International Journal of Research in Information Technology (IJRIT)

www.ijrit.com ISSN 2001-5569

Servlet Security

Vipul Asri[1] , Aman Purohit[2] , Nishant Narula[3]

[1,2,3] (Student, Dept of Computer Engineering, Dronacharya College of Engineering, Gurgaon, India)

Abstract— Servlet technology has become the leading web development technology in the recent years due to its safety, portability, efficiency and elegance in design. The servlets are specialized in browsing the web (web documents and various databases) for data mining, information extraction and meaningful presentation inside the web browser. This paper gives a brief introduction about the JAVA servlets and the servlets and the issues related to the security of JAVA servlets. The paper also explains the possible ways from where someone can break the security and how to handle the issues related to the servlets.

Keywords- JAVA, Java Servlet, Internet, SERVLET SECURITY

Introduction

Java Servlets are Java classes that extend the functionality of the server. They are executed on a Web server and act as a connection between the client and the server and their life cycle lasts for as long as there is a connection between the client and the server. JSP technology is a mixture of HTML and Java. It is used to display text on the Web, but has full functionality of Java. The application is created in the three-layer architecture. It is divided into three layers -presentation, business and data layer. Three-layer architecture is used because the applications built into it are easy to transfer and layers are independent. These kinds of servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET. The servlets are object-oriented and therefore highly modular and multifunctional.

Security in Servlets

Security is the science of keeping sensitive information in the hands of authorized users. On the web, there are three important issues:

Authentication:
Being able to verify the identities of the parties involved
Confidentiality:
Ensuring that only the parties involved can understand the communication
Integrity:
Being able to verify that the content of the communication is not changed during transmission.
A client wants to be sure that any information it transmits, such as credit card numbers, is not subject to eavesdropping (confidentiality). The server is also concerned with authentication and confidentiality. If a company is selling a service or providing sensitive information to its own employees, it has a vested interest in making sure that nobody but an authorized user can access it. And both sides need integrity to make sure that whatever information they send gets to the other party unaltered.

HTTP Authentication

The HTTP protocol provides built-in authentication support--called basic authentication--based on a simple challenge/response, username/password model. With this technique, the web server maintains a database of usernames and passwords and identifies certain resources (files, directories, servlets, etc.) as protected. When a user requests access to a protected resource, the server responds with a request for the client's username and password. At this point, the browser usually pops up a dialog box where the user enters the information, and that input is sent back to the server as part of a second authorized request. If the submitted username and password match the information in the server's database, access is Granted. The whole authentication process is handled by the server itself. Basic authentication is very weak. It provides no confidentiality, no integrity, and only the most basic authentication. The problem is that passwords are transmitted over the network, thinly disguised by a well-known and easily reversed Base64 encoding. Anyone monitoring the TCP/IP data stream has full and immediate access to all the information being exchanged, including the username and password. Plus, passwords are often stored on the server in clear text, making them vulnerable to anyone cracking into the server's file system. While it's certainly better than nothing, sites that rely exclusively on basic authentication cannot be considered really secure. A servlet can retrieve information about the server's authentication using two methods : getRemoteUser() and getAuthType().

Custom Authorization

Client authentication is handled by the web server. The server administrator tells the server which resources are to be restricted to the users. To handle these situations, we can use servlets. A servlet can be implemented so that it learns about users from a specially formatted file or a relational database; it can also be written to enforce any security policy you like. Such a servlet can even add, remove, or manipulate user entries--something that isn't supported directly in the Servlet API, except through proprietary server extensions. A servlet uses status codes and HTTP headers to manage its own security policy. The servlet receives encoded authorization credentials in the Authorization header. If it chooses to deny those credentials, it does so by sending the SC_UNAUTHORIZED status code and a WWW-Authenticate header that describes the desired credentials. A web server normally handles these details without involving its servlets, but for a servlet to do its own authorization, it must handle these details itself, while the server is told not to restrict access to the servlet. The Authorization header, if sent by the client, contains the client's username and password. With the basic Authorization scheme, the Authorization header contains the string of "username: password" encoded in Base64. For example, the username of "webmaster" with the password "try2gueSS" is sent in an Authorization header with the value

```
BASIC d2VibWFzdGVyOnRyeTJndWVTU
```

To retrieve the Base64-.encoded username and password, the servlet needs to use a Base64 decoder. Fortunately, there are several freely available decoders. For this servlet, we have chosen to use the sun.misc.BASE64Decoder class that accompanies the JDK. Being in the sun.* hierarchy means it's unsupported and subject to change, but it also means it's probably already on your system.

Form-based Custom Authentication

Form-based authentication allows the developer to control the look and feel of the login authentication screens by customizing the login screen and error pages that an HTTP browser presents to the end user. When form-based authentication is declared, the following actions occur.

1. A client requests access to a protected resource.
2. If the client is unauthenticated, the server redirects the client to a login page.
3. The client submits the login form to the server.
4. The server attempts to authenticate the user.

A. If authentication succeeds, the authenticated user's principal is checked to ensure that it is in a role that is authorized to access the resource. If the user is authorized, the server redirects the client to the resource by using the stored URL path.
B. If authentication fails, the client is forwarded or redirected to an error page.

Role based Authentication

This method of authentication allows a user to authenticate to a role. For authentication to be successful, the user must belong to the role and they must authenticate to each module defined in the Authentication Configuration Service instance configured for that role. For each instance of role-based authentication, the following attributes can be specified:

1. Conflict Resolution Level- This sets a priority level for the Authentication Configuration Service instance defined for different roles that both may contain the same user.
2. Authentication Configuration- This defines the authentication modules configured for the role's authentication process.
3. Login Success URL- This defines the URL to which a user is redirected on successful authentication.
4. Login Failed URL- This defines the URL to which a user is redirected on failed authentication.
5. Authentication Post Processing Classes- This defines the post-authentication interface.

Figures

Fig 1: Working of Servlet

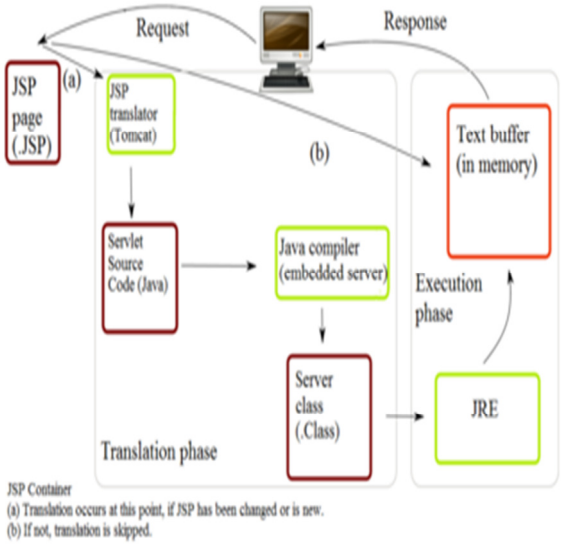
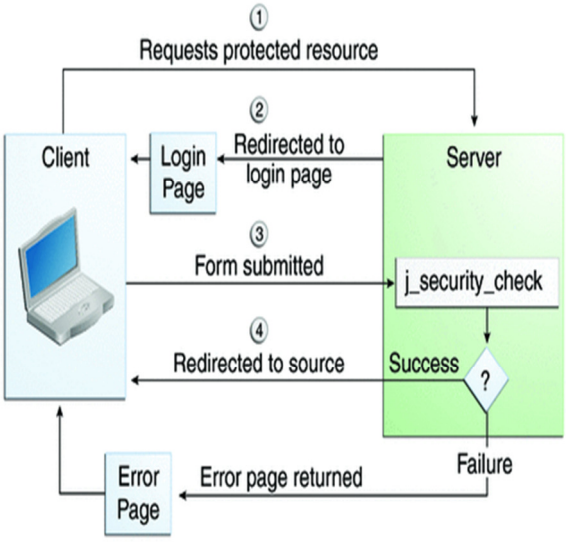


Fig 2: Form Based Authentication



References

- Professional Java Security, *Jess Garms, Danial Somerfield,* ISBN1-861004-25-7
- Java Security Handbook, *Jamie Jaworski, Paul J. Perrone*, ISBN 0-672-31602-1
- http://docstore.mik.ua/orelly/java-ent/servlet/ch08_01.htm
- http://en.wikipedia.org/wiki/File:JSPLife.png
- http://www.jeanquartier.at/fleur/?page_id=195