International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 1, Issue 3, November 2014

Implementing a Technique for Protection of Session Data Dependencies in Web Application

Vaishali Malekar[1]
Assistant Professor
Department of computer Technology
KITS, Ramtek
E-mail:vaishalimalekar@gmail.com[1]

Abstract: Security is the essential topic in web application. Online banking, e-mails, e-shopping have become an important part of our routine life. Web application vulnerabilities cause run-time errors. These vulnerabilities due to broken data dependencies leads to forceful browsing. With the forceful browsing attack, the attacker gains access to a restricted page within a Web application by supplying a URL (Uniform Resource Locator) directly (forcing the URL) instead of accessing it by following links from other pages in the application. The paper provides solution to the problem of broken data dependencies. The solution employs the tree-based data dependency concept on session data. The solution is developed using JSP/Servlets technology. The aim of this paper is to protect web application from broken data dependencies.

Keywords: Web application, forceful browsing, broken data dependencies, session data**.**

I. INTRODUCTION

The importance of web application and its security increasing day by day. At present there is an increasing number of web applications in all aspects of business and education. Also, web applications are becoming more and more important part of any system. Web application tends to be error prone and implementation vulnerabilities are commonly exploited by attacks. So, increasing the reliability and security of web applications has become most important issue.

Present technologies such as anti-virus software programs and network firewalls provide the secure protection at the host and network levels, but not at the application level. Existing solution for the broken session data dependencies is Web Application Firewall (WAF) [9]. WAFs are applied to mitigate a range of vulnerabilities, including vulnerabilities to forceful browsing. But a malicious user will typically apply forceful browsing to exploit implementation-specific broken session dependencies in data-centered web applications in a more or less controlled way. So, existing security solutions do not provide adequate support to protect web applications against such implementation-specific bugs. This paper introduces a technique to protect against broken data dependencies vulnerability. This is an application level vulnerability. The solution uses the java server pages (JSP)/ servlet technology for server side implementation. Tree-based dependency concept is used for defining the interactions between the components. Hashtable class is used in implementing the solution.

A. Data Dependencies In Data-Centered Application

In data-centered application, there is a central data structure and set of components interacting with the repository. Each component can indirectly interact with other component, but all the components can indirectly interacting with the shared repository. At the time of execution, implicit semantically dependencies exist between the components and the shared repository.

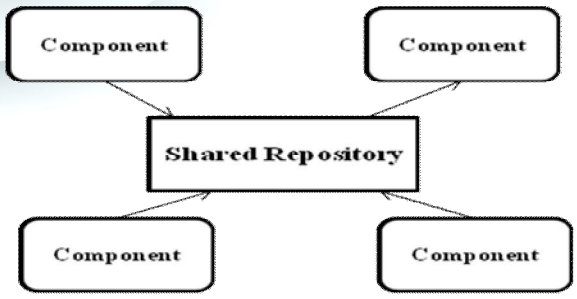


Figure 1 Data-dependencies in data-centered web application

Breaking data dependencies is one of the common risk in data-centered application. These broken data dependencies vulnerabilities results in run-time errors like Null Pointer Exceptions. It decreases the reliability and security of web application. These run-time errors results in the execution of unexpected application logic, information leakage and the denial-of-service.

B. Broken Data Dependencies

Broken data dependencies may cause two types of exceptions in data-centered web application. They are as below

a. Null Pointer Exception

A data item is not available on the shared repository although a reading component expects it on the repository during execution.

b. Class Cast Exception

The type of an available data item does not correspond with the type expected by the reading component. These mismatches lead to the runtime errors. Due to these runtime errors and loose coupling the data can break in data-centered applications. These broken data dependencies decrease the reliability and security of the web applications.

The paper presents the solution to the problem of broken data-dependencies on session data and how it secure the application from being exploited. Section I explains the data dependencies con cept in data-centered application and run-time errors due to broken data dependencies. Section II explains the related work and overview of the solution to the problem of broken data dependencies. Section III explains the implemented work. Section IV explains the results and discussion. Section V provides the conclusion.

II. RELATED WORK

Most of the security solutions related to web application vulnerabilities have already been implemented [1][2][3][4][5][6][7], but most of them focused on SQL Injection, cross-site scripting and use tainting, pointer or data flow analysis. This paper is mainly focused on the vulnerability of no broken data dependencies. Gould et al.aim to reduce the number of runtime errors in Web applications by applying static verification [8]. The solution focused on the reduction of SQL runtime exceptions and used a static analysis tool to verify the correctness of all dynamically generated query strings within an application.

This paper secures the web application from broken data dependencies vulnerability which is an application level vulnerability. The interactions between the various components are defined using the tree-based dependency concept. At run-time the components will be accessed according to these already defined dependencies. Dynamic filter will allow only valid client-server interactions.

III. IMPLEMENTED WORK

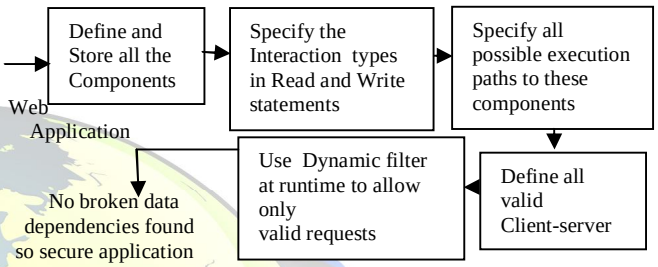


Figure 2 Overview of solution

An online shopping application is designed which is a data-centered web application. JSP/servlet technology is used for designing. The technology supports the management of user sessions, also track to which user session a web request belongs. These technology also provide the server-side state for each user session. While processing a web request, server-side web components can store non-persistent user-specific data (for example, a shopping cart in an e-commerce site) in a shared data repository bound to the user session.

The basic operation of this application are given as below

- The user visits the online shopping web site. If the user is existing user, he logs in otherwise create an account
- The user browse the items to be purchased and add the items to be purchased to the cart
- User provides the delivery information to deliver the product to the respective address
- User submit the payment information for billing purpose
- User gets the order confirmation message.
- The user is informed that the process is finished.

Now the steps for implementing the solution are discussed below.

- Define and store all the interacting components using Hashtable (). Starting path for all the pages(or components) are stored in the hashtable. Hashtable algorithm makes the storage and retrieval of these

paths easier. All the pages are identified by some particular names like A, B, C, D and so on.

```
node.put("…JSP/ProductList.jsp",
"A");
 node.put("…/JSP/AddToCart.jsp",
"B");
node.put("…/JSP/DeliveryInformation.
jsp", "C");
node.put("…/JSP/PaymentInformation.j
sp", "D");
node.put("…/JSP/OrderConfirmation.js
p", "E");
node.put("…/JSP/Finished.jsp", "F");
node.put("…/JSP/UpdateCart.jsp",
"G");
```

Then define the types of interactions between the components and the shared repository in the form of read and write statements. All the pages interact with the repository by two types of interactions. These interactions may be read and write. For e.g. productlist page is used to read the various types of products whereas productcart is used for both read and write operation.

```
sessionAtrrbute.put("productlist_read",
"read");
sessionAtrribute.put("productcart",
"write");
sessionAtrrbute.put("productcart_read",
"read");
 sessionAtrribute.put("delivery_read",
"read");
 sessionAtrribute.put("orderconfirm_read",
"read");
 sessionAtrrbute.put("finish_read",
"read");
```

- Next, apply the tree-based data dependency concept on these components. Constraints are applied for accessing the pages in the application. Dependencies between the components are already defined. All the pages will be accessed at run-time according to the tree-based concept. Thus run-time errors can be reduced. For e.g. the starting point of the application should always be either A or B, The page B add to cart can be accessed only after accessing the pages A or C or G or B, the page for update cart G can be accesses only after accessing the pages A or C or B. Dependencies between the components are defined as shown below.

```
 tree.put ("Start", "A|B");
tree.put ("A", "B|A");
tree.put ("B", "A|C|G|B");
tree.put("G", "A|C|B");
tree.put("C", "D|A|B");
tree.put("D", "E|A|B");
tree.put("E", "F|A|B");
tree.put("F", "Start|A|B");
```

All possible access paths for all the pages in web application are already specified as given above.

```
treeCond.put("G","productlist_read|productcar
t|productcart_read");
treeCond.put("C","productlist_read|productcar
t|productcart_read");
treeCond.put("D",
"productlist_read|productcart|productcart_rea
d|delivery_read");
 treeCond.put("E",
"productlist_read|productcart|productcart_rea
d|delivery_read");
treeCond.put("F","productlist_read|productcar
t|productcart_read|delivery_read|orderconfirm
 read"):
```

Dynamic filter is used that will allow only valid requests at runtime. The above implemented solution guarantees the absence of broken data dependencies on session data.

Client/Server Interaction: Web application is triggered by the client who sends a web request for a certain URL to which the server-side web application will reply. The interactions between the client and the web server are typically nondeterministic, since the web user can decide which links to follow next within the web application. As a consequence, the protocol between a web client and a given web application is very open.

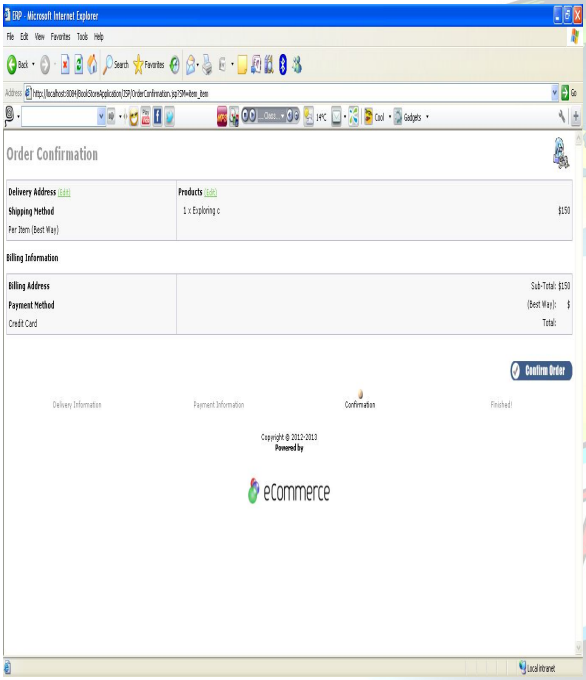
This nondeterministic sequence of web requests makes analysis of the dataflow dependencies in the session, context or application scope much harder. In contrast to dataflow dependencies in the request scope, the protocol history determines here whether or not a dataflow dependency is satisfied. In fact, satisfying the intended dataflow dependencies in such a reactive system imposes extra constraints on the client/server interaction protocol. In an e-commerce application for example, it makes little or no sense to contact the payment service before items are added to the shopping basket. In particular, retrieving a non-existing shopping basket from the shared session repository leads in this application to a run-time error for the prepare basket order. Thus, to satisfy this particular dataflow dependency, the client/server protocol must be constrained to only allow the pay command after items are added to the shopping basket.

The broken data dependencies results in run time errors which may cause execution of unexpected logic application, breaking data integrity. Thus the security and

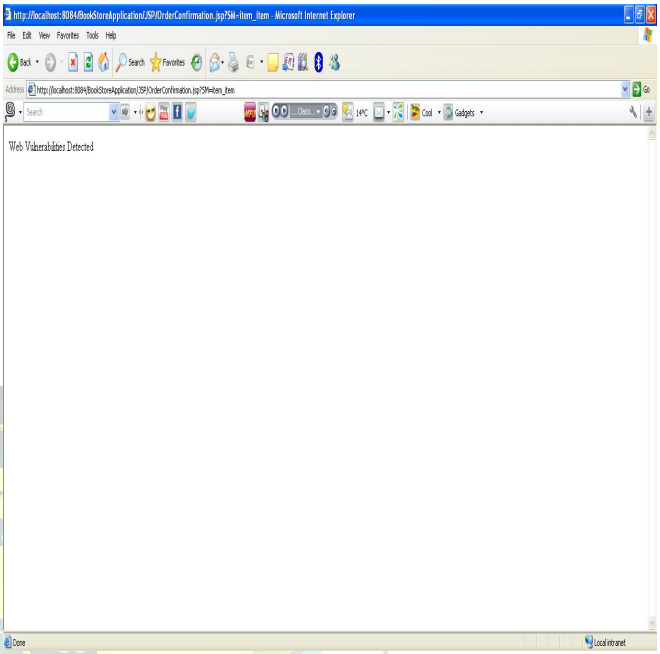
reliability of web application can be improved with this solution.

IV. Results And Discussion

User access the online shopping web site. User browses the items and adds the items to be purchased to the cart. Then he complete the payment procedure and after the payment is done, an Order Confirmation page is displayed. But an attacker has already copied the session link of Order Confirmation page as shown in diagram below with hyperlin "http://localhost:8084/BookStoreApplication/JSP/OrderConfirmation.jsp?SM=item_item".



By forceful browsing, if an attacker tries to access the same session link without following the dataflow dependency orders i.e. without selecting the item and adding an item to the cart and submitting the payment information, he will not be successful to get the Oder Confirmation page and buying the products without being an genuine user and adding the product and make payment . Thus here data depedencies between the pages are not broken. If there are no broken data dependencies then the there is no chance for an attacker to exploit the web application. Thus it is protecting the web application from getting being exploited by the broken data dependencies vulnerabilities.



Thus it guarantees that there are no broken data dependencies exist between the components and make the web application more secure.

V. Conclusion

The web applications are prone to error and easily exploited. If there are broken data dependencies in web application, it may be the severe problem. By applying forceful browsing, it becomes easy to exploit the web application.

If the data dependencies between the components are already defined explicitly with the help of tree-based dependencies concept and defined all valid client server interactions. These interactions are dynamically verified. If the user session does not follow the type of interactions then that session will not be proceed further and ends with an error page. All the webpages should follow the tree-based structure where all the possible execution paths for the webpages is defined. Thus it protect the session data. It may be possible to remove the broken data dependencies in the web application due to which there will be no runtime errors. This paper improves the security of web application by providing solution to the problem of broken data dependencies on session data.

REFERENCES

[1]. T. Pietraszek and C.V. Berghe, "Defending against Injection Attacks through Context-Sensitive String Evaluation," Proc. Eighth Int'l Symp. Recent Advances in Intrusion Detection, pp. 124-145, 2005.

[2]. V. Haldar, D. Chandra, and M. Franz, "Dynamic Taint Propagation for Java," Proc. 21st Ann. Computer Security Applications Conf. pp. 303-311, 2005.

[3]. Nguyen-Tuong, S. Guarnieri, D. Greene, J. Shirley, and D. Evans, "Automatically Hardening Web Applications Using Precise Tainting," Proc. 20th IFIP Int'l Information Security Conf., R. Sasaki, S. Qing, E. Okamoto, and H. Yoshiura, eds., pp. 295-308,2005.

[4]. W.G.J. Halfond and A. Orso, "Amnesia: Analysis and Monitoring for Neutralizing SQL-Injection Attacks," Proc. 20th IEEE/ACM Int'l Conf. Automated Software Eng., pp. 174-183, 2005.

[5]. W. Xu, S. Bhatkar, and R. Sekar, "Taint-Enhanced Policy Enforcement: A Practical Approach to Defeat a Wide Range of Attacks," Proc. 15th Usenix Security Symp., p. 9, 2006.

[6]. V.B. Livshits and M.S. Lam, "Finding Security Errors in Java Programs with Static Analysis," Proc. 14th Usenix Security Symp., pp. 271-286, Aug. 2005.

[7]. N. Jovanovic, C. Kruegel, and E. Kirda, "Precise Alias Analysis for Static Detection of Web Application Vulnerabilities," Proc. ACM SIGPLAN Workshop Programming Languages and Analysis for Security, pp. 27-36, 2006.

[8]. Gould, Z. Su, and P. Devanbu, "Static Checking of Dynamically Generated Queries in Database Applications," Proc. 26th Int'l Conf. Software Eng., pp. 645-654, 2004.

[9]. T.E. Uribe and S. Cheung, "Automatic Analysis of Firewall and Network Intrusion Detection System Configurations," Proc. ACM Workshop Formal Methods in Security Eng., pp. 66-74, 2004.

[10]. V.B. Livshits and M.S. Lam, "Finding Security Errors in Java Programs with Static Analysis," Proc. 14th Usenix Security Symp., pp. 271-286, Aug. 2005.

[11]. Desmet, L., Piessens, F., Joosen, W., Verbaeten, "Provable Protection against Web Application Vulnerabilities Related to Session Data Dependencies," IEEE Trans. Software Eng., vol. 34 no. 1, pp. 357-370, Jan/Feb 2008.

[12]. Gould, Z. Su, and P. Devanbu, "Static Checking of Dynamically Generated Queries in Database Applications," Proc. 26th Int'l Conf. Software Eng., pp. 645-654, 2004.

[13]. Dhanya Pramod, "A study of various approaches to assess and provide web based application security," International Journal of Innovation, Management and Technology, 2(1), February, 2011.

[14]. Katkar Anjali S., Kulkarni Raj B., "Web vulnerability detection and security mechanism," International Journal of Soft Computing and Engineering, Volume, 2(4), September 2012.

[15]. Nilesh Kochare, B.B.Meshram., "Tool to detect and prevent web attack," International Journal of Advanced Research in Computer Engineering and Technology, Volume, 1(4), Jun 2012.