Design and Implementation of New Data Validation Service using Web Technologies in web Applications

Tejinder Singh , Research Scholar of Computer Science, Lecturer (BFGI, Bathinda)
Email: tejinder31.singh@gmail.com

Abstract

We have designed a novel server-side data validation service, based upon semantic web technologies to solve the lack of data validation and bypassing validation issues. The New Data Validation Service consists of five components: Resource Description Framework annotation for elements of web pages, interceptor, Resource Description Framework extractor, Resource Description Framework parser, and data validator. Our solution is implemented as a prototype. In this paper, we have conducted a direct study to prevent the security vulnerabilities at the application level such as SQL injections. The results of this initial study have shown that the proposed service could provide a high coverage of prevention of security vulnerabilities.

Keywords: *Web application, data integrity, RDF, web system, web technologies, data validation, vulnerabilities*

1. Introduction

The Computer Emergency Response Team clearly demonstrate that the available security mechanisms have not made system break-ins impossible. Furthermore, the Gartner study found that 75% of Internet assaults are targeted at the web application level the data integrity can be violated on the server even though the communication channel between the server and client is secure. Web applications is organized into three tiers: a web browser tier, a web server tier, and a backend database tier. The user interaction is proposed in a web browser tier, the program logic (such as JSP or Servlet) is run in a web server tier, and the data operations (such as addition, deletion, and updating) are performed in a database server tier. It often have direct access to backend databases and, hence, sensitive data is much more difficult to secure. If there is no direct access to backend databases, attacks can use legitimate application protocols such as HTTP, and Simple Object Access Protocol (SOAP) to capture data and transmissions. Data validation scheme is the first defense against web attacks at the application level. Web developers have adopted a number of validation approaches to prevent loss of data integrity.

1.1 Client-side validation

This is effective for minimizing the number of necessary communication hits between the submitted form and received error message. However, the form validation modules of this approach can be removed. In addition, this approach cannot ensure that the client and server are valid.

1.2 Server-side validation

This approach can be used to validate sensitive data on a server before processing them by an application server. Depending upon the application and network traffic, the time taken between the submitted form on a web browser and the error message that is returned from a web server can be considerable. However, inside criminal might bypass the server-side input validation modules through using malicious manipulation software that intercept the user inputs at the server-side.

2. Hacker Break Validation

Hacker could break the client-side Validation modules. Bypassing input validation is a serious problem because it might cause failures in the software, and can also break the security upon web applications such as an unauthorized access to data. Even the Hacker cannot bypass the client and/or server input validation, web application flaws, such as cross-site scripting or SQL injection, now account for more than two thirds of the reported web security vulnerabilities. In an attempt to preparation this, we develop a new data validation services, based on semantic web technologies.

3. Data validation Bypassing

A validation scheme is necessary for both client and server-sides, but is not sufficient to ensure data integrity of web applications, because fundamentally a client-side input validation scheme is designed to validate basic properties of the input data: length, range, format, default value, and type. In addition, input validation can be used to enhance resistance to injection attacks such as SQL injection attack because SQL injection vulnerabilities result from insufficient input validation. However, an input validation scheme is useless if any malicious script or listener is already installed on a server.

The following approaches can cause loss of data integrity at the XHTML form level:

- ✓ Hidden fields manipulation.
- ✓ Script manipulation.

4. Architecture of New Data Validation Service.

I am presenting a new data validation service which is based upon semantic web technologies to prevent the Security vulnerabilities at the application level and to secure the web system even if the input validation modules are bypassed. As illustration in Figure 1, the data validation service architecture consists of the following components: RDF annotation for elements of web pages, interceptor, RDF extractor, RDF parser, and data validator. The next subsection will describe the functional overview of the proposed solution.

It should be noted that the components of the proposed architecture framework do not need to run on a dedicated machine, they can be run as separate processes on the server.

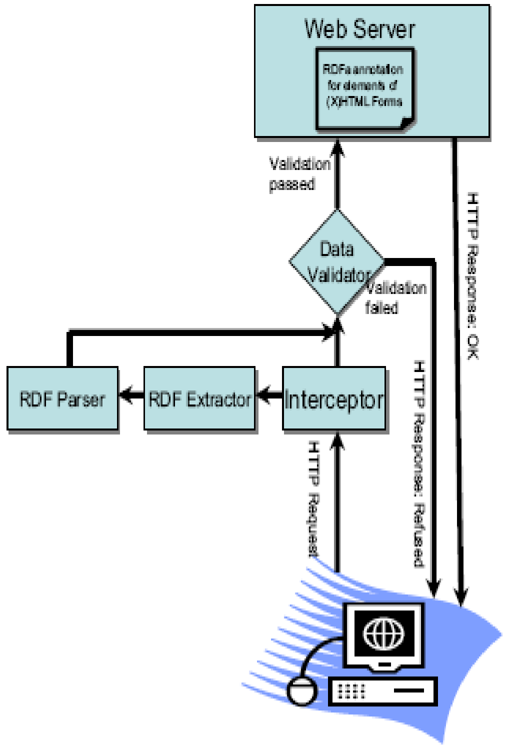

Fig. 1 view of new data validation service Architecture

4.1 Architecture of New Data Validation Service Working Steps:

- End User Requests XHTML Form.
- Interceptor component intercepts each HTTP request at the server-side before the request arrives to web server application for processing.
- Extracting the RDF annotations from RDF ontology vocabulary using the online RDF extractor.
- Invoking the validator component to validate all user inputs.
- If the validation is correct then the request sends to web server application for processing, otherwise, the request is refused.

4.2 Overview of the future framework architecture.

This framework consists of five components:

RDF annotation for elements of web pages: An RDF document is a set of triples of the form (subject, predicate, object). Currently, many documents are explained via RDF due to its simple data model and its formal semantics. For example, It is embedded in (X)HTML web pages using the RDF language.

2. **Interceptor:** Intermediates between the server and client machines by managing the HTTP requests. It intercepts HTTP request, checks the availability of HTTP request on the designated directories of web server, and invokes the RDF extractor.

3. **RDF extractor**: The online RDF distiller is used to extract the RDF annotation from the (X)HTML web page .

4. **RDF parser:** Parses the form inputs and their attributes for validation process.

5. **Data validator:** when the description is extracted using RDF extractor, the validator takes the user inputs for validation process. The validation process checks to see if the value of user input is satisfied the conditions of its attributes (such as length, data type, minimum length, and if the value contains code or special characters) the since it is used. If the integrity check passes, the web content is sent to the running process straight away. If it fails, it is refused the user request.

5. Implementation of New Data Validation Service

The proposed service (NDVS) is implemented in Java using JBuilder (2007) and Java Servlet and filters. The web servers used are Apache 1.3.20 running on MS Windows Server 2003, and Apache Tomcat 5.01 on MS Windows Server 2003. As far as performance is concerned, NDVS is able to prevent infinite number of application attacks. NDVS Of three major components: HTTP Interceptor Mechanism, RDF parser , and Data validator.

1. **HTTP Interceptor Mechanism:** The HTTP Interceptor takes advantage of the fact that browser requests are directed at both a specific host and a specific port. User Create a JSP program, the Tomcat server listens on port 8081. The utility listens for browser requests on a default port 80 and redirects to Tomcat. Responses coming to this mechanism are both sent to the client on port 80.
2. **RDF parser:** It is written in Java programming language to parse the form inputs and their attributes. Each form input is parsed, the id of input is sent to the Data validator mechanism. It should be noted the attributes of each input also is sent to the Data validator mechanism.
3. **Data validator:** when the description is extracted using RDF extractor, the validator takes the user inputs for validation process ,If the integrity check passes, the web content is sent to the running process straight away. If it fails, it is refused the user request.

6. Related Work

A number of researchers are developing solutions to address this problem. For example, Scott and Sharp proposed a gateway model which is an application-level firewall on a server for checking invalid user inputs and detecting malicious script (e.g. SQL injection attack and cross-site scripting attack). This approach offers protection through the enforcement of a number of defined policies, but fails to assess the code itself or to identify the actual weaknesses. They have developed a security policy description language (SPDL) based on XML to describe a set of validation constraints and transformation rules. This language is translated into code by a policy compiler, which is sent to a security gateway on a server. The gateway analyzes the request and augments it with a Message Authentication Code (MAC).

7. Conclusions and further work

Because of the possibility of bypassing input validation either on client-side or server-side, data integrity of web application can be violated even though the communication channel between the server and client is secure. Therefore, we present the proposed web technology-based architecture for new data validation in the web applications. This architecture includes a real-time framework consisting of five components: RDF annotation for elements of web pages, interceptor, RDF extractor, DF parser, and data validator. It might be suggested that the proposed data validation service could provide a detection, and prevention of some web application attacks. In future work, we are intended to optimize the implementation of our solution to increase the effectiveness and performance. Furthermore, we will investigate a number of experiments for security and performance objectives.

8. References

[1] Acunetix. Web applications: What are they? what of them?., 2007. http://www.acunetix.com/websitesecurity/ webapplications.htm, Accessed Data: 15/2/2007.

[2] C. Brabrand, A. Moller, M. Ricky, and M. I. Schwartzbach. PowerForms: Declarative client-side formfield validation. *World Wide.*

[3] H. Chen and D. Wagner. Mops: an infrastructure for examining security properties of software. In *In Proceedingsof the 9th ACM Conference on Computer and Communications Security*, pages 235–244. ACM Press, 2002.

[4] B. Gehling and D. Stankard. eCommerce security. In *Proceedings of Information Security Curriculum Development (InfoSecCD) Conference ´05*, pages 32–37, Kennesaw, GA, USA, Sep 23–24 2005.

[5] Open Web Application Security Project. The Ten Most Critical Web Application Security Vulnerabilities. Version 1.1, January 13 2003.

[7] F. Ricca and P. Tonella. Analysis and testing of web applications. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 25–34, Washington, DC, USA, 2001. IEEE Computer Society.

[9] D. Scott and R. Sharp. Specifying and Enforcing Application-Level Web Security Policies. *IEEE. Knowl. Data Eng*, 15(4):771–783, 2003.

[10] S. Sedaghat, J. Pieprzyk, and E. Vossough. On-thefly web content integrity check boosts users' confidence. *Commun. ACM*, 45(11):33–37, 2002.

[11] J. Tzay, J. Huang, F. Wang, and W. Chu. Constructing an Object-Oriented Architecture for Web Application Testing. *IJ. Information Science and Eng.*, 18(1):59–84, 2002.

[12] CERT. CERT Statistics 1988–2006., Jan 2007.http://www.cert.org/stats.