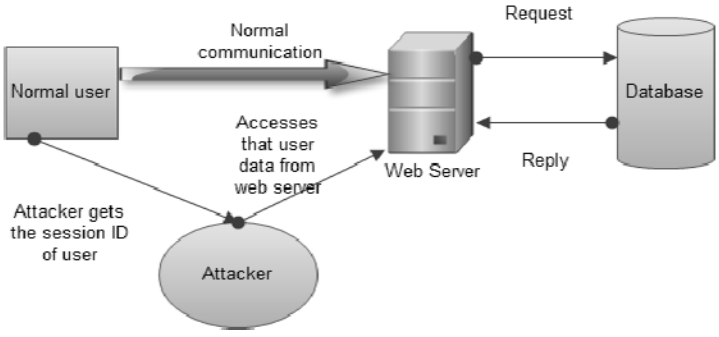

Figure 4 : SQL injection attack

4) SESSION FIXATION ATTACK:

Figure 6 shows how session fixation attack executed by the attacker. Attacker stoles the user's session id and takes over his/her communication. Attacker may change or edit the data of user. But our system can prevent this kind of attack. As the session allocated to one user get destroyed immediately as the user logs out from the system. As the session id of one user at one time is never repeated.

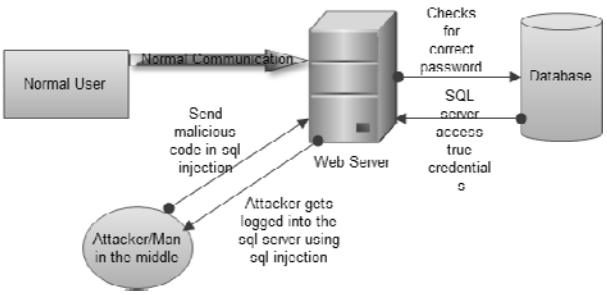

Figure 5: Session Fixation attack

V. Detecting And Preventing Attacks

The model built, can be used to detect and prevent the intrusion or malicious sessions for static and dynamic web sites. I have collected total 30 session for 3 users continuously for 10 days. I have created half of my project to show the attacks on the web pages. I have created some attacks to hack or attack the web site. and in the next half I have show how I prevented the web sites from those attacks.

A. Privilege escalation attack:

For this attack , according to our pervious discussion the attacker visit the web site as normal user and compromises the web server to issue the set of admin credentials. Using these credentials she can access the sensitive information . In the first portion I have shown how the normal user takes over the admin credentials.

" lacalhost:5682/Blogbook/BlogHome.aspx?id=Normal%20User", this the user changes

" lacalhost:5682/Blogbook/Admin/AdminHome.aspx"

In this way editing the URL the normal user can takes over the admin user. But to prevent this I have coded the URL as following .

" lacalhost:5682/Blogbook/BlogHome.aspx?id=Adh6jmkkokhd56.aspx".

B. Hijack Future Session Attack:

In this attack if attacker wants to hack any user session by copying and pasting the URL she gets from cache or history, then she will not be able to find the session of that user it will her to the login page not the account page.

C. SQL Injection attack:

In SQL injection attack as we have seen , if the attacker send the query " 'OR 1 = 1;--"with any random password then she is able to take over the session of user she wanted. But as we are using different sessions for different user she will not be able to hack any normal user's account.

D. Session fixation attack:

In this attack the session will not allow the attacker to go back to the home page even after click on back button.

VI. Conclusion

I presented an intrusion detection system that builds models of normal behavior for multi- tiered 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 IDS, ADG forms session-based IDS with multiple input streams to produce alerts. Such correlation of different data 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. Rather it also can prevent the web applications from intrusions.

ACKNOWLEDGEMENT

I would like to thank Shri Chairman D.Y.Patil and Management and the Director/Principal Dr. Uttam Kalwane, Guide, Head,Coordinator and Colleagues of the Department of Computer Engineering, Dr.D.Y.Patil School Of Engineering and Technology, Charoli, B.K.Via –Lohegaon, Pune, Maharstra,India, for their support, suggestions and encouragement.

References

1. Meixing Le, Angelos Stavrou, Brent ByungHoon Kang.” DoubleGuard: Detecting Intrusions In Multi- tier Web Applications” IEEE transaction on dependable and secure computing vol.9 no.4 year 2012
2. "A Comprehensive Approach to Intrusion Detection Alert Correlation", Fredrik Valeur, Giovanni Vigna, IEEE transaction on dependable and secure engineering, Vol. 1NO. 3,2014
3. "CLAMP: Practical Prevention of Large-Scale Data Leaks" Bryan Parno, Jonathan M. McCune, Dan Wendlandt, David G. Andersen, Adrian Perrig Springer-Verlag Berlin, Heidelberg ©2006.
4. "A Stateful Intrusion Detection System forWorld-Wide Web Servers", IEEE Computer Society Washington, DC, USA ©2003 B. Parno, J. M. McCune, D. Wendlandt, D. G. Andersen, and A. Perrig. CLAMP:
5. "A New Algorithm for Inferring User Search Goals with Feedback Sessions", IEEE Transaction of knowledge and data engineering, march 2013.
6. Practical prevention of large-scale data leaks. In IEEE Symposium on Security and Privacy. IEEE Computer Society, 2009.
7. Udaya Kiran Tupakula Vijay Varadharaj an , "A Practical Method to Counteract Denial of Service Attacks" 2003.Stavrou, G. Cretu-Ciocarlie, M. Locasto, and S. Stolfo. Keep your friends close: the necessity for updating an anomaly sensor with legitimate environment changes. In Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence, 2009.
8. D. Wagner and D. Dean." Intrusion detection via static analysis". In Symposium on Security and Privacy (SSP 01), May 2001.
9. M. Christodorescu and S. Jha. "Static analysis of executables to detect malicious patterns."
10. P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Krugel, and G. Vigna. "Cross site scripting prevention with dynamic data tainting and static analysis". In NDSS 2007.
11. R. Sekar." An efficient black-box technique for defeating web application attacks." In NDSS. The Internet Society, 2009
12. V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna. "Toward Automated Detection of Logic Vulnerabilities in Web Applications." In Proceedings of the USENIX Security
13. Apiary: Easy-to-Use Desktop Application Fault Containment on Commodity Operating Systems, USENIX Association 2010.

