Implementation of Customized UTP Algorithm for Attack Detection in Multitier Web Applications

Ms. Ashwini Patil
Department of Computer Engineering, Vidyalankar Institute of Technology, Wadala
Email: patil.ashvini86@gmail.com

Prof. Sachin Deshpande
Associate Professor, Department of Computer Engineering, Vidyalankar Institute of Technology, Wadala
Email: sachin.deshpande@vit.edu.in

Abstract:- Internet services and application have gained lots of importance in our daily life such as banking, travel and social networking. Personal information from any of the remote location can be communicated and managed with the help of Internet. Due to their omnipresent use for daily task, web applications have been target for attack. To deal with increasing demand and data complexity web services and applications have moved to a multitiered design. The idea is to detect attacks in multitier architecture to model the network behavior of user sessions across both the front-end web server and the back-end database. The attacks like SQL injection, cross site scripting attack, privilege escalation attack and direct DB attack can be monitored with both the web and subsequent database requestusing customized UTP algorithm, which an independent system cannot do.

*****

1. Introduction

A lot of importance has been achieved by internet services and application in our daily life. They provide communication and the management of personal information from any remote location. Since they have to handle large amount of data, web services have moved to multitier design. Multitier architecture is a logical partition of the separation of operations in the system. Each tier is assigned its unique responsibility in the system. Multitier architecture typically consists of a presentation tier, a business tier, and a data tier which are logically separated. In multitier design files are uploaded to the web server run as front end and the data is outsourced to the database server run as backend.

Web servers and web based applications are popular attack targets because of their ubiquitous use. All computing devices are connected to the World Wide Web via the client-server architecture. In this architecture, the client requests some information from web server through a web browser. Then to fetch data the web server is connected to a database server. So the connection between client and web server as well as the connection between web server and database server needs to be well secured, which is very challenging task. To protect these connections in multitier web services various web IDSs and database IDSs are available. Both web IDS and database IDS can detect abnormal traffic sent to them. But sometimes neither the web IDS nor the database IDS would detect cases wherein normal traffic is used to attack the web servers and database server[1]. For example, if an attacker with non admin privileges can log in as a normal user, and then he/she can find a way to fire a privileged query, in this case web IDS only see the logic traffic and the database IDS would see only normal traffic of privileged user.

So the idea is to provide an improved approach to an intrusion detection system for multitier web applications. This approach can build a casual mapping model which can map the web server requests and its subsequent database query. By monitoring both web server request and subsequent database query, system can ferret out attacks like SQL injection, cross site scripting attack, privilege escalation attack and direct DB attack.

2. Attack Scenarios :

Our system provides a protection against following four types of attack:

2.1 SQL Injection Attack[1]:

In SQL Injection attack Attackers can use existing vulnerabilities in the web server logic to inject the data or string content that contains the exploits and then use the web server to relay these exploits to attack the back-end database. Since we have a mapping mode between generate unique number and DB queries, it will find out deviation from the SQL query structure that would normally follow such a web request.

2.2 Direct DB Attack[1]:

It is possible for an attacker to bypass the web server or firewalls and connect directly to the database. 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.

2.3 Privilege Escalation Attack[1]:

For Privilege Escalation Attacks, attacker visits the website as a normal user aiming to compromise the web server process or exploit vulnerabilities to bypass authentication. At that point, the attacker issues a set of privileged (e.g., admin-level) DB queries to retrieve sensitive information. We log and process both legitimate web requests and database queries in the session traffic, but there are no mappings among them. IDSs working at either end can hardly detect this attack since the traffic they capture appears to be legitimate. Using the mapping model our system can capture the unmatched cases.

2.4 Cross site scripting[1] :

Cross site scripting is yet another variety of attacks on web applications. In this attackers embed malicious client scripts via legitimate user inputs. Attackers utilize the fact that there is a trust relationship between a web server and a browser. Such type of attacks can occur when data sent to the server are put onto the web site without being properly analysed for possible security threats. Our system is able to detect this attack.

3. Overview and working

This system is used to detect attacks in multi-tiered web services. This approach can create mapping model of isolated user sessions that include both the web front-end and back-end network transactions. Proposed system provides two layer of security .In First layer, the system will filter all requests coming to server based on some parameter. In second layer, all queries which hitting to database will go through query parsing. The system parses all queries and fetch out database Table name and type of database query. After parsing database query, it will check Table name against user role and based on checking system will allow or disallow database query for further process.

3.1 SystemArchitecture

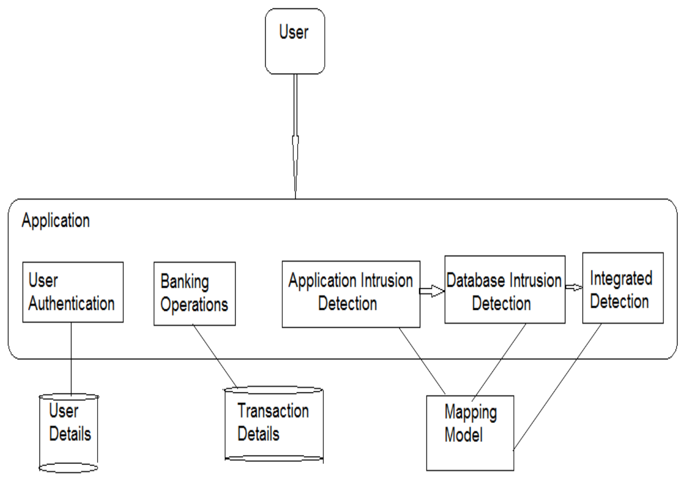


Fig.4.1.1System architecture

3.2 Modules of the proposed system

The number of possible modules for the application is as follows:

1. **User Authentication**:

This module facilitates the system to validate the user before using the application and thereby enabling only valid users to utilize the application. Thereby, the system facilitates access to selected users.

2. **Banking Operations**:

This module will facilitate the user to perform analysis in terms of the certain banking features to be utilized by various users and thereby facilitate it to execute certain crucial operations related to the application.

3. **Application Intrusion Detection**:

This module will facilitate the system to detect the Intrusion related to the features of the application wherein each type of user will be allocated a set of features identified by the mapping model created by the application for the user and thereby prevent invalid access operations to be executed.

4. **Database Intrusion Detection Module**:

This module will facilitate detection intrusions occurring due to invalid structure of data which affects the backend systems of the applications. In order to facilitate detection of such invalid data structures this module enables a second level of detection and thereby facilitate efficient protection to the application. It identifies the valid data structures in terms of backend communication for each kind of user based on a mapping model and thereby facilitate it efficient detection.

5. **Dual Detection Mechanism**:

This module is an integration of the Database and Application Intrusion detection process and ensure high-level of safety to the application.

3.3 Algorithm Design

Working of our system is based on customized UTP model building algorithm. UTP stands for union, tautology and piggybacking.
Tautology-based attack is to inject code in one or more conditional statements so that they always evaluate to true. The most common usages of this technique are to bypass authentication pages and extract data. If the attack is successful when the code either displays all of the returned records or performs some action if at least one record is returned.
In union-query attacks, Attackers do this by injecting astatement of the form: UNION SELECT <rest of injected query> because the attackers completely control the second/injected query they can use that query to retrieve information from a specified table. The result of this attack is that the database returns a dataset that is the union of the results of the original first query and the results of the injected second query.
In the piggy-backed query attacker tries to append additional queries to the original query string.On the successful attack the database receives and executes a query string that contains multiple distinct queries. In this method the first query is original whereas the subsequent queries are injected.

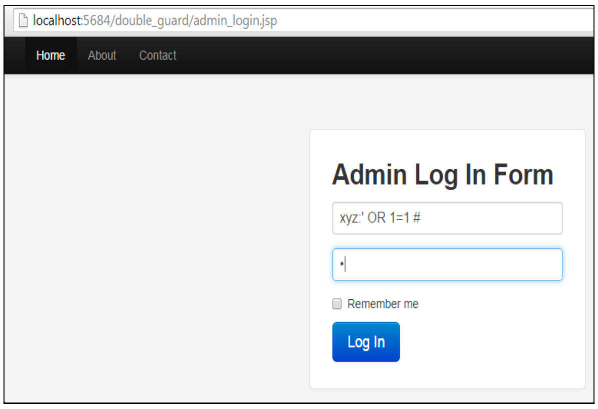
UTP algorithm is as follows:

Steps:

1. Identify query model for Data Layer
2. Eliminate comment data generated from final query structure
3. Identify query termination tokens
4. Split multiple queries generated at Data layer
5. Execute CRUD analysis for each query generated
6. Identify CRUD tokens
7. Identify data models (tables) specified by query
8. Execute mapping model analysis
9. Identify current user type (user, admin, employee) from session
10. Identify mapping model for CRUD tokens (or other SQL tokens) from static model for current user type
11. for each CRUD token
12. compare data model(s) for current query with data model for current token
13. if data models similar then
14. set query_status as valid
15. end for
16. else
17. repeat for with next token
18. Identify conditions specified in query
19. Identify alterations in conditions w.r.t to valid query
20. if alteration identified then
21. set tautology status = true
22. set query_status as invalid
23. else split queries if UNION token present
24. Execute mapping model analysis on each query generated
25. identify query_status of above step
26. if query_status is valid then
27. execute piggy backing analysis after identification of query termination tokens
28. split queries based on these tokens
29. Execute mapping model analysis on each query generated
30. identify query_status of above step
31. returnfinalquery_status

5.Results:

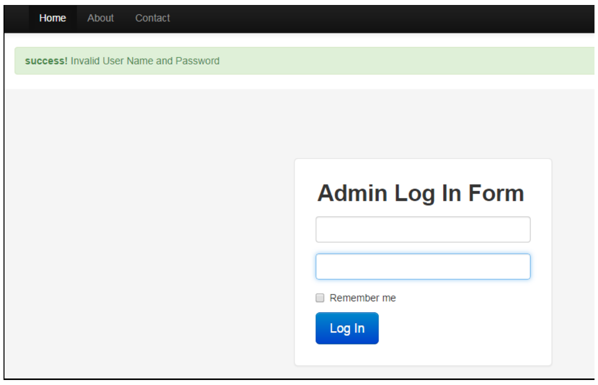
5.1 SQL Injection Attack Detection:


localhost:5684/double_guard/admin_login.jsp

Admin Log In Form

xyz:' OR 1=1 #

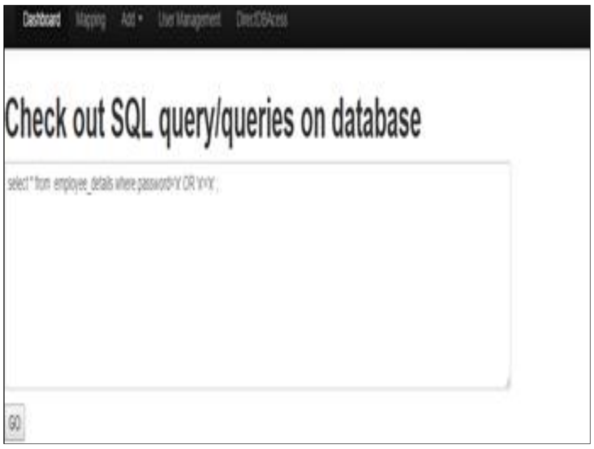
Fig.5.1.a. SQL injection input


success! Invalid User Name and Password

Admin Log In Form

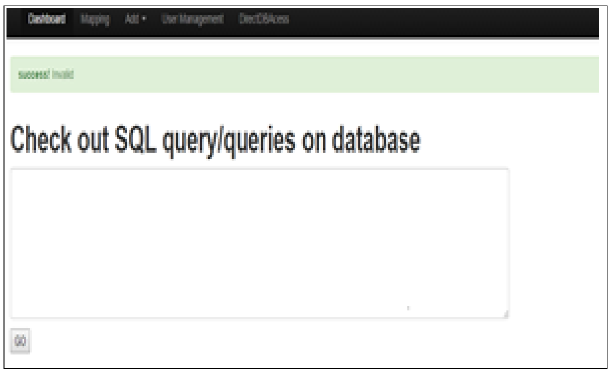
Fig.5.1.b. SQL injection output

5.2 Direct DB Attack Detection:


Check out SQL query/queries on database

select * from employee_details where password='x' OR 'x'='x' ;

Fig.5.2.a. Direct DB input

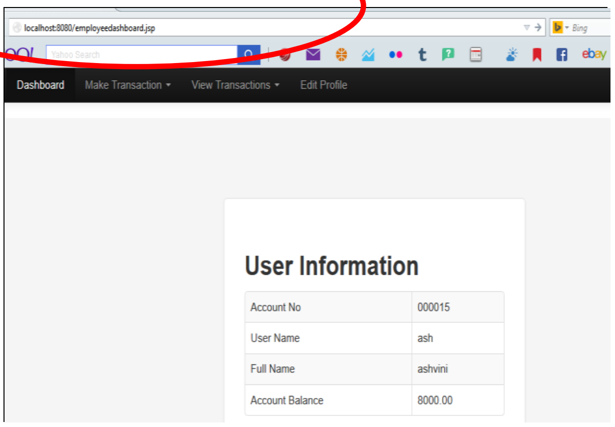


success! Invalid

Check out SQL query/queries on database

Fig.5.2.b. Direct DB output

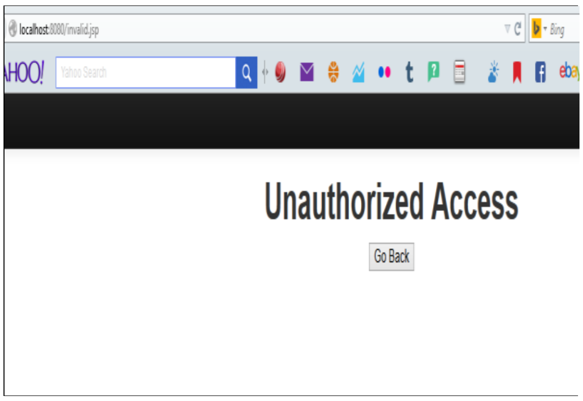
5.3 Privilege Escalation Attack Detection:



User Information

Account No	000015
User Name	ash
Full Name	ashvini
Account Balance	8000.00

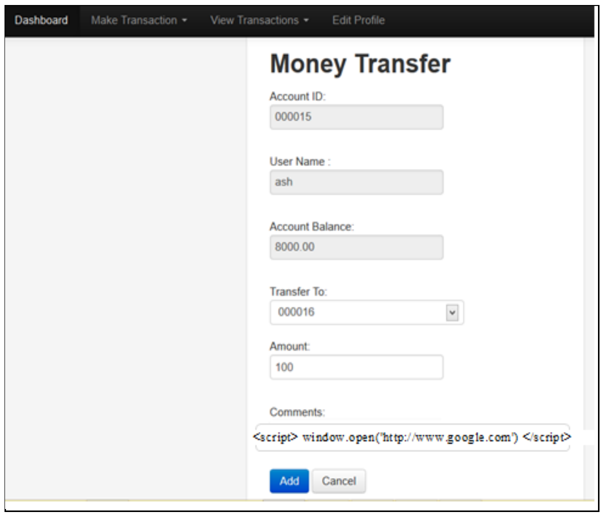
Fig.5.3.a. Privilege escalation input where normal user try to change his role



Unauthorized Access

Fig.5.3.b. Privilege escalation output

5.4 Cross Site Scripting Attack Detection :



Money Transfer

Account ID: 000015

User Name : ash

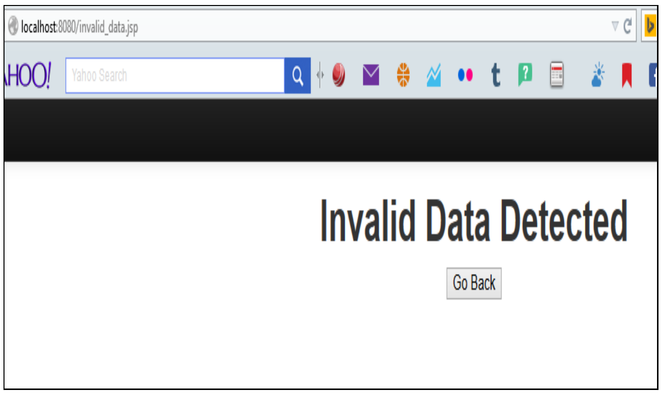
Account Balance: 8000.00

Transfer To: 000016

Amount: 100

Comments: <script> window.open('http://www.google.com') </script>

Fig.5.4.a. Cross site scripting input



Invalid Data Detected

Fig.5.4.b. Cross site scripting output

6. Conclusion

We presented an intrusion detection system that can detect various web attacks namely SQL injection attack, cross site scripting attack, privilege escalation attack and direct DB attack occurred in multi-tier web applications. We achieved this with help of customized model building algorithm and by creating mapping model.

7. References

[1] Meixing Le, AngelosStavrou, Brent ByungHoon Kang,” DoubleGuard: Detecting Intrusions in Multitier WebApplications”, IEEE Transactions on dependable and secure computing, vol. 9, no. 4, July/august 2012.

[2] Ludinard R.,Totel E.,Tronel F. NicometteV,Alata E, Akrout R., BachyY.,“Detecting attacks against data in the web applications” Risk and Security of Internet and Systems, 7th International Conference ,2012

[3] Dessiantnikoff A.,Akrout R.,Alata E.,Kaaniche M.” A clustering approach for web vulnerabilities detection”

[4] Cristian Pinzon, Alvaro Herrero, Juan F.De Paz, Emilio Corchado and Javier Bajo "CBRID4SQL"A CBR intrusion detector for SQL injection attacks" IEEE 2010.

[5] R. Sekar." An efficient black-box technique for defeating web application attacks" In NDSS.The Internet Society, 2009.

[6] G. Vigna, F. Valeur, D. Balzarotti, W. K. Robertson, C. Kruegel and E.Kirda, "Reducing Errors in the Anomaly-Based Detection of Web-Based Attacks through the Combined Analysis of Web Requests and SQL Queries", J.Computer Security, vol.17, no.3, 2009

[7] 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.

[8] 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.

[9] A.Srivastava,S.Sural, and A.K.Majumdar "Database intrusion detection using weighted sequence mining" JCP, 1(4), 2006.