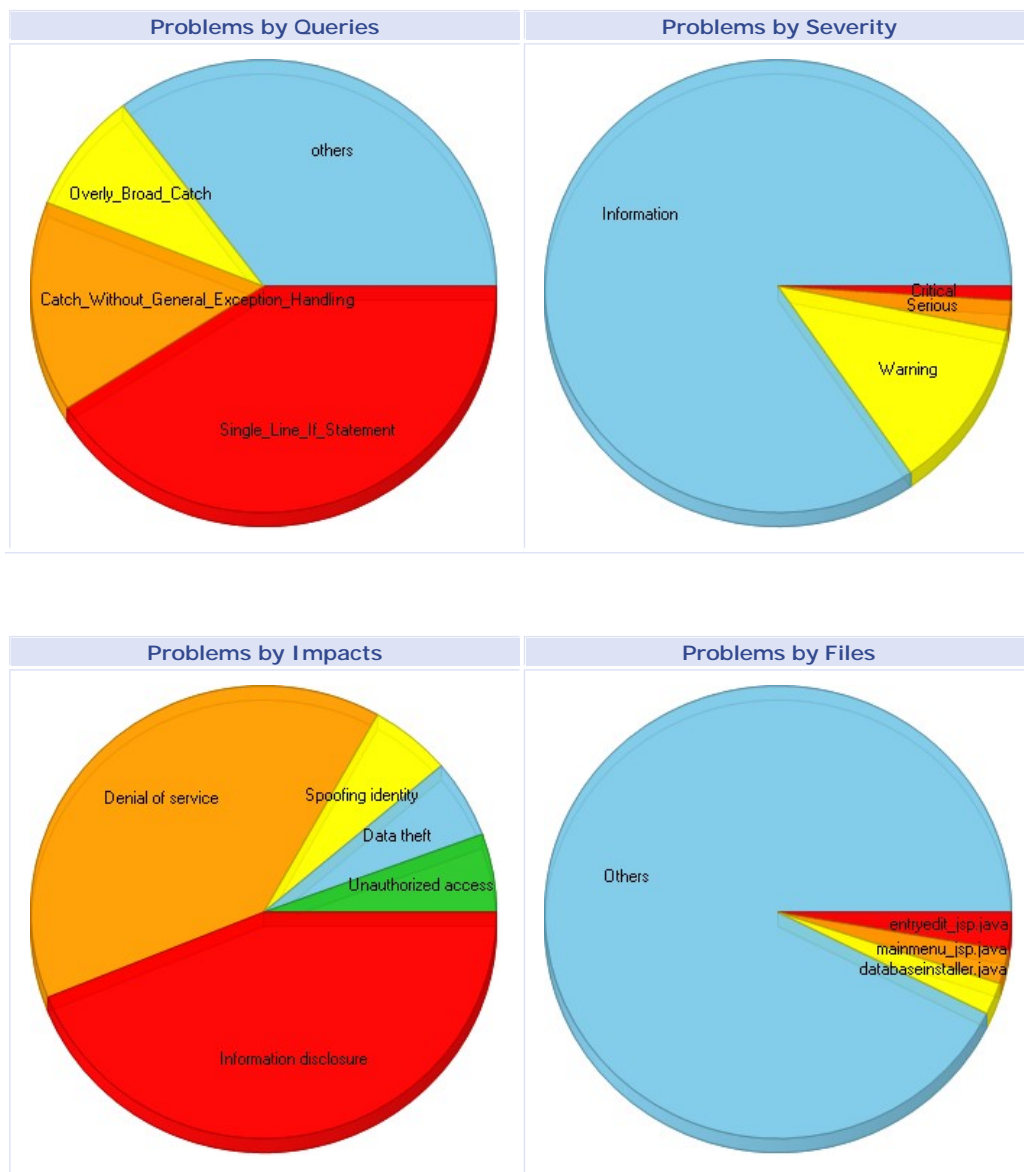


Project Name: c:\documents and settings\sate\desktop\sate_2009\roller
Load Time: 03/09/2009 16:10:00
CxSuite Version: 2.7.5.0



Summary

Query	Group	Problems found	Severity
_Reflected XSS All Clients	Java High Risk	10	
_Stored XSS	Java High Risk	32	
_Verbose Error Reporting	Java Medium Threat	77	
_Files Manipulation	Java Medium Threat	4	
_XSRE	Java Medium Threat	7	
_HttpSplitting	Java Medium Threat	1	
_Log Forgery	Java Low Visibility	129	
_Password Misuse	Java Low Visibility	9	
_DoS by Unreleased Resources	Java Low Visibility	7	
_Hardcoded Password	Java Low Visibility	2	
_Thread Safety Issue	Java Low Visibility	1	
_URL Redirection Attack	Java Low Visibility	8	
_Equals without GetHashCode	Java Low Visibility	12	
_Improper Transaction Handling	Java Low Visibility	12	
_Singleton HttpServlet	Java Low Visibility	19	
_Improper Exception Handling	Java Low Visibility	170	
_Cookie not Sent Over SSL	Java Low Visibility	1	
_Thread Safety Violation In Action Class	Java Struts	138	
_Unclosed Objects	Java Best Coding Practice	62	
_Catch Without General Exception Handling	Java Best Coding Practice	626	
_Pages Without Global Error Handler	Java Best Coding Practice	109	
_Single Line If Statement	Java Best Coding Practice	1701	
_Empty Catch	Java Best Coding Practice	126	
_Threads in WebApp	Java Best Coding Practice	2	
_No Default Case	Java Best Coding Practice	2	
_Magic Numbers	Java Best Coding Practice	39	
_Confusing Naming	Java Best Coding Practice	1	
_Leftover Debug Code	Java Best Coding Practice	16	
_Missing Catch Block	Java Best Coding Practice	29	
_Overly Broad Catch	Java Best Coding Practice	359	
_Overly Broad Throws	Java Best Coding Practice	284	
_Use of System Output Stream	Java Best Coding Practice	61	
_Visible Fields	Java Best Coding Practice	97	

Top 10 Files

File name	Problems found
apache-roller-src-4.0.1\apache-roller-src-4.0.1\atomcat\work\catalina\localhost\roller\org\apache\jsp\web_002dinf\jsps\editor\entryedit_jsp.java	112
apache-roller-src-4.0.1\apache-roller-src-4.0.1\atomcat\work\catalina\localhost\roller\org\apache\jsp\web_002dinf\jsps\core\mainmenu_jsp.java	100
apache-roller-src-4.0.1\apache-roller-src-4.0.1\apps\weblogger\src\java\org\apache\roller\weblogger\business\startup\databaseinstaller.java	95
apache-roller-src-4.0.1\apache-roller-src-4.0.1\atomcat\work\catalina\localhost\roller\org\apache\jsp\web_002dinf\jsps\editor\entrysidebar_jsp.java	89
apache-roller-src-4.0.1\apache-roller-src-4.0.1\atomcat\work\catalina\localhost\roller\org\apache\jsp\web_002dinf\jsps\editor\themeedit_jsp.java	83
apache-roller-src-4.0.1\apache-roller-src-4.0.1\atomcat\work\catalina\localhost\roller\org\apache\jsp\web_002dinf\jsps\editor\templateedit_jsp.java	74
apache-roller-src-4.0.1\apache-roller-src-4.0.1\apps\weblogger\src\java\org\apache\roller\weblogger\business\jpa\jpasusermanagerimpl.java	65
apache-roller-src-4.0.1\apache-roller-src-4.0.1\apps\weblogger\src\java\org\apache\roller\weblogger\webservices\atomprotocol\rolleratomhandler.java	64
apache-roller-src-4.0.1\apache-roller-src-4.0.1\apps\weblogger\src\java\org\apache\roller\weblogger\util\passwordutility.java	62
apache-roller-src-4.0.1\apache-roller-src-4.0.1\apps\weblogger\src\java\org\apache\roller\weblogger\business\jpa\jpaweblogmanagerimpl.java	61

Cross-site Scripting (XSS)

CWE ID	79
Description	A cross-site scripting weakness occurs when dynamically generated web pages display unvalidated, unfiltered, and unencoded user input allowing an attacker to embed malicious scripts into the generated page. This can be leveraged to execute scripting code as if it came from the site's server on the computer of anyone who uses the site.
Alternate Terms	"CSS" was once used as the acronym for this problem, but this can cause confusion with the "Cascading Style Sheets," so this acronym has declined significantly. Its use is discouraged by CWE.
Likelihood of Exploit	High to Very High
Weakness Ordinality	Resultant (Weakness is typically related to the presence of some other weaknesses)
Causal Nature	Explicit (This is an explicit weakness resulting from behavior of the developer)
Common Consequences	<p>Confidentiality: The most common attack performed with cross-site scripting involves the disclosure of information stored in user cookies.</p> <p>Access control: In some circumstances it may be possible to run arbitrary code on a victim's computer when cross-site scripting is combined with other flaws.</p> <p>If successful, cross-site scripting vulnerabilities can be exploited to manipulate or steal cookies, create requests that can be mistaken for those of a valid user, compromise confidential information, or execute malicious code on the end user systems for a variety of nefarious purposes.</p>
Potential Mitigations	<p>Carefully check each input parameter against a rigorous positive specification (white list) defining the specific characters and format allowed. All input should be sanitized, not just parameters that the user is supposed to specify, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so forth. A common mistake that leads to continuing XSS vulnerabilities is to validate only fields that are expected to be redisplayed by the site. We often encounter data from the request that is reflected by the application server or the application that the development team did not anticipate. Also, a field that is not currently reflected may be used by a future developer. Therefore, validating ALL parts of the HTTP request is recommended.</p> <p>This involves "HTML Entity Encoding" all non-alphanumeric characters from data that was received from the user and is now being written to the request.</p> <p>With Struts, you should write all data from form beans with the bean's filter attribute set to true.</p> <p>To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as Internet Explorer), this attribute prevents the user's session cookie from being accessed by client-side scripts, including scripts inserted due to a XSS attack.</p>
Demonstrative Examples	<p>The following JSP code segment reads an employee ID, eid, from an HTTP request and displays it to the user.</p> <p>JSP Example:</p> <pre><% String eid = request.getParameter("eid"); %> ... Employee ID: <%= eid %></pre> <p>The following ASP.NET code segment reads an employee ID number from an HTTP request and displays it to the user.</p> <p>ASP.NET Example:</p> <pre>protected System.Web.UI.WebControls.TextBox Login; protected System.Web.UI.WebControls.Label EmployeeID; ... EmployeeID.Text = Login.Text;</pre>

The following ASP.NET code segment reads an employee ID number from an HTTP request and displays it to the user.

ASP.NET Example:

```
protected System.Web.UI.WebControls.TextBox Login;
protected System.Web.UI.WebControls.Label EmployeeID;
...
EmployeeID.Text = Login.Text;
```

The code in this example operates correctly if eid contains only standard alphanumeric text. If eid has a value that includes meta-characters or source code, then the code will be executed by the web browser as it displays the HTTP response. Initially this might not appear to be much of a vulnerability. After all, why would someone enter a URL that causes malicious code to run on their own computer? The real danger is that an attacker will create the malicious URL, then use e-mail or social engineering tricks to lure victims into visiting a link to the URL. When victims click the link, they unwittingly reflect the malicious content through the vulnerable web application back to their own computers. This mechanism of exploiting vulnerable web applications is known as Reflected XSS.

The following JSP code segment queries a database for an employee with a given ID and prints the corresponding employee's name.

JSP Example:

```
<%...
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp where id="+eid);
if (rs != null) {
    rs.next();
    String name = rs.getString("name");
}%>

Employee Name: <%= name %>
```

The following ASP.NET code segment queries a database for an employee with a given employee ID and prints the name corresponding with the ID.

ASP.NET Example:

```
protected System.Web.UI.WebControls.Label EmployeeName;
...
string query = "select * from emp where id=" + eid;
sda = new SqlDataAdapter(query, conn);
sda.Fill(dt);
string name = dt.Rows[0]["Name"];
...
EmployeeName.Text = name;
```

This code functions correctly when the values of name are well-behaved, but it does nothing to prevent exploits if they are not. Again, this code can appear less dangerous because the value of name is read from a database, whose contents are apparently managed by the application. However, if the value of name originates from user-supplied data, then the database can be a conduit for malicious content. Without proper input validation on all data stored in the database, an attacker can execute malicious commands in the user's web browser. This type of exploit, known as Stored XSS, is particularly insidious because the indirection caused by the data store makes it more difficult to identify the threat and increases the possibility that the attack will affect multiple users. XSS got its start in this form with web sites that offered a "guestbook" to visitors. Attackers would include JavaScript in their guestbook entries, and all subsequent visitors to the guestbook page would execute the malicious code. As the examples demonstrate, XSS vulnerabilities are caused by code that includes unvalidated data in an HTTP response. There are three vectors by which an XSS attack can reach a victim: * As in the previous example, data is read directly from the HTTP request and reflected back in the HTTP response. Reflected XSS exploits occur when an attacker causes a user to supply dangerous content to a vulnerable web application, which is then reflected back to the user and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to victims. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces victims to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the user, the content is executed and proceeds to transfer private information, such as cookies that may include session information, from the user's machine to the attacker or perform other nefarious activities. * As in this example, the application stores dangerous data in a database or other trusted data store. The dangerous data is subsequently read back into the application and included in dynamic content. Stored XSS exploits occur when an attacker injects dangerous content into a data store that is later read and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user. * A source outside the application stores dangerous data in a database or other data store, and the dangerous data is subsequently read back into the application as trusted data and included in dynamic content.

Context Notes

Cross-site scripting (XSS) vulnerabilities occur when an attacker uses a web application to send malicious code, generally JavaScript, to a different end user. When a web application uses input from a user in the output it generates without filtering it, an attacker can insert an attack in that input and the web application sends the attack to other users. The end user trusts the web application, and the attacks exploit that trust to do things that would not normally be allowed. Attackers frequently use a variety of methods to encode the malicious portion of the tag, such as using Unicode, so the request looks less suspicious to the user. XSS attacks can generally be categorized into two categories: stored and reflected. Stored attacks are those where the injected code is permanently stored on the target servers in a database, message forum, visitor log, and so forth. Reflected attacks are those where the injected code takes another route to the victim, such as in an email message, or on some other server. When a user is tricked into clicking a link or submitting a form, the injected code travels to the vulnerable web server, which reflects the attack back to the user's browser. The browser then executes the code because it came from a 'trusted' server. For a reflected XSS attack to work, the victim must submit the attack to the server. This is still a very dangerous attack given the number of possible ways to trick a victim into submitting such a malicious request, including clicking a link on a malicious Web site, in an email, or in an inner-office posting. XSS flaws are very likely in web applications, as they require a great deal of developer discipline to avoid them in most applications. It is relatively easy for an attacker to find XSS vulnerabilities. Some of these vulnerabilities can be found using scanners, and some exist in older web application servers. The consequence of an XSS attack is the same regardless of whether it is stored or reflected. The difference is in how the payload arrives at the server. XSS can cause a variety of problems for the end user that range in severity from an annoyance to complete account compromise. The most severe XSS attacks involve disclosure of the user's session cookie, which allows an attacker to hijack the user's session and take over their account. Other damaging attacks include the disclosure of end user files, installation of Trojan horse programs, redirecting the user to some other page or site, and modifying presentation of content. Cross-site scripting (XSS) vulnerabilities occur when: 1. Data enters a Web application through an untrusted source, most frequently a web request. 2. The data is included in dynamic content that is sent to a web user without being validated for malicious code. The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data like cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

Cross-site scripting attacks can occur wherever an untrusted user has the ability to publish content to a trusted web site. Typically, a malicious user will craft a client-side script, which -- when parsed by a web browser -- performs some activity (such as sending all site cookies to a given E-mail address). If the input is unchecked, this script will be loaded and run by each user visiting the web site. Since the site requesting to run the script has access to the cookies in question, the malicious script does also. There are several other possible attacks, such as running "Active X" controls (under Microsoft Internet Explorer) from sites that a user perceives as trustworthy; cookie theft is however by far the most common. All of these attacks are easily prevented by ensuring that no script tags -- or for good measure, HTML tags at all -- are allowed in data to be posted publicly.

Cross-site scripting attacks may occur anywhere that possibly malicious users are allowed to post unregulated material to a trusted web site for the consumption of other valid users. The most common example can be found in bulletin-board web sites which provide web based mailing list-style functionality.

When used in conjunction with Cross-Site Request Forgery (CSRF), XSS worms can cause significant damage in web sites with large user populations.

References

Jeremiah Grossman, Robert "RSnake" Hansen, Petko "pdp" D. Petkov, Anton Rager and Seth Fogie. "XSS Attacks".

Syngress. 2007.

M. Howard and D. LeBlanc. "Writing Secure Code". 2nd Edition. Microsoft. 2003.

Node Relationships	ChildOf - Injection (74)
	ResultsIn - Mobile Code: Invoking Untrusted Mobile Code (494)
	Peer - Cross-Site Request Forgery (CSRF) (352)
	ChildOf - Weaknesses in OWASP Top Ten (629)
	ChildOf - Weaknesses Used by NVD (635)
	ParentOf - Basic XSS (80)
	ParentOf - XSS in Error Pages (81)
	ParentOf - Script in IMG Tags (82)
	ParentOf - XSS using Script in Attributes (83)
	ParentOf - XSS using Script Via Encoded URI Schemes (84)
	ParentOf - Doubled Character XSS Manipulations (85)
	ParentOf - Invalid Characters in Identifiers (86)
	ParentOf - Alternate XSS Syntax (87)
Source Taxonomies	PLOVER - Cross-site scripting (XSS)
	7 Pernicious Kingdoms - Cross-site Scripting
	CLASP - Cross-site scripting
Applicable Platforms	All

Path 1:

Query Name - **Reflected_XSS_All_Clients**

Severity -  **Critical**

```
82. public boolean handleRequest(HttpServletRequest request, HttpServletResponse response) //multiplanetrequestmapper.java
    ...
162. String redirectUrl = request.getRequestURI() + "/";
    ...
164. redirectUrl += "?" + request.getQueryString();
    ...
167. response.sendRedirect(redirectUrl);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Reflected_XSS_All_Clients**

Severity -  **Critical**

```
71. public void myPrepare() //planetsubscriptions.java
    ...
81. setGroup(pmgr.getGroup(getPlanet(), getGroupHandle()));

213. public PlanetGroup getGroup(Planet planet, String handle) throws PlanetException //jpaplanetmanagerimpl.java
    ...
216. q.setParameter(2, handle);
    ...
218. return (PlanetGroup)q.getSingleResult();

71. public void myPrepare() //planetsubscriptions.java
    ...
81. setGroup(pmgr.getGroup(getPlanet(), getGroupHandle()));

219. public void setGroup(PlanetGroup group) //planetsubscriptions.java
    ...
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Reflected_XSS_All_Clients**

Severity -  **Critical**

```
76. public void doFilter(ServletRequest request, ServletResponse response, //schemeenforcementfilter.java
    ...
94. redirect += req.getRequestURI();
    ...
97. redirect += "?" + req.getQueryString();
    ...
100. res.sendRedirect(redirect);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Reflected_XSS_All_Clients**

Severity -  Critical

```
76. public void doFilter(ServletRequest request, ServletResponse response, //schemeenforcementfilter.java
    ...
    110. redirect += req.getRequestURI();
    ...
    113. redirect += "?" + req.getQueryString();
    ...
    116. res.sendRedirect(redirect);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Reflected_XSS_All_Clients**

Severity -  Critical

```
91. public boolean handleRequest(HttpServletRequest request, HttpServletResponse response) //weblogrequestmapper.java
    ...
    188. String redirectUrl = request.getRequestURI() + "/";
    ...
    190. redirectUrl += "?" + request.getQueryString();
    ...
    193. response.sendRedirect(redirectUrl);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Reflected_XSS_All_Clients**

Severity -  Critical

```
39. public String getHtml(HttpServletRequest request) //mathcommentauthenticator.java
    ...
    54. answer = request.getParameter("answer");
    55. answer = (answer == null) ? "" : answer;
    ...
    72. sb.append(answer);
    73. sb.append("\n /></p>");
    ...
    75. return sb.toString();
```



```
57. public void doGet(HttpServletRequest request, HttpServletResponse response) //commentauthenticatorservlet.java
    ...
    68. out.println(this.authenticator.getHtml(request));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Reflected_XSS_All_Clients**

Severity -  Critical

```
82. public boolean handleRequest(HttpServletRequest request, HttpServletResponse response) //multiplanetrequestmapper.java
    ...
    164. redirectUrl += "?" + request.getQueryString();
    ...
    167. response.sendRedirect(redirectUrl);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Reflected_XSS_All_Clients**

Severity -  Critical

```
76. public void doFilter(ServletRequest request, ServletResponse response, //schemeenforcementfilter.java
    ...
    97. redirect += "?" + req.getQueryString();
    ...
    100. res.sendRedirect(redirect);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Reflected_XSS_All_Clients**

Severity -  Critical

```

76. public void doFilter(ServletRequest request, ServletResponse response, //schemeenforcementfilter.java
    ...
    113. redirect += "?" + req.getQueryString();
    ...
    116. res.sendRedirect(redirect);

```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Reflected_XSS_All_Clients**

Severity -  **Critical**

```

91. public boolean handleRequest(HttpServletRequest request, HttpServletResponse response) //weblogrequestmapper.java
    ...
    190. redirectUrl += "?" + request.getQueryString();
    ...
    193. response.sendRedirect(redirectUrl);

```

[\[Table of Contents\]](#) [\[Description\]](#)

Cross-site Scripting (XSS)	
CWE ID	79
Description	A cross-site scripting weakness occurs when dynamically generated web pages display unvalidated, unfiltered, and unencoded user input allowing an attacker to embed malicious scripts into the generated page. This can be leveraged to execute scripting code as if it came from the site's server on the computer of anyone who uses the site.
Alternate Terms	"CSS" was once used as the acronym for this problem, but this can cause confusion with the "Cascading Style Sheets," so this acronym has declined significantly. Its use is discouraged by CWE.
Likelihood of Exploit	High to Very High
Weakness Ordinality	Resultant (Weakness is typically related to the presence of some other weaknesses)
Causal Nature	Explicit (This is an explicit weakness resulting from behavior of the developer)
Common Consequences	<p>Confidentiality: The most common attack performed with cross-site scripting involves the disclosure of information stored in user cookies.</p> <p>Access control: In some circumstances it may be possible to run arbitrary code on a victim's computer when cross-site scripting is combined with other flaws.</p> <p>If successful, cross-site scripting vulnerabilities can be exploited to manipulate or steal cookies, create requests that can be mistaken for those of a valid user, compromise confidential information, or execute malicious code on the end user systems for a variety of nefarious purposes.</p>
Potential Mitigations	<p>Carefully check each input parameter against a rigorous positive specification (white list) defining the specific characters and format allowed. All input should be sanitized, not just parameters that the user is supposed to specify, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so forth. A common mistake that leads to continuing XSS vulnerabilities is to validate only fields that are expected to be redisplayed by the site. We often encounter data from the request that is reflected by the application server or the application that the development team did not anticipate. Also, a field that is not currently reflected may be used by a future developer. Therefore, validating ALL parts of the HTTP request is recommended.</p> <p>This involves "HTML Entity Encoding" all non-alphanumeric characters from data that was received from the user and is now being written to the request.</p> <p>With Struts, you should write all data from form beans with the bean's filter attribute set to true.</p> <p>To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as Internet Explorer), this attribute prevents the user's session cookie from being accessed by client-side scripts, including scripts inserted due to a XSS attack.</p>
Demonstrative Examples	<p>The following JSP code segment reads an employee ID, eid, from an HTTP request and displays it to the user.</p> <p>JSP Example:</p> <pre> <% String eid = request.getParameter("eid"); %> ... Employee ID: <%= eid %> </pre> <p>The following ASP.NET code segment reads an employee ID number from an HTTP request and displays it to the user.</p> <p>ASP.NET Example:</p> <pre> protected System.Web.UI.WebControls.TextBox Login; protected System.Web.UI.WebControls.Label EmployeeID; ... EmployeeID.Text = Login.Text; </pre>

The code in this example operates correctly if eid contains only standard alphanumeric text. If eid has a value that includes meta-characters or source code, then the code will be executed by the web browser as it displays the HTTP response. Initially this might not appear to be much of a vulnerability. After all, why would someone enter a URL that causes malicious code to run on their own computer? The real danger is that an attacker will create the

malicious URL, then use e-mail or social engineering tricks to lure victims into visiting a link to the URL. When victims click the link, they unwittingly reflect the malicious content through the vulnerable web application back to their own computers. This mechanism of exploiting vulnerable web applications is known as Reflected XSS.

The following JSP code segment queries a database for an employee with a given ID and prints the corresponding employee's name.

JSP Example:

```
<%...
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp where id="+eid);
if (rs != null) {
    rs.next();
    String name = rs.getString("name");
}%>

Employee Name: <%= name %>
```

The following ASP.NET code segment queries a database for an employee with a given employee ID and prints the name corresponding with the ID.

ASP.NET Example:

```
protected System.Web.UI.WebControls.Label EmployeeName;
...
string query = "select * from emp where id=" + eid;
sda = new SqlDataAdapter(query, conn);
sda.Fill(dt);
string name = dt.Rows[0]["Name"];
...
EmployeeName.Text = name;
```

This code functions correctly when the values of name are well-behaved, but it does nothing to prevent exploits if they are not. Again, this code can appear less dangerous because the value of name is read from a database, whose contents are apparently managed by the application. However, if the value of name originates from user-supplied data, then the database can be a conduit for malicious content. Without proper input validation on all data stored in the database, an attacker can execute malicious commands in the user's web browser. This type of exploit, known as Stored XSS, is particularly insidious because the indirection caused by the data store makes it more difficult to identify the threat and increases the possibility that the attack will affect multiple users. XSS got its start in this form with web sites that offered a "guestbook" to visitors. Attackers would include JavaScript in their guestbook entries, and all subsequent visitors to the guestbook page would execute the malicious code. As the examples demonstrate, XSS vulnerabilities are caused by code that includes unvalidated data in an HTTP response. There are three vectors by which an XSS attack can reach a victim: * As in the previous example, data is read directly from the HTTP request and reflected back in the HTTP response. Reflected XSS exploits occur when an attacker causes a user to supply dangerous content to a vulnerable web application, which is then reflected back to the user and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to victims. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces victims to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the user, the content is executed and proceeds to transfer private information, such as cookies that may include session information, from the user's machine to the attacker or perform other nefarious activities. * As in this example, the application stores dangerous data in a database or other trusted data store. The dangerous data is subsequently read back into the application and included in dynamic content. Stored XSS exploits occur when an attacker injects dangerous content into a data store that is later read and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user. * A source outside the application stores dangerous data in a database or other data store, and the dangerous data is subsequently read back into the application as trusted data and included in dynamic content.

Context Notes

Cross-site scripting (XSS) vulnerabilities occur when an attacker uses a web application to send malicious code, generally JavaScript, to a different end user. When a web application uses input from a user in the output it generates without filtering it, an attacker can insert an attack in that input and the web application sends the attack to other users. The end user trusts the web application, and the attacks exploit that trust to do things that would not normally be allowed. Attackers frequently use a variety of methods to encode the malicious portion of the tag, such as using Unicode, so the request looks less suspicious to the user. XSS attacks can generally be categorized into two categories: stored and reflected. Stored attacks are those where the injected code is permanently stored on the target servers in a database, message forum, visitor log, and so forth. Reflected attacks are those where the injected code takes another route to the victim, such as in an email message, or on some other server. When a user is tricked into clicking a link or submitting a form, the injected code travels to the vulnerable web server, which reflects the attack back to the user's browser. The browser then executes the code because it came from a 'trusted' server. For a reflected XSS attack to work, the victim must submit the attack to the server. This is still a very dangerous attack given the number of possible ways to trick a victim into submitting such a malicious request, including clicking a link on a malicious Web site, in an email, or in an inner-office posting. XSS flaws are very likely in web applications, as they require a great deal of developer discipline to avoid them in most applications. It is relatively easy for an attacker to find XSS vulnerabilities. Some of these vulnerabilities can be found using scanners, and some exist in older web application servers. The consequence of an XSS attack is the same regardless of whether it is stored or reflected. The difference is in how the payload arrives at the server. XSS can cause a variety of problems for the end user that range in severity from an annoyance to complete account compromise. The most severe XSS attacks involve disclosure of the user's session cookie, which allows an attacker to hijack the user's session and take over their account. Other damaging attacks include the disclosure of end user files, installation of Trojan horse programs, redirecting the user to some other page or site, and modifying presentation of content. Cross-site scripting (XSS) vulnerabilities occur when: 1. Data enters a Web application through an untrusted source, most frequently a web request. 2. The data is included in dynamic

content that is sent to a web user without being validated for malicious code. The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data like cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

Cross-site scripting attacks can occur wherever an untrusted user has the ability to publish content to a trusted web site. Typically, a malicious user will craft a client-side script, which -- when parsed by a web browser -- performs some activity (such as sending all site cookies to a given E-mail address). If the input is unchecked, this script will be loaded and run by each user visiting the web site. Since the site requesting to run the script has access to the cookies in question, the malicious script does also. There are several other possible attacks, such as running "Active X" controls (under Microsoft Internet Explorer) from sites that a user perceives as trustworthy; cookie theft is however by far the most common. All of these attacks are easily prevented by ensuring that no script tags -- or for good measure, HTML tags at all -- are allowed in data to be posted publicly.

Cross-site scripting attacks may occur anywhere that possibly malicious users are allowed to post unregulated material to a trusted web site for the consumption of other valid users. The most common example can be found in bulletin-board web sites which provide web based mailing list-style functionality.

When used in conjunction with Cross-Site Request Forgery (CSRF), XSS worms can cause significant damage in web sites with large user populations.

References

Jeremiah Grossman, Robert "RSnake" Hansen, Petko "pdp" D. Petkov, Anton Rager and Seth Fogie. "XSS Attacks". Syngress. 2007.
M. Howard and D. LeBlanc. "Writing Secure Code". 2nd Edition. Microsoft. 2003.

Node Relationships

ChildOf - [Injection](#) (74)
ResultsIn - [Mobile Code: Invoking Untrusted Mobile Code](#) (494)
Peer - [Cross-Site Request Forgery \(CSRF\)](#) (352)
ChildOf - [Weaknesses in OWASP Top Ten](#) (629)
ChildOf - [Weaknesses Used by NVD](#) (635)
ParentOf - [Basic XSS](#) (80)
ParentOf - [XSS in Error Pages](#) (81)
ParentOf - [Script in IMG Tags](#) (82)
ParentOf - [XSS using Script in Attributes](#) (83)
ParentOf - [XSS using Script Via Encoded URI Schemes](#) (84)
ParentOf - [Doubled Character XSS Manipulations](#) (85)
ParentOf - [Invalid Characters in Identifiers](#) (86)
ParentOf - [Alternate XSS Syntax](#) (87)

Source Taxonomies

PLOVER - Cross-site scripting (XSS)
7 Pernicious Kingdoms - Cross-site Scripting
CLASP - Cross-site scripting

Applicable Platforms

All

Path 1:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
193. public List getPlanets() throws PlanetException //jpaplanetmanagerimpl.java
    | 194. return (List)strategy.getNamedQuery("Planet.getAll").getResultList();
```



```
50. public String execute() //planetslist.java
    | ...
    | 53. List planets = pMgr.getPlanets();
    | ...
    | 55. setPlanets(planets);
```



```
103. private void setPlanets(Collection planets) //planetslist.java
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
77. public Map getProperties() throws PlanetException //jpapropertiesmanagerimpl.java
    | ...
    | 80. List list = strategy.getNamedQuery("RuntimeConfigProperty.getAll").getResultList();
    | ...
    | 89. Iterator it = list.iterator();
    | ...
    | 91. prop = (RuntimeConfigProperty) it.next();
    | 92. props.put(prop.getName(), prop);
    | ...
    | 95. return props;
```



```
75. public void myPrepare() //planetconfig.java
    | ...
    | 79. setProperties(pMgr.getProperties());
```

169. public void setProperties(Map properties) //planetconfig.java

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
179. public Planet getPlanet(String handle) throws PlanetException //jpaplanetmanagerimpl.java
    ...
183. return (Planet)q.getSingleResult();
```

```
40. public Planet getPlanet() //planetuiaction.java
    ...
44. planet = pmgr.getPlanet(DEFAULT_PLANET_HANDLE);
    ...
49. return planet;
```

```
71. public void myPrepare() //planetsubscriptions.java
    ...
81. setGroup(pmgr.getGroup(getPlanet(), getGroupHandle()));
```

```
213. public PlanetGroup getGroup(Planet planet, String handle) throws PlanetException //jpaplanetmanagerimpl.java
    ...
215. q.setParameter(1, planet.getHandle());
216. q.setParameter(2, handle);
    ...
218. return (PlanetGroup)q.getSingleResult();
```

```
71. public void myPrepare() //planetsubscriptions.java
    ...
81. setGroup(pmgr.getGroup(getPlanet(), getGroupHandle()));
```

```
219. public void setGroup(PlanetGroup group) //planetsubscriptions.java
    ,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
847. public List getComments( //jpaweblogmanagerimpl.java
    ...
924. return query.getResultList();
```

```
88. public void loadComments() //globalcommentmanagement.java
    ...
94. List rawComments = wmgr.getComments(
    ...
105. comments.addAll(rawComments);
    ...
113. setFirstComment((WeblogEntryComment)comments.get(0));
```

```
361. public void setFirstComment(WeblogEntryComment firstComment) //globalcommentmanagement.java
    ,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
847. public List getComments( //jpaweblogmanagerimpl.java
    ...
924. return query.getResultList();
```

```
88. public void loadComments() //globalcommentmanagement.java
    ...
    94. List rawComments = wmgr.getComments(
    ...
    105. comments.addAll(rawComments);
    ...
    114. setLastComment((WeblogEntryComment)comments.get(comments.size()-1));
```



```
369. public void setLastComment(WeblogEntryComment lastComment) //globalcommentmanagement.java
    ,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
847. public List getComments() //jpaweblogmanagerimpl.java
    ...
    924. return query.getResultList();
```



```
88. public void loadComments() //globalcommentmanagement.java
    ...
    94. List rawComments = wmgr.getComments(
    ...
    105. comments.addAll(rawComments);
    ...
    124. setPager(new CommentsPager(baseUrl, getBean().getPage(), comments, hasMore));
```



```
50. public CommentsPager(String url, int page, List<WeblogEntryComment> comments, boolean hasMore) //commentspager.java
    ...
    53. this.items = comments;
```



```
88. public void loadComments() //globalcommentmanagement.java
    ...
    124. setPager(new CommentsPager(baseUrl, getBean().getPage(), comments, hasMore));
```



```
377. public void setPager(CommentsPager pager) //globalcommentmanagement.java
    ,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
114. public Map getProperties() throws WebloggerException //jpapropertiesmanagerimpl.java
    ...
    117. List list = (List) strategy.getNamedQuery("RuntimeConfigProperty.getAll").getResultList();
    ...
    127. prop = (RuntimeConfigProperty) it.next();
    128. props.put(prop.getName(), prop);
    ...
    131. return props;
```



```
87. public void myPrepare() //globalconfig.java
    ...
    91. setProperties(mgr.getProperties());
```



```
223. public void setProperties(Map properties) //globalconfig.java
    ,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
634. public User getUserByUserName(String userName, Boolean enabled) //jpausermanagerimpl.java
    ...
    675. user = (User)query.getSingleResult();
```

```
...
686. return user;
```

```
67. public void myPrepare() //modifyuser.java
```

```
...
75. setUser(mgr.getUserByUserName(getUserName(), null));
```

```
199. public void setUser(User user) //modifyuser.java
```

```
,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
173. public List getCommonPingTargets() //jpapingtargetmanagerimpl.java
```

```
...
177. return q.getResultList();
```

```
59. public void loadPingTargets() //commonpingtargets.java
```

```
...
62. setPingTargets(pingTargetMgr.getCommonPingTargets());
```

```
135. public void setPingTargets(List pingTargets) //pingtargetsbase.java
```

```
,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
357. public List getAllFolders(Weblog website) //jpabookmarkmanagerimpl.java
```

```
...
364. return q.getResultList();
```

```
101. public String execute() //bookmarks.java
```

```
...
109. List<WeblogBookmarkFolder> folders = bmgr.getAllFolders(getActionWeblog());
...
112. allFolders.add(fd);
...
134. setAllFolders(allFolders);
```

```
286. public void setAllFolders(Set allFolders) //bookmarks.java
```

```
,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 11:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
343. public WeblogBookmarkFolder getRootFolder(Weblog website) //jpabookmarkmanagerimpl.java
```

```
...
351. return (WeblogBookmarkFolder)q.getSingleResult();
```

```
82. public void myPrepare() //bookmarks.java
```

```
...
89. setFolder(bmgr.getRootFolder(getActionWeblog()));
```

```
294. public void setFolder(WeblogBookmarkFolder folder) //bookmarks.java
```

```
,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 12:

Query Name - **Stored_XSS**

Severity -  Critical

```
463. public WeblogCategory getRootWeblogCategory(Weblog website) //jpaweblogmanagerimpl.java
    ...
    472. return (WeblogCategory)q.getSingleResult();
```



```
77. public void myPrepare() //categories.java
    ...
    84. setCategory(wmgr.getRootWeblogCategory(getActionWeblog()));
```



```
181. public void setCategory(WeblogCategory category) //categories.java
    ,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 13:

Query Name - **Stored_XSS**

Severity -  Critical

```
481. public List getWeblogCategories(Weblog website, boolean includeRoot) //jpaweblogmanagerimpl.java
    ...
    491. return q.getResultList();
```



```
92. public String execute() //categories.java
    ...
    100. List<WeblogCategory> cats = wmgr.getWeblogCategories(getActionWeblog(), true);
    ...
    103. allCategories.add(cat);
    ...
    125. setAllCategories(allCategories);
```



```
205. public void setAllCategories(Set allCategories) //categories.java
    ,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 14:

Query Name - **Stored_XSS**

Severity -  Critical

```
481. public List getWeblogCategories(Weblog website, boolean includeRoot) //jpaweblogmanagerimpl.java
    ...
    491. return q.getResultList();
```



```
85. public String execute() //categoryremove.java
    ...
    93. List<WeblogCategory> cats = wmgr.getWeblogCategories(getActionWeblog(), true);
    ...
    96. allCategories.add(cat);
    ...
    106. setAllCategories(allCategories);
```



```
177. public void setAllCategories(Set allCategories) //categoryremove.java
    ,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 15:

Query Name - **Stored_XSS**

Severity -  Critical

```
847. public List getComments( //jpaweblogmanagerimpl.java
    ...
    924. return query.getResultList();
```



```
92. public void loadComments() //comments.java
    ...
    105. List rawComments = wmgr.getComments(
    ...
    115. comments.addAll(rawComments);
    ...
```

```
| 122. setFirstComment((WeblogEntryComment)comments.get(0));
```

```
449. public void setFirstComment(WeblogEntryComment firstComment) //comments.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 16:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
847. public List getComments( //jpaweblogmanagerimpl.java
    {
    ...
    924. return query.getResultList();
```

```
92. public void loadComments() //comments.java
    {
    ...
    105. List rawComments = wmgr.getComments(
    ...
    115. comments.addAll(rawComments);
    ...
    123. setLastComment((WeblogEntryComment)comments.get(comments.size()-1));
```

```
457. public void setLastComment(WeblogEntryComment lastComment) //comments.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 17:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
847. public List getComments( //jpaweblogmanagerimpl.java
    {
    ...
    924. return query.getResultList();
```

```
92. public void loadComments() //comments.java
    {
    ...
    105. List rawComments = wmgr.getComments(
    ...
    115. comments.addAll(rawComments);
    ...
    133. setPager(new CommentsPager(baseUrl, getBean().getPage(), comments, hasMore));
```

```
50. public CommentsPager(String url, int page, List<WeblogEntryComment> comments, boolean hasMore) //commentspager.java
    {
    ...
    53. this.items = comments;
```

```
92. public void loadComments() //comments.java
    {
    ...
    133. setPager(new CommentsPager(baseUrl, getBean().getPage(), comments, hasMore));
```

```
465. public void setPager(CommentsPager pager) //comments.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 18:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
511. public List getWeblogEntries( //jpaweblogmanagerimpl.java
    {
    ...
    645. return query.getResultList();
```

```
76. public String execute() //entries.java
    {
    ...
    88. List<WeblogEntry> rawEntries = wmgr.getWeblogEntries(
    ...
    103. entries.addAll(rawEntries);
```

```
...
112. setFirstEntry((WeblogEntry)entries.get(0));
```

```
222. public void setFirstEntry(WeblogEntry firstEntry) //entries.java
{
    ...
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 19:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
511. public List getWeblogEntries( //jpaweblogmanagerimpl.java
{
    ...
645. return query.getResultList();
}
```

```
76. public String execute() //entries.java
{
    ...
88. List<WeblogEntry> rawEntries = wmgr.getWeblogEntries(
    ...
103. entries.addAll(rawEntries);
    ...
113. setLastEntry((WeblogEntry)entries.get(entries.size()-1));
}
```

```
230. public void setLastEntry(WeblogEntry lastEntry) //entries.java
{
    ...
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 20:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
511. public List getWeblogEntries( //jpaweblogmanagerimpl.java
{
    ...
645. return query.getResultList();
}
```

```
76. public String execute() //entries.java
{
    ...
88. List<WeblogEntry> rawEntries = wmgr.getWeblogEntries(
    ...
103. entries.addAll(rawEntries);
    ...
123. setPager(new EntriesPager(baseUrl, getBean().getPage(), entries, hasMore));
}
```

```
50. public EntriesPager(String url, int page, List<WeblogEntry> entries, boolean hasMore) //entriespager.java
{
    ...
53. this.items = entries;
}
```

```
76. public String execute() //entries.java
{
    ...
123. setPager(new EntriesPager(baseUrl, getBean().getPage(), entries, hasMore));
}
```

```
238. public void setPager(EntriesPager pager) //entries.java
{
    ...
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 21:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
173. public List getCommonPingTargets() //jpapingtargetmanagerimpl.java
{
    ...
177. return q.getResultList();
}
```

```
80. public void myPrepare() //pings.java
{
    ...
93. setCommonPingTargets(pingTargetMgr.getCommonPingTargets());
}
```



```
273. public void setCommonPingTargets(List commonPingTargets) //pings.java
    {
        ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 22:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
180. public List getCustomPingTargets(Weblog website) //jpapingtargetmanagerimpl.java
    {
        ...
        185. return q.getResultList();
    }
}
```



```
80. public void myPrepare() //pings.java
    {
        ...
        97. setCustomPingTargets(pingTargetMgr.getCustomPingTargets(getActionWeblog()));
    }
}
```



```
281. public void setCustomPingTargets(List customPingTargets) //pings.java
    {
        ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 23:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
266. public List getTodaysReferers(Weblog website) throws WebloggerException //jpareferermanagerimpl.java
    {
        ...
        270. return q.getResultList();
    }
}
```



```
66. public String execute() //referrers.java
    {
        ...
        71. setReferrers(refmgr.getTodaysReferers(getActionWeblog()));
    }
}
```



```
148. public void setReferrers(List<WeblogReferrer> referrers) //referrers.java
    {
        ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 24:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
891. public WeblogTemplate getPageByLink(Weblog website, String pagelink) //jpausermanagerimpl.java
    {
        ...
        904. return (WeblogTemplate)query.getSingleResult();
    }
}
```



```
65. public void myPrepare() //stylesheetedit.java
    {
        ...
        78. setTemplate(mgr.getPageByLink(getActionWeblog(), stylesheet.getLink()));
    }
}
```



```
221. public void setTemplate(WeblogTemplate template) //stylesheetedit.java
    {
        ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 25:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
958. public List getPages(Weblog website) throws WebloggerException //jpausermanagerimpl.java
    {
        ...
        964. return q.getResultList();
    }
}
```



```
67. public String execute() //templates.java
    {
        ...
        74. List<WeblogTemplate> raw = mgr.getPages(getActionWeblog());
        ...
    }
}
```

```
76. pages.addAll(row);
...
81. setTemplates(pages);
```

```
201. public void setTemplates(List<WeblogTemplate> templates) //templates.java
{
    ...
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 26:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
481. public List getWeblogCategories(Weblog website, boolean includeRoot) //jpaweblogmanagerimpl.java
{
    ...
491. return q.getResultList();
}
```

```
77. public void myPrepare() //weblogconfig.java
{
    ...
83. setWeblogCategories(wmgr.getWeblogCategories(getActionWeblog(), false));
}
```

```
218. public void setWeblogCategories(List weblogCategories) //weblogconfig.java
{
    ...
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 27:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
508. public Weblog getWebsiteByHandle(String handle, Boolean enabled) //jpausermanagerimpl.java
{
    ...
536. website = (Weblog)query.getSingleResult();
    ...
549. return website;
}
```

```
501. public Weblog getWebsiteByHandle(String handle) throws WebloggerException //jpausermanagerimpl.java
{
502. return getWebsiteByHandle(handle, Boolean.TRUE);
}
```

```
44. public String intercept(ActionInvocation invocation) throws Exception //uiactioninterceptor.java
{
    ...
72. weblog = mgr.getWebsiteByHandle(weblogHandle);
    ...
74. theAction.setActionWeblog(weblog);
}
```

```
198. public void setActionWeblog(Weblog workingWeblog) //uiaction.java
{
    ...
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 28:

Query Name - **Stored_XSS**

Severity -  **Critical**

```
213. public PlanetGroup getGroup(Planet planet, String handle) throws PlanetException //jpaplanetmanagerimpl.java
{
    ...
218. return (PlanetGroup)q.getSingleResult();
}
```

```
71. public void myPrepare() //planetsubscriptions.java
{
    ...
81. setGroup(pmgr.getGroup(getPlanet(), getGroupHandle()));
}
```

```
219. public void setGroup(PlanetGroup group) //planetsubscriptions.java
{
    ...
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 29:

Query Name - **Stored_XSS**

Severity -  Critical

```
180. public List getCustomPingTargets(Weblog website) //jppapingtargetmanagerimpl.java
    ...
    185. return q.getResultList();
```



```
61. public void loadPingTargets() //custompingtargets.java
    ...
    65. setPingTargets(pingTargetMgr.getCustomPingTargets(getActionWeblog()));
```



```
135. public void setPingTargets(List pingTargets) //pingtargetsbase.java
    ,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 30:

Query Name - **Stored_XSS**

Severity -  Critical

```
497. public List getWeblogCategories(Weblog website) //jpaweblogmanagerimpl.java
    ...
    505. return q.getResultList();
```



```
481. public List getWeblogCategories(Weblog website, boolean includeRoot) //jpaweblogmanagerimpl.java
    ...
    486. if (includeRoot) return getWeblogCategories(website);
```



```
92. public String execute() //categories.java
    ...
    100. List<WeblogCategory> cats = wmgr.getWeblogCategories(getActionWeblog(), true);
    ...
    103. allCategories.add(cat);
    ...
    125. setAllCategories(allCategories);
```



```
205. public void setAllCategories(Set allCategories) //categories.java
    ,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 31:

Query Name - **Stored_XSS**

Severity -  Critical

```
497. public List getWeblogCategories(Weblog website) //jpaweblogmanagerimpl.java
    ...
    505. return q.getResultList();
```



```
481. public List getWeblogCategories(Weblog website, boolean includeRoot) //jpaweblogmanagerimpl.java
    ...
    486. if (includeRoot) return getWeblogCategories(website);
```



```
85. public String execute() //categoryremove.java
    ...
    93. List<WeblogCategory> cats = wmgr.getWeblogCategories(getActionWeblog(), true);
    ...
    96. allCategories.add(cat);
    ...
    106. setAllCategories(allCategories);
```



```
177. public void setAllCategories(Set allCategories) //categoryremove.java
    ,
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 32:

Query Name - **Stored_XSS**

Severity -  Critical

```
497. public List getWeblogCategories(Weblog website) //jpaweblogmanagerimpl.java
```

...
505. return q.getResultList();

▼

481. public List getWeblogCategories(Weblog website, boolean includeRoot) //jpaweblogmanagerimpl.java

...
486. if (includeRoot) return getWeblogCategories(website);

▼

77. public void myPrepare() //weblogconfig.java

...
83. setWeblogCategories(wmgr.getWeblogCategories(getActionWeblog()), false));

▼

218. public void setWeblogCategories(List weblogCategories) //weblogconfig.java


,

[\[Table of Contents\]](#) [\[Description\]](#)

Error Message Information Leaks	
CWE ID	209
Description	Server messages need to be parsed before being passed on to the user.
Likelihood of Exploit	High
Common Consequences	Confidentiality: Often this will either reveal sensitive information which may be used for a later attack or private information stored in the server.
Potential Mitigations	Design: When an application detects illegal input, error messages should only provide generic feedback, such as "Illegal characters were detected." Messages should provide few, if any, implementation details. Implementation: Any error should be parsed for dangerous revelations. Build: Debugging information should not make its way into a production release. Handle exceptions internally and do not display errors containing potentially sensitive information to a user. Create default error pages if necessary.
Demonstrative Examples	<div>Java Example: <pre>try { /.../ } catch (Exception e) { System.out.println(e); }</pre></div> <p>Here you are passing much more data than is needed. Another example is passing the SQL exceptions to a WebUser without filtering.</p>
Context Notes	Error messages should not provide attackers with any implementation details when the application detects an illegal action. This includes indicating exactly what is allowable, or exactly what was illegal about the user input. Such detailed information can help an attacker craft another attack that now will pass through the validation filters. The first thing an attacker may use -- once an attack has failed -- to stage the next attack is the error information provided by the server. SQL Injection attacks generally probe the server for information in order to stage a successful attack.
Node Relationships	ChildOf - Information Leak (Information Disclosure) (200) ParentOf - Product-Generated Error Message Information Leak (210) ParentOf - Product-External Error Message Information Leak (211)
Source Taxonomies	CLASP - Accidental leaking of sensitive information through error messages
Applicable Platforms	All

Path 1:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**


34. public class PlanetConfig //planetconfig.java

...
125. e.printStackTrace();

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

119. public void runScript(//sqlscriptrunner.java

...
137. + "]" : " + ex.getLocalizedMessage();
...
140. ex.printStackTrace(new PrintWriter(sw));

Path 3:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
35. public class WebloggerConfig //webloggerconfig.java
    ...
    124. e.printStackTrace();
```

Path 4:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
89. public void doPost(HttpServletRequest request, HttpServletResponse response) //trackbackervlet.java
    ...
    141. error = e.getMessage();
    ...
    146. pw.println(this.getErrorResponse(error));
```



```
244. private String getErrorResponse(String message) //trackbackervlet.java
    ...
    252. output.append(message);
    253. output.append("</message>");
    ...
    256. return output.toString();
```



```
89. public void doPost(HttpServletRequest request, HttpServletResponse response) //trackbackervlet.java
    ...
    146. pw.println(this.getErrorResponse(error));
```

Path 5:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
89. public void doPost(HttpServletRequest request, HttpServletResponse response) //trackbackervlet.java
    ...
    141. error = e.getMessage();
    ...
    222. pw.println(this.getErrorResponse(error));
```



```
244. private String getErrorResponse(String message) //trackbackervlet.java
    ...
    252. output.append(message);
    253. output.append("</message>");
    ...
    256. return output.toString();
```



```
89. public void doPost(HttpServletRequest request, HttpServletResponse response) //trackbackervlet.java
    ...
    222. pw.println(this.getErrorResponse(error));
```

Path 6:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
47. protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java
    ...
    61. res.sendError(he.getStatus(), he.getMessage());
```

Path 7:

Query Name - **Verbose_Error_Reporting**

Severity -  Serious

```
47. protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java

...
61. res.sendError(he.getStatus(), he.getMessage());
62. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Verbose_Error_Reporting**

Severity -  Serious

```
71. protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java

...
85. res.sendError(he.getStatus(), he.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Verbose_Error_Reporting**

Severity -  Serious

```
71. protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java

...
85. res.sendError(he.getStatus(), he.getMessage());
86. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Verbose_Error_Reporting**

Severity -  Serious

```
95. protected void doPut(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java

...
109. res.sendError(he.getStatus(), he.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 11:

Query Name - **Verbose_Error_Reporting**

Severity -  Serious

```
95. protected void doPut(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java

...
109. res.sendError(he.getStatus(), he.getMessage());
110. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 12:

Query Name - **Verbose_Error_Reporting**

Severity -  Serious

```
118. protected void doDelete(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java

...
132. res.sendError(he.getStatus(), he.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 13:

Query Name - **Verbose_Error_Reporting**

Severity -  Serious

```
118. protected void doDelete(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java
```

```
...
132. res.sendError(he.getStatus(), he.getMessage());
133. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 14:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
...
140. res.sendError(ae.getStatus(), ae.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 15:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
...
142. res.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, e.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 16:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
157. protected void doPost(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
...
228. res.sendError(ae.getStatus(), ae.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 17:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
157. protected void doPost(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
...
230. res.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, e.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 18:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
245. protected void doPut(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
...
287. res.sendError(ae.getStatus(), ae.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 19:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
245. protected void doPut(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
...
289. res.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, e.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 20:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
303. protected void doDelete(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
    ...
    319. res.sendError(ae.getStatus(), ae.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 21:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
303. protected void doDelete(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
    ...
    321. res.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, e.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 22:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
210. public void testCommentParentDeletes() throws Exception //commenttest.java
    ...
    263. e.printStackTrace(pw);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 23:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
168. public void testUniquenessOfFolderNames() throws Exception //folderfunctionalitytest.java
    ...
    192. t.printStackTrace(pw);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 24:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
342. public void testCreateAnEntryWithTagsShortcut() throws Exception //weblogentrytest.java
    ...
    387. t.printStackTrace(pw);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 25:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
447. public void testRemoveTagsViaShortcut() throws Exception //weblogentrytest.java
    ...
    483. t.printStackTrace(pw);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 26:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
540. public void testGetEntriesByTag() throws Exception //weblogentrytest.java
    ...
    569. t.printStackTrace(pw);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 27:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**


```
714. public void testTagAggregates() throws Exception //weblogentrytest.java
    |
    | ...
    | 862. t.printStackTrace(pw);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 28:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
62. protected void setUp() throws Exception //handlerbasetest.java
    |
    | ...
    | 67. t.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 29:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
62. protected void setUp() throws Exception //handlerbasetest.java
    |
    | ...
    | 72. t.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 30:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
62. protected void setUp() throws Exception //handlerbasetest.java
    |
    | ...
    | 77. t.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 31:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
81. protected void tearDown() throws Exception //handlerbasetest.java
    |
    | ...
    | 86. t.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 32:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
81. protected void tearDown() throws Exception //handlerbasetest.java
    |
    | ...
    | 91. t.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 33:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
81. protected void tearDown() throws Exception //handlerbasetest.java
    |
    | ...
    | 96. t.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 34:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
26. public void testHandler() //memberhandlertest.java
    |
    |...
    |65. ioe.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 35:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
26. public void testHandler() //memberhandlertest.java
    |
    |...
    |68. je.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 36:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
26. public void testHandler() //memberhandlertest.java
    |
    |...
    |71. uree.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 37:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
31. public void testHandler() //userhandlertest.java
    |
    |...
    |62. ioe.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 38:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
31. public void testHandler() //userhandlertest.java
    |
    |...
    |65. je.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 39:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
31. public void testHandler() //userhandlertest.java
    |
    |...
    |68. uree.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 40:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
73. public void testEnabled() //userhandlertest.java
    |
    |...
    |88. fail(ioe.getMessage());
    |89. ioe.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 41:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
73. public void testEnabled() //userhandlertest.java
    ...
    91. fail(je.getMessage());
    92. je.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 42:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
73. public void testEnabled() //userhandlertest.java
    ...
    89. ioe.printStackTrace();
    ...
    95. uree.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 43:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
27. public void testHandler() //webloghandlertest.java
    ...
    60. fail(ioe.getMessage());
    61. ioe.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 44:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
27. public void testHandler() //webloghandlertest.java
    ...
    63. fail(je.getMessage());
    64. je.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 45:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
27. public void testHandler() //webloghandlertest.java
    ...
    66. fail(uree.getMessage());
    67. uree.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 46:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
27. public void testHandler() //webloghandlertest.java
    ...
    73. e.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 47:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
78. public void testEnabled() //webloghandlertest.java
    ...
    93. fail(ioe.getMessage());
    94. ioe.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 48:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
78. public void testEnabled() //webloghandlertest.java
    ...
    96. fail(je.getMessage());
    97. je.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 49:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
78. public void testEnabled() //webloghandlertest.java
    ...
    99. fail(uree.getMessage());
    100. uree.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 50:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
119. public void runScript( //sqlscriptrunner.java
    ...
    140. ex.printStackTrace(new PrintWriter(sw));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 51:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
89. public void doPost(HttpServletRequest request, HttpServletResponse response) //trackbackservlet.java
    ...
    215. error = e.getMessage();
    ...
    217. error = e.getClass().getName();
    ...
    222. pw.println(this.getErrorResponse(error));
```



```
244. private String getErrorResponse(String message) //trackbackservlet.java
    ...
    252. output.append(message);
    253. output.append("</message>");
    ...
    256. return output.toString();
```



```
89. public void doPost(HttpServletRequest request, HttpServletResponse response) //trackbackservlet.java
    ...
    222. pw.println(this.getErrorResponse(error));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 52:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
89. public void doPost(HttpServletRequest request, HttpServletResponse response) //trackbackservlet.java
    ...
    217. error = e.getClass().getName();
    ...
    222. pw.println(this.getErrorResponse(error));
```



```
244. private String getErrorResponse(String message) //trackbackservlet.java
    ...
    252. output.append(message);
    253. output.append("</message>");
```

```
...
256. return output.toString();
```

```
89. public void doPost(HttpServletRequest request, HttpServletResponse response) //trackbackervlet.java
...
222. pw.println(this.getErrorResponse(error));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 53:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
47. protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminservlet.java
...
61. res.sendError(he.getStatus(), he.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 54:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
47. protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminservlet.java
...
61. res.sendError(he.getStatus(), he.getMessage());
62. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 55:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
47. protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminservlet.java
...
62. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 56:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
71. protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminservlet.java
...
85. res.sendError(he.getStatus(), he.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 57:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
71. protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminservlet.java
...
85. res.sendError(he.getStatus(), he.getMessage());
86. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 58:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
71. protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminservlet.java
```

```
...
86. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 59:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
95. protected void doPut(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java
```

```
...
109. res.sendError(he.getStatus(), he.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 60:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
95. protected void doPut(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java
```

```
...
109. res.sendError(he.getStatus(), he.getMessage());
110. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 61:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
95. protected void doPut(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java
```

```
...
110. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 62:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
118. protected void doDelete(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java
```

```
...
132. res.sendError(he.getStatus(), he.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 63:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
118. protected void doDelete(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java
```

```
...
132. res.sendError(he.getStatus(), he.getMessage());
133. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 64:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
118. protected void doDelete(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException //adminervlet.java
```

```
...
133. he.printStackTrace(res.getWriter());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 65:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
    ...
    140. res.sendError(ae.getStatus(), ae.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 66:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
157. protected void doPost(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
    ...
    268. writer.close();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 67:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
245. protected void doPut(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
    ...
    287. res.sendError(ae.getStatus(), ae.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 68:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
303. protected void doDelete(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
    ...
    319. res.sendError(ae.getStatus(), ae.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 69:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
73. public void testEnabled() //userhandlertest.java
    ...
    89. ioe.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 70:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**

```
73. public void testEnabled() //userhandlertest.java
    ...
    92. je.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)


Path 71:

Query Name - **Verbose_Error_Reporting**

Severity -  **Serious**


```
73. public void testEnabled() //userhandlertest.java
    ...
    95. uree.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 72:
Query Name - **Verbose_Error_Reporting**
Severity -  **Serious**


```
27. public void testHandler() //webloghandlertest.java
    |
    |...
    |61. ioe.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 73:
Query Name - **Verbose_Error_Reporting**
Severity -  **Serious**


```
27. public void testHandler() //webloghandlertest.java
    |
    |...
    |64. je.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 74:
Query Name - **Verbose_Error_Reporting**
Severity -  **Serious**

```
27. public void testHandler() //webloghandlertest.java
    |
    |...
    |67. uree.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 75:
Query Name - **Verbose_Error_Reporting**
Severity -  **Serious**


```
78. public void testEnabled() //webloghandlertest.java
    |
    |...
    |94. ioe.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 76:
Query Name - **Verbose_Error_Reporting**
Severity -  **Serious**

```
78. public void testEnabled() //webloghandlertest.java
    |
    |...
    |97. je.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 77:
Query Name - **Verbose_Error_Reporting**
Severity -  **Serious**

```
78. public void testEnabled() //webloghandlertest.java
    |
    |...
    |100. uree.printStackTrace();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Resource Injection	
CWE ID	99
Description	Allowing user input to control resource identifiers may enable an attacker to access or modify otherwise protected system resources.
Likelihood of Exploit	High
Weakness Ordinality	Primary (Weakness exists independent of other weaknesses)
Causal Nature	Explicit (This is an explicit weakness resulting from behavior of the developer)

Potential Mitigations	Assume all input is malicious. Use an appropriate combination of black lists and white lists to ensure only valid and expected input is processed by the system.
Demonstrative Examples	<p>The following Java code uses input from an HTTP request to create a file name. The programmer has not considered the possibility that an attacker could provide a file name such as "../../../tomcat/conf/server.xml", which causes the application to delete one of its own configuration files.</p> <p>Java Example:</p> <pre>String rName = request.getParameter("reportName"); File rFile = new File("/usr/local/apfr/reports/" + rName); ... rFile.delete();</pre> <p>The following code uses input from the command line to determine which file to open and echo back to the user. If the program runs with privileges and malicious users can create soft links to the file, they can use the program to read the first part of any file on the system.</p> <p>C++ Example:</p> <pre>ifstream ifs(argv[0]); string s; ifs >> s; cout << s;</pre> <p>The kind of resource the data affects indicates the kind of content that may be dangerous. For example, data containing special characters like period, slash, and backslash, are risky when used in methods that interact with the file system. (Resource Injection, when it is related to file system resources, sometimes goes by the name "path manipulation.") Similarly, data that contains URLs and URIs is risky for functions that create remote connections.</p>
Context Notes	A resource injection issue occurs when the following two conditions are met: 1. An attacker can specify the identifier used to access a system resource. For example, an attacker might be able to specify part of the name of a file to be opened or a port number to be used. 2. By specifying the resource, the attacker gains a capability that would not otherwise be permitted. For example, the program may give the attacker the ability to overwrite the specified file, run with a configuration controlled by the attacker, or transmit sensitive information to a third-party server. Note: Resource injection that involves resources stored on the filesystem goes by the name path manipulation and is reported in separate category. See the path manipulation description for further details of this vulnerability.
Node Relationships	ChildOf - Injection (74) CanAlsoBe - Path Manipulation (73) ChildOf - Weaknesses Examined by SAMATE (630)
Source Taxonomies	7 Pernicious Kingdoms - Resource Injection
Applicable Platforms	All

Path 1:
Query Name - **Files_Manipulation**
Severity -  **Serious**

```
157. protected void doPost(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
    ...
196. String slug = req.getHeader("Slug");
    ...
200. pathInfo, title, slug, req.getContentType(), req.getInputStream());

709. public Entry postMedia(String[] pathInfo, //rolleratomhandler.java
    710. String title, String slug, String contentType, InputStream is)
    ...
726. (slug != null) ? slug : Utilities.replaceNonAlphanumeric(title, ' '), contentType);

792. private String createFileName(Weblog weblog, String slug, String contentType) //rolleratomhandler.java
    ...
808. StringTokenizer token = new StringTokenizer(slug);
    ...
812. String s = token.nextToken();
813. s = s.toLowerCase();
814. tmp = (tmp == null) ? s : tmp + "_" + s;
    ...
817. if (!tmp.endsWith(".") + ext)) {
    ...
820. fileName = tmp;
    ...
830. return fileName;

709. public Entry postMedia(String[] pathInfo, //rolleratomhandler.java
    ...
725. String fileName = createFileName(website,
    ...
728. tempFile = File.createTempFile(fileName, "tmp");
729. FileOutputStream fos = new FileOutputStream(tempFile);
```

Path 2:

Query Name - **Files_Manipulation**

Severity -  **Serious**

```
157. protected void doPost(HttpServletRequest req, HttpServletResponse res) //atomservlet.java
    ...
196. String slug = req.getHeader("Slug");
    ...
200. pathInfo, title, slug, req.getContentType(), req.getInputStream());
```



```
709. public Entry postMedia(String[] pathInfo, //rolleratomhandler.java
    ...
710. String title, String slug, String contentType, InputStream is)
    ...
726. (slug != null) ? slug : Utilities.replaceNonAlphanumeric(title, ' '), contentType);
```



```
792. private String createFileName(Weblog weblog, String slug, String contentType) //rolleratomhandler.java
    ...
808. StringTokenizer token = new StringTokenizer(slug);
    ...
812. String s = token.nextToken();
813. s = s.toLowerCase();
814. tmp = (tmp == null) ? s : tmp + "_" + s;
    ...
817. if (!tmp.endsWith(".") + ext)) {
    ...
820. fileName = tmp;
    ...
830. return fileName;
```



```
709. public Entry postMedia(String[] pathInfo, //rolleratomhandler.java
    ...
725. String fileName = createFileName(website,
    ...
728. tempFile = File.createTempFile(fileName, "tmp");
    ...
737. FileInputStream fis = new FileInputStream(tempFile);
```

Path 3:

Query Name - **Files_Manipulation**

Severity -  **Serious**

```
157. protected void doPost(HttpServletRequest req, HttpServletResponse res) //atomservlet.java
    ...
200. pathInfo, title, slug, req.getContentType(), req.getInputStream());
```



```
709. public Entry postMedia(String[] pathInfo, //rolleratomhandler.java
    ...
710. String title, String slug, String contentType, InputStream is)
    ...
726. (slug != null) ? slug : Utilities.replaceNonAlphanumeric(title, ' '), contentType);
```



```
792. private String createFileName(Weblog weblog, String slug, String contentType) //rolleratomhandler.java
    ...
803. String[] typeTokens = contentType.split("/");
804. String ext = typeTokens[1];
    ...
818. fileName = tmp + "." + ext;
    ...
830. return fileName;
```



```
709. public Entry postMedia(String[] pathInfo, //rolleratomhandler.java
    ...
725. String fileName = createFileName(website,
    ...
728. tempFile = File.createTempFile(fileName, "tmp");
729. FileOutputStream fos = new FileOutputStream(tempFile);
```

Path 4:

Query Name - **Files_Manipulation**

Severity  Serious

```
157. protected void doPost(HttpServletRequest req, HttpServletResponse res) //atomservlet.java
```

```
...
200. pathInfo, title, slug, req.getContentType(), req.getInputStream());
```

```
709. public Entry postMedia(String[] pathInfo, //rolleratomhandler.java
```

```
710. String title, String slug, String contentType, InputStream is)
```

```
...
726. (slug != null) ? slug : Utilities.replaceNonAlphanumeric(title, ' '), contentType);
```

```
792. private String createFileName(Weblog weblog, String slug, String contentType) //rolleratomhandler.java
```

```
...
803. String[] typeTokens = contentType.split("/");
```

```
804. String ext = typeTokens[1];
```

```
...
818. fileName = tmp + "." + ext;
```

```
...
830. return fileName;
```

```
709. public Entry postMedia(String[] pathInfo, //rolleratomhandler.java
```

```
...
725. String fileName = createFileName(website,
```

```
...
728. tempFile = File.createTempFile(fileName, "tmp");
```

```
...
737. FileInputStream fis = new FileInputStream(tempFile);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Cross-Site Request Forgery (CSRF)

CWE ID	352
Description	The web product does not, or can not, sufficiently verify whether a well-formed, valid, consistent request was intentionally provided by the user who submitted the request. Note: CSRF is multi-channel: 1. Attacker-to-victim (injection; external or internal channel) 2. Victim-to-server (activation; internal channel)
Alternate Terms	Session Riding Cross Site Reference Forgery XSRF
Observed Examples	CVE-2004-1703 - CVE-2004-1995 - CVE-2004-1967 - CVE-2004-1842 - CVE-2005-1947 - CVE-2005-2059 - CVE-2005-1674 - CSRF
Context Notes	Could be resultant from XSS, although XSS is not necessarily required.
References	Peter W. "Cross-Site Request Forgeries (Re: The Dangers of Allowing Users to Post Images)". Bugtraq. < http://marc.info/?l=bugtraq&m=99263135911884&w=2 >. Robert Auger. "CSRF - The Cross-Site Request Forgery (CSRF/XSRF) FAQ". < http://www.cgisecurity.com/articles/csrf-faq.shtml >.
Node Relationships	ChildOf - Insufficient Verification of Data (345) ChildOf - Weaknesses in OWASP Top Ten (629) ChildOf - Weaknesses Used by NVD (635)
Source Taxonomies	PLOVER - Cross-Site Request Forgery (CSRF)
Applicable Platforms	All

Path 1:

Query Name - **XSRF**

Severity  Serious

```
114. public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,  
IOException //pageservlet.java
```

```
...
383. initData.put("requestParameters", request.getParameterMap());
```

```
...
403. ModelLoader.loadModels(siteModels, model, initData, true);
```

```
86. public static void loadModels(String modelsString, Map model, //modelloader.java
```

```
...
95. pageModel.init(initData);
```

```
78. public void init(Map initData) throws WebloggerException //sitemodel.java
```

```
...
81. this.weblogRequest = (WeblogRequest) initData.get("parsedRequest");
...
92. this.feedRequest = (WeblogFeedRequest) weblogRequest;
...
94. pageNum = feedRequest.getPage();
...
104. weblog = weblogRequest.getWeblog();
```

```
185. public Weblog getWeblog() //weblogrequest.java
```

```
...
190. weblog = umgr.getWebsiteByHandle(weblogHandle, Boolean.TRUE);
```

```
508. public Weblog getWebsiteByHandle(String handle, Boolean enabled) //jpausermanagerimpl.java
```

```
...
533. query.setParameter(1, handle);
...
536. website = (Weblog)query.getSingleResult();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **XSRF**

Severity -  **Serious**

```
335. protected String[] getPathInfo(HttpServletRequest request) //atomervlet.java
```

```
336. String mPathInfo = request.getPathInfo();
337. mPathInfo = (mPathInfo!=null) ? mPathInfo : "";
338. return StringUtils.split(mPathInfo, "/");
```

```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
```

```
...
90. String[] pathInfo = getPathInfo(req);
...
92. if (handler.isIntrospectionURI(pathInfo)) {
```

```
894. public boolean isIntrospectionURI(String[] pathInfo) //rolleratomhandler.java
```

```
,
```

```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
```

```
...
92. if (handler.isIntrospectionURI(pathInfo)) {
...
104. else if (handler.isCollectionURI(pathInfo)) {
```

```
919. public boolean isCollectionURI(String[] pathInfo) //rolleratomhandler.java
```

```
,
```

```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
```

```
...
104. else if (handler.isCollectionURI(pathInfo)) {
...
106. Feed col = handler.getCollection(pathInfo);
```

```
288. public Feed getCollection(String[] pathInfo) throws AtomException //rolleratomhandler.java
```

```
289. if (pathInfo.length > 0 && pathInfo[1].equals("entries")) {
290. return getCollectionOfEntries(pathInfo);
```

```
300. public Feed getCollectionOfEntries(String[] pathInfo) throws AtomException //rolleratomhandler.java
```

```
...
307. String s = pathInfo[2].trim();
308. start = Integer.parseInt(s);
...
334. start, // offset (for range paging)
```

```
511. public List getWeblogEntries( //jpaweblogmanagerimpl.java
```

```
...
```

```
523. int offset,
...
639. query.setFirstResult(offset);
...
645. return query.getResultList();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **XSRF**

Severity -  **Serious**

```
335. protected String[] getPathInfo(HttpServletRequest request) //atomervlet.java
```

```
336. String mPathInfo = request.getPathInfo();
337. mPathInfo = (mPathInfo!=null) ? mPathInfo : "";
338. return StringUtils.split(mPathInfo, "/");
```



```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
```

```
...
90. String[] pathInfo = getPathInfo(req);
...
92. if (handler.isIntrospectionURI(pathInfo)) {
```



```
894. public boolean isIntrospectionURI(String[] pathInfo) //rolleratomhandler.java
```

```
,
```



```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
```

```
...
92. if (handler.isIntrospectionURI(pathInfo)) {
...
104. else if (handler.isCollectionURI(pathInfo)) {
```



```
919. public boolean isCollectionURI(String[] pathInfo) //rolleratomhandler.java
```

```
,
```



```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
```

```
...
104. else if (handler.isCollectionURI(pathInfo)) {
...
106. Feed col = handler.getCollection(pathInfo);
```



```
288. public Feed getCollection(String[] pathInfo) throws AtomException //rolleratomhandler.java
```

```
289. if (pathInfo.length > 0 && pathInfo[1].equals("entries")) {
290. return getCollectionOfEntries(pathInfo);
```



```
300. public Feed getCollectionOfEntries(String[] pathInfo) throws AtomException //rolleratomhandler.java
```

```
...
313. String handle = pathInfo[0];
...
315. Weblog website = roller.getUserManager().getWebsiteByHandle(handle);
```



```
501. public Weblog getWebsiteByHandle(String handle) throws WebloggerException //jpausermanagerimpl.java
```

```
502. return getWebsiteByHandle(handle, Boolean.TRUE);
```



```
508. public Weblog getWebsiteByHandle(String handle, Boolean enabled) //jpausermanagerimpl.java
```

```
...
519. (String) this.weblogHandleToIdMap.get(handle));
```



```
497. public Weblog getWebsite(String id) throws WebloggerException //jpausermanagerimpl.java
```

```
498. return (Weblog) this.strategy.load(Weblog.class, id);
```



```
213. public Object load(Class clazz, String id) //jpapersistencestrategy.java
```

```
...
216. return em.find(clazz, id);
```



```
497. public Weblog getWebsite(String id) throws WebloggerException //jpausermanagerimpl.java
```

```
498. return (Weblog) this.strategy.load(Weblog.class, id);
```



```
508. public Weblog getWebsiteByHandle(String handle, Boolean enabled) //jpausermanagerimpl.java
```

```
,
```

```
...
518. Weblog weblog = this.getWebsite(
...
522. if(enabled == null || enabled.equals(weblog.getEnabled())) {
```

```
501. public Weblog getWebsiteByHandle(String handle) throws WebloggerException //jpausermanagerimpl.java
| 502. return getWebsiteByHandle(handle, Boolean.TRUE);
```

```
300. public Feed getCollectionOfEntries(String[] pathInfo) throws AtomException //rolleratomhandler.java
| ...
315. Weblog website = roller.getUserManager().getWebsiteByHandle(handle);
| ...
323. website, // website
```

```
511. public List getWeblogEntries( //jpaweblogmanagerimpl.java
| 512. Weblog website,
| ...
528. cat = getWeblogCategoryByPath(website, catName);
```

```
971. public WeblogCategory getWeblogCategoryByPath(Weblog website, //jpaweblogmanagerimpl.java
| ...
973. return getWeblogCategoryByPath(website, null, categoryPath);
```

```
980. public WeblogCategory getWeblogCategoryByPath(Weblog website, //jpaweblogmanagerimpl.java
| ...
997. q.setParameter(2, website);
| ...
999. return (WeblogCategory)q.getSingleResult();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **XSRF**

Severity -  **Serious**

```
57. public WeblogCommentRequest(HttpServletRequest request) //weblogcommentrequest.java
| ...
125. this.name = Utilities.removeHTML(request.getParameter("name"));
```

```
139. public static String removeHTML(String str) //utilities.java
| 140. return removeHTML(str, true);
```

```
152. public static String removeHTML(String str, boolean addSpace) //utilities.java
| ...
159. return str;
```

```
139. public static String removeHTML(String str) //utilities.java
| 140. return removeHTML(str, true);
```

```
57. public WeblogCommentRequest(HttpServletRequest request) //weblogcommentrequest.java
| ...
125. this.name = Utilities.removeHTML(request.getParameter("name"));
```

```
156. public void doPost(HttpServletRequest request, HttpServletResponse response) //commentservlet.java
| ...
192. commentRequest = new WeblogCommentRequest(request);
| ...
204. entry = commentRequest.getWeblogEntry();
```

```
202. public WeblogEntry getWeblogEntry() //weblogcommentrequest.java
| ...
207. weblogEntry = wmgr.getWeblogEntryByAnchor(getWeblog(), weblogAnchor);
```

```
185. public Weblog getWeblog() //weblogrequest.java
| ...
190. weblog = umgr.getWebsiteByHandle(weblogHandle, Boolean.TRUE);
```

```
508. public Weblog getWebsiteByHandle(String handle, Boolean enabled) //jpausermanagerimpl.java
```

```
...
533. query.setParameter(1, handle);
...
536. website = (Weblog)query.getSingleResult();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **XSRF**

Severity -  **Serious**

```
48. public WeblogPreviewResourceRequest(HttpServletRequest request) //weblogpreviewresourcerequest.java
```

```
...
57. this.themeName = request.getParameter("theme");
```

```
73. public void doGet(HttpServletRequest request, HttpServletResponse response) //previewresourceservlet.java
```

```
...
84. resourceRequest = new WeblogPreviewResourceRequest(request);
...
86. weblog = resourceRequest.getWeblog();
```

```
185. public Weblog getWeblog() //weblogrequest.java
```

```
...
190. weblog = umgr.getWebsiteByHandle(weblogHandle, Boolean.TRUE);
```

```
508. public Weblog getWebsiteByHandle(String handle, Boolean enabled) //jpausermanagerimpl.java
```

```
...
533. query.setParameter(1, handle);
...
536. website = (Weblog)query.getSingleResult();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **XSRF**

Severity -  **Serious**

```
335. protected String[] getPathInfo(HttpServletRequest request) //atomervlet.java
```

```
336. String mPathInfo = request.getPathInfo();
337. mPathInfo = (mPathInfo!=null) ? mPathInfo : "";
338. return StringUtils.split(mPathInfo, "/");
```

```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
```

```
...
90. String[] pathInfo = getPathInfo(req);
...
92. if (handler.isIntrospectionURI(pathInfo)) {
```

```
894. public boolean isIntrospectionURI(String[] pathInfo) //rolleratomhandler.java
```

```
;
```

```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
```

```
...
92. if (handler.isIntrospectionURI(pathInfo)) {
...
104. else if (handler.isCollectionURI(pathInfo)) {
```

```
919. public boolean isCollectionURI(String[] pathInfo) //rolleratomhandler.java
```

```
;
```

```
84. protected void doGet(HttpServletRequest req, HttpServletResponse res) //atomervlet.java
```

```
...
104. else if (handler.isCollectionURI(pathInfo)) {
...
106. Feed col = handler.getCollection(pathInfo);
```

```
288. public Feed getCollection(String[] pathInfo) throws AtomException //rolleratomhandler.java
```

```
289. if (pathInfo.length > 0 && pathInfo[1].equals("entries")) {
...
292. return getCollectionOfResources(pathInfo);
```

```

394. public Feed getCollectionOfResources(String[] rawPathInfo) throws AtomException //rolleratomhandler.java
    ...
399. String[] pathInfo = rawPathInfo;
    ...
409. String path = filePathFromPathInfo(pathInfo);
    ...
412. String handle = pathInfo[0];
    ...
414. Weblog website = roller.getUserManager().getWebsiteByHandle(handle);

```

```

501. public Weblog getWebsiteByHandle(String handle) throws WebloggerException //jpausermanagerimpl.java
    ...
502. return getWebsiteByHandle(handle, Boolean.TRUE);

```

```

508. public Weblog getWebsiteByHandle(String handle, Boolean enabled) //jpausermanagerimpl.java
    ...
533. query.setParameter(1, handle);
    ...
536. website = (Weblog)query.getSingleResult();

```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **XSRF**

Severity -  **Serious**

```

55. public void doGet(HttpServletRequest request, HttpServletResponse response) //tagstatsservlet.java
    ...
71. String pathInfo = request.getPathInfo();
    ...
78. if(pathInfo.startsWith("/"))
79. pathInfo = pathInfo.substring(1);
    ...
86. handle = pathInfo.substring(0,slash);
    ...
113. website = umgr.getWebsiteByHandle(handle, Boolean.TRUE);

```

```

508. public Weblog getWebsiteByHandle(String handle, Boolean enabled) //jpausermanagerimpl.java
    ...
533. query.setParameter(1, handle);
    ...
536. website = (Weblog)query.getSingleResult();

```

[\[Table of Contents\]](#) [\[Description\]](#)

HTTP Response Splitting

CWE ID	113
Description	Writing unvalidated data into an HTTP header allows an attacker to specify the entirety of the HTTP response rendered by the browser. HTTP response splitting occurs when an HTTP request contains unexpected CR and LF characters. The server may respond with an output stream that is interpreted as two different HTTP responses (instead of one). An attacker can control the second response and mount attacks such as cross-site scripting and cache poisoning attacks.
Potential Mitigations	Construct HTTP headers very carefully, avoiding the use of non-validated input data. Assume all input is malicious. Use an appropriate combination of black lists and white lists to ensure only valid and expected input is processed by the system. Input (specifically, unexpected CRLFs) that is not appropriate should not be processed into HTTP headers.
Demonstrative Examples	The following code segment reads the name of the author of a weblog entry, author, from an HTTP request and sets it in a cookie header of an HTTP response.

```

String author = request.getParameter(AUTHOR_PARAM);
...
Cookie cookie = new Cookie("author", author);
cookie.setMaxAge(cookieExpiration);
response.addCookie(cookie);

```

Assuming a string consisting of standard alpha-numeric characters, such as "Jane Smith", is submitted in the request the HTTP response including this cookie might take the following form: HTTP/1.1 200 OK ... Set-Cookie: author=Jane Smith ... However, because the value of the cookie is formed of unvalidated user input the response will only maintain this form if the value submitted for AUTHOR_PARAM does not contain any CR and LF characters. If an attacker submits a malicious string, such as "Wiley Hacker\r\nHTTP/1.1 200 OK\r\n...", then the HTTP response would be split into two responses of the following form: HTTP/1.1 200 OK ... Set-Cookie: author=Wiley Hacker HTTP/1.1 200 OK ... Clearly, the second response is completely controlled by the attacker and can be constructed with any header and body content desired. The ability of attacker to construct arbitrary HTTP responses permits a variety of resulting attacks, including: cross-user defacement, web and browser cache poisoning, cross-site scripting and page hijacking. Others examples: Cross-User Defacement: An attacker can make a single request to a vulnerable server that will cause the sever to create two responses, the second of which may be misinterpreted as a response to a different request,

possibly one made by another user sharing the same TCP connection with the sever. This can be accomplished by convincing the user to submit the malicious request themselves, or remotely in situations where the attacker and the user share a common TCP connection to the server, such as a shared proxy server. In the best case, an attacker can leverage this ability to convince users that the application has been hacked, causing users to lose confidence in the security of the application. In the worst case, an attacker may provide specially crafted content designed to mimic the behavior of the application but redirect private information, such as account numbers and passwords, back to the attacker. Cache Poisoning: The impact of a maliciously constructed response can be magnified if it is cached either by a web cache used by multiple users or even the browser cache of a single user. If a response is cached in a shared web cache, such as those commonly found in proxy servers, then all users of that cache will continue receive the malicious content until the cache entry is purged. Similarly, if the response is cached in the browser of an individual user, then that user will continue to receive the malicious content until the cache entry is purged, although the user of the local browser instance will be affected. Cross-Site Scripting: Once attackers have control of the responses sent by an application, they have a choice of a variety of malicious content to provide users. Cross-site scripting is common form of attack where malicious JavaScript or other code included in a response is executed in the user's browser. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data like cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site. The most common and dangerous attack vector against users of a vulnerable application uses JavaScript to transmit session and authentication information back to the attacker who can then take complete control of the victim's account. Page Hijacking: In addition to using a vulnerable application to send malicious content to a user, the same root vulnerability can also be leveraged to redirect sensitive content generated by the server and intended for the user to the attacker instead. By submitting a request that results in two responses, the intended response from the server and the response generated by the attacker, an attacker can cause an intermediate node, such as a shared proxy server, to misdirect a response generated by the server for the user to the attacker. Because the request made by the attacker generates two responses, the first is interpreted as a response to the attacker's request, while the second remains in limbo. When the user makes a legitimate request through the same TCP connection, the attacker's request is already waiting and is interpreted as a response to the victim's request. The attacker then sends a second request to the server, to which the proxy server responds with the server generated request intended for the victim, thereby compromising any sensitive information in the headers or body of the response intended for the victim.

Observed Examples	CVE-2005-1951 - Application accepts CRLF in an object ID, allowing HTTP response splitting. CVE-2004-2146 - Application accepts CRLF in an object ID, allowing HTTP response splitting. CVE-2004-1620 - HTTP response splitting via CRLF in parameter related to URL. CVE-2004-1656 - HTTP response splitting via CRLF in parameter related to URL. CVE-2004-1687 - HTTP response splitting via CRLF in parameter related to URL. CVE-2005-2060 - Bulletin board allows response splitting via CRLF in parameter. CVE-2005-2065 - Bulletin board allows response splitting via CRLF in parameter. CVE-2004-2512 - Response splitting via CRLF in PHPSESSID.
Context Notes	HTTP response splitting vulnerabilities occur when: 1. Data enters a web application through an untrusted source, most frequently an HTTP request. 2. The data is included in an HTTP response header sent to a web user without being validated for malicious characters. As with many software security vulnerabilities, HTTP response splitting is a means to an end, not an end in itself. At its root, the vulnerability is straightforward: an attacker passes malicious data to a vulnerable application, and the application includes the data in an HTTP response header. To mount a successful exploit, the application must allow input that contains CR (carriage return, also given by %0d or \r) and LF (line feed, also given by %0a or \n)characters into the header. These characters not only give attackers control of the remaining headers and body of the response the application intends to send, but also allows them to create additional responses entirely under their control. Note that HTTP response splitting is probably only multi-factor in an environment that uses intermediaries.
Node Relationships	ChildOf - CRLF Injection (93) ChildOf - Unprotected Alternate Channel (420) Peer - Web Problems (442)
Source Taxonomies	PLOVER - HTTP response splitting 7 Pernicious Kingdoms - HTTP Response Splitting
Applicable Platforms	All

Path 1:

Query Name - HttpSplitting

Severity -  Serious

64. public OldFeedRequest(HttpServletRequest request) throws Exception //oldfeedrequest.java

...
120. this.flavor = request.getParameter("flavor");

▼

247. private String figureFeedRedirect(HttpServletRequest request) //redirectervlet.java

...
252. feedRequest = new OldFeedRequest(request);
...
264. newUrl += "/"+"weblog+"/feed/entries/"+feedRequest.getFlavor();

▼

149. public String getFlavor() //oldfeedrequest.java

150. return flavor;

▼

247. private String figureFeedRedirect(HttpServletRequest request) //redirectervlet.java

...
264. newUrl += "/"+"weblog+"/feed/entries/"+feedRequest.getFlavor();

```
...
274. return newUrl + URLUtilities.getQueryString(params);


76. public void doGet(HttpServletRequest request, HttpServletResponse response) //redirectservlet.java
    ...
    118. redirectUrl = figureFeedRedirect(request);
    ...
    154. response.setHeader("Location", redirectUrl);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Log Forging	
CWE ID	117
Description	Writing unvalidated user input into log files can allow an attacker to forge log entries or inject malicious content into logs.
Likelihood of Exploit	Medium
Weakness Ordinality	Primary (Weakness exists independent of other weaknesses)
Causal Nature	Explicit (This is an explicit weakness resulting from behavior of the developer)
Potential Mitigations	Assume all input is malicious. Use an appropriate combination of black lists and white lists to ensure only valid, expected and appropriate input is processed by the system.
Demonstrative Examples	<p>The following web application code attempts to read an integer value from a request object. If the value fails to parse as an integer, then the input is logged with an error message indicating what happened.</p> <pre>... String val = request.getParameter("val"); try { int value = Integer.parseInt(val); } catch (NumberFormatException) { log.info("Failed to parse val = " + val); } ...</pre>
	<p>If a user submits the string "twenty-one" for val, the following entry is logged: INFO: Failed to parse val=twenty-one However, if an attacker submits the string "twenty-one%0a%0aINFO: +User+logged+out%3dbadguy", the following entry is logged: INFO: Failed to parse val=twenty-one INFO: User logged out=badguy Clearly, attackers can use this same mechanism to insert arbitrary log entries.</p>
Context Notes	Log forging vulnerabilities occur when: 1. Data enters an application from an untrusted source. 2. The data is written to an application or system log file. Applications typically use log files to store a history of events or transactions for later review, statistics gathering, or debugging. Depending on the nature of the application, the task of reviewing log files may be performed manually on an as-needed basis or automated with a tool that automatically culls logs for important events or trending information. Interpretation of the log files may be hindered or misdirected if an attacker can supply data to the application that is subsequently logged verbatim. In the most benign case, an attacker may be able to insert false entries into the log file by providing the application with input that includes appropriate characters. If the log file is processed automatically, the attacker can render the file unusable by corrupting the format of the file or injecting unexpected characters. A more subtle attack might involve skewing the log file statistics. Forged or otherwise, corrupted log files can be used to cover an attacker's tracks or even to implicate another party in the commission of a malicious act [22]. In the worst case, an attacker may inject code or other commands into the log file and take advantage of a vulnerability in the log processing utility [17].
References	[17] G. Hoglund and G. McGraw. "Exploiting Software: How to Break Code". Addison-Wesley. February 2004. [22] A. Muffet. "The night the log was forged". < http://doc.novsu.ac.ru/oreilly/tcpip/puis/ch10_05.htm >.
Node Relationships	ChildOf - Output Validation (116)
Source Taxonomies	7 Pernicious Kingdoms - Log Forging
Applicable Platforms	All

Path 1:

Query Name - **Log_Forgery**


Severity -  **Warning**

```
55. public void prepare() throws Exception //planetform.java
    ...
    61. log.debug("Loading Planet ... "+getPlanetid());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
107. public String deleteGroup() //planetform.java
    ...
    111. log.debug("Deleting Planet Group ... "+getGroupid());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
68. public String deletePlanet() //planetslist.java
    ...
    72. log.debug("Deleting Planet ... "+getPlanetid());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
82. public boolean handleRequest(HttpServletRequest request, HttpServletResponse response) //multiplanetrequestmapper.java
    ...
    95. log.debug("evaluating [" +request.getRequestURI()+"]");
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
64. public PlanetRequest(HttpServletRequest request) //planetrequest.java
    ...
    70. String path = request.getPathInfo();
    ...
    72. log.debug("parsing path "+path);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
64. public PlanetRequest(HttpServletRequest request) //planetrequest.java
    ...
    70. String path = request.getPathInfo();
    ...
    78. path = path.substring(1);
    ...
    82. path = path.substring(0, path.length() - 1);
    ...
    85. String[] pathElements = path.split("/", 2);
    ...
    87. this.planetHandle = pathElements[0];
    ...
    103. log.debug("planetHandle = "+this.planetHandle);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
64. public PlanetRequest(HttpServletRequest request) //planetrequest.java
    ...
    70. String path = request.getPathInfo();
    ...
    78. path = path.substring(1);
    ...
    82. path = path.substring(0, path.length() - 1);
    ...
    85. String[] pathElements = path.split("/", 2);
    ...
    96. pathInfo = pathElements[1];
    ...
    104. log.debug("pathInfo = "+this.pathInfo);
```

Path 8:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
57. public WeblogCommentRequest(HttpServletRequest request) //weblogcommentrequest.java
    ...
    125. this.name = Utilities.removeHTML(request.getParameter("name"));
```



```
139. public static String removeHTML(String str) //utilities.java
    140. return removeHTML(str, true);
```



```
152. public static String removeHTML(String str, boolean addSpace) //utilities.java
    ...
    159. return str;
```



```
139. public static String removeHTML(String str) //utilities.java
    140. return removeHTML(str, true);
```



```
57. public WeblogCommentRequest(HttpServletRequest request) //weblogcommentrequest.java
    ...
    125. this.name = Utilities.removeHTML(request.getParameter("name"));
```



```
156. public void doPost(HttpServletRequest request, HttpServletResponse response) //commentservlet.java
    ...
    192. commentRequest = new WeblogCommentRequest(request);
    ...
    204. entry = commentRequest.getWeblogEntry();
```



```
202. public WeblogEntry getWeblogEntry() //weblogcommentrequest.java
    ...
    207. weblogEntry = wmgr.getWeblogEntryByAnchor(getWeblog(), weblogAnchor);
```



```
185. public Weblog getWeblog() //weblogrequest.java
    ...
    190. weblog = umgr.getWebsiteByHandle(weblogHandle, Boolean.TRUE);
```



```
508. public Weblog getWebsiteByHandle(String handle, Boolean enabled) //jpausermanagerimpl.java
    ...
    523. log.debug("weblogHandleToId CACHE HIT - "+handle);
```

Path 9:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
57. public WeblogCommentRequest(HttpServletRequest request) //weblogcommentrequest.java
    ...
    125. this.name = Utilities.removeHTML(request.getParameter("name"));
```



```
139. public static String removeHTML(String str) //utilities.java
    140. return removeHTML(str, true);
```



```
152. public static String removeHTML(String str, boolean addSpace) //utilities.java
    ...
    159. return str;
```



```
139. public static String removeHTML(String str) //utilities.java
    140. return removeHTML(str, true);
```



```
57. public WeblogCommentRequest(HttpServletRequest request) //weblogcommentrequest.java
    ...
    125. this.name = Utilities.removeHTML(request.getParameter("name"));
```

156. public void doPost(HttpServletRequest request, HttpServletResponse response) //commentervlet.java

```
...
192. commentRequest = new WeblogCommentRequest(request);
...
204. entry = commentRequest.getWeblogEntry();
```

202. public WeblogEntry getWeblogEntry() //weblogcommentrequest.java

```
...
207. weblogEntry = wmgr.getWeblogEntryByAnchor(getWeblog(), weblogAnchor);
```

185. public Weblog getWeblog() //weblogrequest.java

```
...
190. weblog = umgr.getWebsiteByHandle(weblogHandle, Boolean.TRUE);
```

508. public Weblog getWebsiteByHandle(String handle, Boolean enabled) //jpausermanagerimpl.java

```
...
543. log.debug("weblogHandleToId CACHE MISS - "+handle);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Log_Forgery**

Severity -  **Warning**

67. public void myPrepare() //modifyuser.java

```
...
75. setUser(mgr.getUserByUserName(getUserName(), null));
```

634. public User getUserByUserName(String userName, Boolean enabled) //jpausermanagerimpl.java

```
...
649. log.debug("userNameToIdMap CACHE HIT - "+userName);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 11:

Query Name - **Log_Forgery**

Severity -  **Warning**

67. public void myPrepare() //modifyuser.java

```
...
75. setUser(mgr.getUserByUserName(getUserName(), null));
```

634. public User getUserByUserName(String userName, Boolean enabled) //jpausermanagerimpl.java

```
...
682. log.debug("userNameToIdMap CACHE MISS - "+userName);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 12:

Query Name - **Log_Forgery**

Severity -  **Warning**

77. public WeblogPageRequest(HttpServletRequest request) //weblogpagerequest.java

```
...
187. String anchor = request.getParameter("entry");
...
189. this.weblogAnchor = anchor;
```

114. public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException //pageservlet.java

```
...
139. pageRequest = new WeblogPageRequest(request);
...
167. if (!pageRequest.isLoggedIn()) {
...
186. if ((!this.excludeOwnerPages || !pageRequest.isLoggedIn()) && request.getAttribute("skipCache") == null) {
...
233. } else if ("page".equals(pageRequest.getContext())) {
234. page = pageRequest.getWeblogPage();
...
301. if (!pageRequest.getWeblog().isEnabledMultiLang()) {
```

```
...
305. if (pageRequest.getWeblogAnchor() != null) {
...
309. WeblogEntry entry = pageRequest.getWeblogEntry();
```

```
330. public WeblogEntry getWeblogEntry() //weblogpagerequest.java
```

```
...
335. weblogEntry = wmgr.getWeblogEntryByAnchor(getWeblog(), weblogAnchor);
```

```
694. public WeblogEntry getWeblogEntryByAnchor(Weblog website, //jpaweblogmanagerimpl.java
695. String anchor) throws WebloggerException {
...
704. String mappingKey = website.getHandle()+"."+anchor;
...
712. log.debug("entryAnchorToIdMap CACHE HIT - "+mappingKey);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 13:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
77. public WeblogPageRequest(HttpServletRequest request) //weblogpagerequest.java
```

```
...
187. String anchor = request.getParameter("entry");
...
189. this.weblogAnchor = anchor;
```

```
114. public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException //pageservlet.java
...
139. pageRequest = new WeblogPageRequest(request);
...
167. if (!pageRequest.isLoggedIn()) {
...
186. if ((!this.excludeOwnerPages || !pageRequest.isLoggedIn()) && request.getAttribute("skipCache") == null) {
...
233. } else if ("page".equals(pageRequest.getContext())) {
234. page = pageRequest.getWeblogPage();
...
301. if (!pageRequest.getWeblog().isEnabledMultiLang()) {
...
305. if (pageRequest.getWeblogAnchor() != null) {
...
309. WeblogEntry entry = pageRequest.getWeblogEntry();
```

```
330. public WeblogEntry getWeblogEntry() //weblogpagerequest.java
...
335. weblogEntry = wmgr.getWeblogEntryByAnchor(getWeblog(), weblogAnchor);
```

```
694. public WeblogEntry getWeblogEntryByAnchor(Weblog website, //jpaweblogmanagerimpl.java
695. String anchor) throws WebloggerException {
...
704. String mappingKey = website.getHandle()+"."+anchor;
...
734. log.debug("entryAnchorToIdMap CACHE MISS - "+mappingKey);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 14:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
515. private boolean processReferrer(HttpServletRequest request) //pageservlet.java
...
548. reqsb.append(request.getQueryString());
...
550. String requestUrl = reqsb.toString();
...
600. referrer.setRequestUrl(requestUrl);
```

```
54. public void setRequestUrl(String requestUrl) //incomingreferrer.java
55. this.requestUrl = requestUrl;
```

```
515. private boolean processReferrer(HttpServletRequest request) //pageservlet.java
```

```
...
600. referrer.setRequestUrl(requestUrl);
...
606. refQueue.processReferrer(referrer);
```



```
130. public void processReferrer(IncomingReferrer referrer) //referrerqueueanagerimpl.java
...
133. mLogger.debug("QUEUING: "+referrer.getRequestUrl());
```



```
50. public String getRequestUrl() //incomingreferrer.java
51. return requestUrl;
```



```
130. public void processReferrer(IncomingReferrer referrer) //referrerqueueanagerimpl.java
...
133. mLogger.debug("QUEUING: "+referrer.getRequestUrl());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 15:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
71. public void myPrepare() //planetsubscriptions.java
...
83. log.error("Error looking up planet group - "+getGroupHandle(), ex);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 16:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
99. public String save() //planetsubscriptions.java
...
109. log.debug("Adding New Subscription - "+getSubUrl());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 17:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
99. public String save() //planetsubscriptions.java
...
118. log.debug("Adding Existing Subscription - "+getSubUrl());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 18:

Query Name - **Log_Forgery**

Severity -  **Warning**

```
77. public WeblogPageRequest(HttpServletRequest request) //weblogpagerequest.java
...
187. String anchor = request.getParameter("entry");
...
189. this.weblogAnchor = anchor;
```



```
515. private boolean processReferrer(HttpServletRequest request) //pageservlet.java
...
525. pageRequest = new WeblogPageRequest(request);
...
586. } else if (BlacklistChecker.checkReferrer(pageRequest.getWeblog(), referrerUrl)) {
```



```
65. public static boolean checkReferrer(Weblog website, String referrerURL) //blacklistchecker.java
...
70. website.getBlacklist(), stringRules, regexRules, null);
```



```
392. public String getBlacklist() //weblog.java
```



```
154. ResultSet users = userquery.executeQuery();
...
158. String passphrase = users.getString(2);
159. newprops.put(username, passphrase);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Password_Misuse**

Severity -  **Warning**

```
169. private static void encryptionOn( //passwordutility.java
...
180. ResultSet users = userQuery.executeQuery();
...
184. String passphrase = users.getString(2);
185. props.put(username, passphrase);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Password_Misuse**

Severity -  **Warning**

```
169. private static void encryptionOn( //passwordutility.java
...
180. ResultSet users = userQuery.executeQuery();
...
183. String username = users.getString(1);
...
185. props.put(username, passphrase);
...
187. Enumeration usernames = props.keys();
...
190. String username = (String)usernames.nextElement();
191. String passphrase = (String)props.get(username);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Password_Misuse**

Severity -  **Warning**

```
169. private static void encryptionOn( //passwordutility.java
...
180. ResultSet users = userQuery.executeQuery();
...
183. String username = users.getString(1);
...
185. props.put(username, passphrase);
...
187. Enumeration usernames = props.keys();
...
190. String username = (String)usernames.nextElement();
191. String passphrase = (String)props.get(username);
...
193. userUpdate.setString(1, Utilities.encodePassword(passphrase, algorithm));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Password_Misuse**

Severity -  **Warning**

```
169. private static void encryptionOn( //passwordutility.java
...
180. ResultSet users = userQuery.executeQuery();
...
183. String username = users.getString(1);
...
185. props.put(username, passphrase);
...
187. Enumeration usernames = props.keys();
...
190. String username = (String)usernames.nextElement();
191. String passphrase = (String)props.get(username);
...
193. userUpdate.setString(1, Utilities.encodePassword(passphrase, algorithm));
```



```
464. public static String encodePassword(String password, String algorithm) //utilities.java
    {
        ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:
Query Name - Password_Misuse
Severity -  **Warning**

```
169. private static void encryptionOn( //passwordutility.java
    {
        ...
        180. ResultSet users = userQuery.executeQuery();
        ...
        183. String username = users.getString(1);
        ...
        185. props.put(username, passphrase);
        ...
        187. Enumeration usernames = props.keys();
        ...
        190. String username = (String)usernames.nextElement();
        191. String passphrase = (String)props.get(username);
        ...
        193. userUpdate.setString(1, Utilities.encodePassword(passphrase, algorithm));
    }
}
```

```
464. public static String encodePassword(String password, String algorithm) //utilities.java
    {
        465. byte[] unencodedPassword = password.getBytes();
        ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:
Query Name - Password_Misuse
Severity -  **Warning**

```
169. private static void encryptionOn( //passwordutility.java
    {
        ...
        180. ResultSet users = userQuery.executeQuery();
        ...
        183. String username = users.getString(1);
        ...
        185. props.put(username, passphrase);
        ...
        187. Enumeration usernames = props.keys();
        ...
        190. String username = (String)usernames.nextElement();
        191. String passphrase = (String)props.get(username);
        ...
        193. userUpdate.setString(1, Utilities.encodePassword(passphrase, algorithm));
    }
}
```

```
464. public static String encodePassword(String password, String algorithm) //utilities.java
    {
        ...
        474. return password;
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Improper Resource Shutdown or Release	
CWE ID	404
Description	The program can potentially fail to release or incorrectly release a system resource. A resource is not properly cleared and made available for re-use.
Functional Area	Non-specific
Potential Mitigations	It is good practice to be responsible for freeing all resources you allocate. Memory should be allocated/freed using matching functions such as malloc/free, new/delete, and new[]/delete[].
Demonstrative Examples	The following method never closes the file handle it opens. The Finalize() method for StreamReader eventually calls Close(), but there is no guarantee as to how long it will take before the Finalize() method is invoked. In fact, there is no guarantee that Finalize() will ever be invoked. In a busy environment, this can result in the VM using up all of its available file handles.

```
private void processFile(string fName) {
    StreamWriter sw = new
    StreamWriter(fName);
    string line;
    while ((line = sr.ReadLine()) != null) processLine(line);
}
```

If an exception occurs after establishing the database connection and before the same connection closes, the pool of database connections may become exhausted. If the number of available connections is exceeded, other users cannot access this resource, effectively denying access to the application. Using the following database connection pattern will ensure that all opened connections are closed. The `con.close()` call should be the first executable statement in the finally block.

```
try {
    Connection con = DriverManager.getConnection(some_connection_string)
}
catch ( Exception e ) {
    log( e )
}
finally {
    con.close()
}
```

Under normal conditions the following C# code executes a database query, processes the results returned by the database, and closes the allocated `SqlConnection` object. But if an exception occurs while executing the SQL or processing the results, the `SqlConnection` object is not closed. If this happens often enough, the database will run out of available cursors and not be able to execute any more SQL queries.

C# Example:

```
.....
SqlConnection conn = new SqlConnection(connString);
SqlCommand cmd = new SqlCommand(queryString);
cmd.Connection = conn; conn.Open();
SqlDataReader rdr = cmd.ExecuteReader();
HarvestResults(rdr);
conn.Connection.Close();
...
```

The following C function does not close the file handle it opens if an error occurs. If the process is long-lived, the process can run out of file handles.

C Example:

```
int decodeFile(char* fName) {
    char buf[BUF_SZ];
    FILE* f = fopen(fName, "r");
    if (!f) {
        printf("cannot open %s\n", fName);
        return DECODE_FAIL;
    }
    else {
        while (fgets(buf, BUF_SZ, f)) {
            if (!checkChecksum(buf)) {
                return DECODE_FAIL;
            }
            else {
                decodeBlock(buf);
            }
        }
        fclose(f);
        return DECODE_SUCCESS;
    }
}
```

In this example, the program fails to use matching functions such as `malloc/free`, `new/delete`, and `new[]/delete[]` to allocate/deallocate the resource.

C++ Example:

```
class A{
    void foo();
};
void A::foo(){
    int *ptr;
    ptr = (int*)malloc(sizeof(int));
    delete ptr;
}
```

In this example, the program calls the `delete[]` function on non-heap memory.


C++ Example:

```
class A{
    void foo(bool);
};
void A::foo(bool heap) {
    int localArray[2] = {11,22};
    int *p = localArray;
    if (heap){
        p = new int[2];
    }
    delete[] p;
}
```

	CVE-2001-0830 - Sockets not properly closed when attacker repeatedly connects and disconnects from server. CVE-2002-1372 - Return values of file/socket operations not checked, allowing resultant consumption of file descriptors.
Context Notes	<p>It is important to be consistent with how and where you free memory in a function. If you allocate memory that you intend to free upon completion of the function, you must be sure to free the memory at all exit points for that function including error conditions.</p> <p>Before freeing a pointer, the programmer should make sure that the pointer was previously allocated and that the memory belongs to the programmer. Freeing an unallocated pointer will cause undefined behavior in the program.</p> <p>Can be resultant from improper error handling or insufficient resource tracking.</p> <p>Overlaps memory leaks, asymmetric resource consumption, malformed input errors.</p> <p>Most unreleased resource issues result in general software reliability problems, but if an attacker can intentionally trigger a resource leak, the attacker might be able to launch a denial of service attack by depleting the resource pool.</p> <p>Resource leaks have at least two common causes: - Error conditions and other exceptional circumstances. - Confusion over which part of the program is responsible for releasing the resource.</p>
Node Relationships	ChildOf - Resource Management Errors (399) Peer - Memory Leak (401) Peer - Asymmetric Resource Consumption (Amplification) (405)
Source Taxonomies	PLOVER - Improper resource shutdown or release 7 Pernicious Kingdoms - Unreleased Resource
Applicable Platforms	All

Path 1:

Query Name - DoS_by_Unreleased_Resources


Severity - Warning

```
53. public Technorati() throws IOException //technorati.java
    ...
    55. BufferedReader br = new BufferedReader(new InputStreamReader(is));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - DoS_by_Unreleased_Resources


Severity - Warning

```
60. public String render(final WeblogEntryComment comment, String text) //autoformatplugin.java
    ...
    72. BufferedReader br = new BufferedReader(new StringReader(text));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - DoS_by_Unreleased_Resources


Severity - Warning

```
80. public String render(WeblogEntry entry, String str) //convertlinebreaksplugin.java
    ...
    94. BufferedReader br = new BufferedReader(new StringReader(str));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - DoS_by_Unreleased_Resources


Severity - Warning

```
318. private String loadTemplateFile(File templateFile) //sharedthemefromdir.java
    ...
    328. FileInputStream stream = new FileInputStream(templateFile);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - DoS_by_Unreleased_Resources

Severity - Warning

```
318. private String loadTemplateFile(File templateFile) //sharedthemefromdir.java
    ...
    329. InputStreamReader reader = new InputStreamReader(stream, "UTF-8");
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:
Query Name - DoS_by_Unreleased_Resources
Severity - ⚠️Warning

```
56. public int validate(WeblogEntryComment comment, RollerMessages messages) //akismetcommentvalidator.java
    |
    |...
    |84. BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:
Query Name - DoS_by_Unreleased_Resources
Severity - ⚠️Warning

```
343. private String fileToString( InputStream is ) throws java.io.IOException //bookmarktest.java
    |
    |344. BufferedReader br = new BufferedReader(new InputStreamReader(is));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Hard-Coded Password	
CWE ID	259
Description	Hard coded passwords may compromise system security in a way that cannot be easily remedied. It is never a good idea to hardcode a password. Not only does hardcoding a password allow all of the project's developers to view the password, it also makes fixing the problem extremely difficult. Once the code is in production, the password cannot be changed without patching the software. If the account protected by the password is compromised, the owners of the system will be forced to choose between security and availability.
Likelihood of Exploit	Very High
Weakness Ordinality	Primary (Weakness exists independent of other weaknesses)
Causal Nature	Explicit (This is an explicit weakness resulting from behavior of the developer)
Common Consequences	Authentication: If hard-coded passwords are used, it is almost certain that malicious users will gain access through the account in question.
Potential Mitigations	Design (for default accounts): Rather than hard code a default username and password for first time logins, utilize a "first login" mode which requires the user to enter a unique strong password. Design (for front-end to back-end connections): Three solutions are possible, although none are complete. The first suggestion involves the use of generated passwords which are changed automatically and must be entered at given time intervals by a system administrator. These passwords will be held in memory and only be valid for the time intervals. Next, the passwords used should be limited at the back end to only performing actions valid to for the front end, as opposed to having full access. Finally, the messages sent should be tagged and checksummed with time sensitive values so as to prevent replay style attacks.
Demonstrative Examples	The following code uses a hardcoded password to connect to a database: <div><pre>... DriverManager.getConnection(url, "scott", "tiger"); ...</pre></div>

This code will run successfully, but anyone who has access to it will have access to the password. Once the program is shipped, there is no going back from the database user "scott" with a password of "tiger" unless the program is patched. A devious employee with access to this information can use it to break into the system. Even worse, if attackers have access to the bytecode for application, they can use the javap -c command to access the disassembled code, which will contain the values of the passwords used. The result of this operation might look something like the following for the example above: javap -c ConnMngr.class 22: ldc #36; //String jdbc:mysql://ixne.com/rxsql 24: ldc #38; //String scott 26: ldc #17; //String tiger

C/C++ Example:

```
int VerifyAdmin(char *password) {
if (strcmp(password, "Mew!")) {
    printf("Incorrect Password!\n");
    return(0)
}
printf("Entering Diagnostic Mode...\n"); return(1);
}
```

Java Example:

```
int VerifyAdmin(String password) {
    if (passwd.Equals("Mew!")) {
        return(0)
    } //Diagnostic Mode return(1);
}
```

Every instance of this program can be placed into diagnostic mode with the same password. Even worse is the fact that if this program is distributed as a binary-only distribution, it is very difficult to change that password or disable this "functionality."

Context Notes	The use of a hard-coded password has many negative implications -- the most significant of these being a failure of authentication measures under certain circumstances. On many systems, a default administration account exists which is set to a simple default password which is hard-coded into the program or device. This hard-coded password is the same for each device or system of this type and often is not changed or disabled by end users. If a malicious user comes across a device of this kind, it is a simple matter of looking up the default password (which is freely available and public on the internet) and logging in with complete access. In systems which authenticate with a back-end service, hard-coded passwords within closed source or drop-in solution systems require that the back-end service use a password which can be easily discovered. Client-side systems with hard-coded passwords propose even more of a threat, since the extraction of a password from a binary is exceedingly simple.
Node Relationships	ChildOf - Credentials Management (255) Peer - Use of Hard-coded Cryptographic Key (321) Peer - Storing Passwords in a Recoverable Format (257) ChildOf - Weaknesses Examined by SAMATE (630)
Source Taxonomies	7 Pernicious Kingdoms - Password Management: Hard-Coded Password CLASP - Use of hard-coded password
Applicable Platforms	All

Path 1:

Query Name - **Hardcoded_Password**

Severity -  **Warning**

```
37. public class UserEntry extends Entry //userentry.java
    |
    | ...
    | 44. public static final String PASSWORD = "password";
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Hardcoded_Password**

Severity -  **Warning**

```
31. public abstract class AappTest extends TestCase //aapptest.java
    |
    | ...
    | 35. private static final String DEFAULT_PASSWORD = "admin";
```

[\[Table of Contents\]](#) [\[Description\]](#)

Unsynchronized Access to Shared Data	
CWE ID	567
Description	Failure to synchronize shared data, such as static variables across threads, can lead to undefined behavior in your program and unpredictable values in your data.
Potential Mitigations	A shared variable vulnerability can be prevented by removing the use of static variables used between servlets or to provide protection when shared access is absolutely needed. In this case, access should be synchronized.
Demonstrative Examples	Java Example: <pre>public static class Counter extends HttpServlet { static int count = 0; protected void doGet(HttpServletRequest in, HttpServletResponse out) throws ServletException, IOException { out.setContentType("text/plain"); PrintWriter p = out.getWriter(); count++; p.println(count + " hits so far!"); } }</pre>
Context Notes	The vulnerability can exist in servlets because a servlet is multi-threaded, and shared static variables are not protected from concurrent access. This is a typical programming mistake in J2EE applications, since the multi-threading is handled by the framework. The use of shared variables can be exploited by attackers to gain information or to cause denial of service conditions. If this shared data contains sensitive information, it may be manipulated or displayed in another user session. If this data is used to control the application, its value can be manipulated to cause the application to crash or perform poorly.
Node Relationships	ChildOf - Concurrency Issues (557) ChildOf - Concurrency Issues (557)
Applicable Platforms	All

Path 1:

Query Name - Thread_Safety_Issue

Severity - ⚠️Warning

```
75. private String getAbsoluteUrl(HttpServletRequest request) //initfilter.java
    ...
79. String fullUrl = request.getRequestURL().toString();
    ...
84. return fullUrl.substring(0, fullUrl.length()-1);

49. public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) //initfilter.java
    ...
59. String absPath = this.getAbsoluteUrl(request);
    ...
62. WebloggerRuntimeConfig.setAbsoluteContextURL(absPath);

179. public static void setAbsoluteContextURL(String url) //webloggerruntimeconfig.java
    180. absoluteContextURL = url;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Unsafe URL Redirection	
CWE ID	601
Description	An http parameter may contain a URL value and could cause the web application to redirect the request to the specified URL. By modifying the URL value to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials. The fact that the server name in the modified link is identical to the original site helps the attacker by giving his phishing attempts a more reliable appearance.
Observed Examples	- CVE-2005-4206 - Blackboard Learning and Community Portal System in Academic Suite 6.3.1.424, 6.2.3.23, and other versions before 6 allows remote attackers to redirect users to other URLs and conduct phishing attacks via a modified url parameter to frameset.jsp, which loads the URL into a frame and causes it to appear to be part of a valid page.
Context Notes	Phishing is a general term for deceptive attempts to coerce private information from users that will be used for identity theft.
Node Relationships	ChildOf - API Abuse (227)
Source Taxonomies	Anonymous Tool Vendor (under NDA) -

Path 1:

Query Name - URL_Redirection_Attack

Severity - ⚠️Warning

```
82. public boolean handleRequest(HttpServletRequest request, HttpServletResponse response) //multiplanetrequestmapper.java
    ...
162. String redirectUrl = request.getRequestURI() + "/";
    ...
167. response.sendRedirect(redirectUrl);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - URL_Redirection_Attack

Severity - ⚠️Warning

```
76. public void doFilter(ServletRequest request, ServletResponse response, //schemeenforcementfilter.java
    ...
94. redirect += req.getRequestURI();
    ...
100. res.sendRedirect(redirect);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - URL_Redirection_Attack


Severity - ⚠️Warning

```
76. public void doFilter(ServletRequest request, ServletResponse response, //schemeenforcementfilter.java
    ...
110. redirect += req.getRequestURI();
    ...
116. res.sendRedirect(redirect);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - URL_Redirection_Attack


Severity - Warning

```
91. public boolean handleRequest(HttpServletRequest request, HttpServletResponse response) //weblogrequestmapper.java
    ...
188. String redirectUrl = request.getRequestURI() + "/";
    ...
190. redirectUrl += "?" + request.getQueryString();
    ...
193. response.sendRedirect(redirectUrl);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - URL_Redirection_Attack


Severity - Warning

```
82. public boolean handleRequest(HttpServletRequest request, HttpServletResponse response) //multiplanetrequestmapper.java
    ...
164. redirectUrl += "?" + request.getQueryString();
    ...
167. response.sendRedirect(redirectUrl);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - URL_Redirection_Attack


Severity - Warning

```
76. public void doFilter(ServletRequest request, ServletResponse response, //schemeenforcementfilter.java
    ...
97. redirect += "?" + req.getQueryString();
    ...
100. res.sendRedirect(redirect);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - URL_Redirection_Attack


Severity - Warning

```
76. public void doFilter(ServletRequest request, ServletResponse response, //schemeenforcementfilter.java
    ...
113. redirect += "?" + req.getQueryString();
    ...
116. res.sendRedirect(redirect);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - URL_Redirection_Attack

Severity - Warning

```
91. public boolean handleRequest(HttpServletRequest request, HttpServletResponse response) //weblogrequestmapper.java
    ...
190. redirectUrl += "?" + request.getQueryString();
    ...
193. response.sendRedirect(redirectUrl);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Object Model Violation: Just One of Equals and Hascode Defined	
CWE ID	581
Description	Software fails to maintain equal hashcodes for equal objects.
Common Consequences	Failure to uphold this invariant is likely to cause trouble if objects of this class are stored in a collection. If the objects of the class in question are used as a key in a Hashtable or if they are inserted into a Map or Set, it is critical that equal objects have equal hashcodes.
Context Notes	Java objects are expected to obey a number of invariants related to equality. One of these invariants is that equal

objects must have equal hashcodes. In other words, if a.equals(b) == true then a.hashCode() == b.hashCode().

Node Relationships	ChildOf - Failure to Follow Specification (573)
Applicable Platforms	Java

Path 1:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
44. public class UtilitiesModel implements Model //utilitiesmodel.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
41. public class OldUtilities //oldutilities.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
46. public class UtilitiesModel // implements Model //utilitiesmodel.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
46. public class UtilitiesModel implements Model //utilitiesmodel.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
26. public class OldStringUtils //oldstringutils.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
41. public class OldUtilities //oldutilities.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
383. public class WeblogResourceComparator implements Comparator //resources.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
33. public abstract class Entry //entry.java
    {
        // ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
31. public abstract class EntrySet extends Entry //entryset.java
    {
        // ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
36. public class MemberEntry extends Entry //memberentry.java
    {
        // ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 11:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
37. public class UserEntry extends Entry //userentry.java
    {
        // ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 12:

Query Name - **Equals_without_GetHashCode**

Severity -  **Warning**

```
41. public class WeblogEntry extends Entry //weblogentry.java
    {
        // ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Improper Transaction Handling	
CWE ID	10016
Description	If an exception occurred during a database transaction, a rollback should be committed.
Likelihood of Exploit	Low
Common Consequences	Database lock.
Potential Mitigations	Perform rollback in the catch clause.
Applicable Platforms	All

Path 1:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
139. public void flush() throws PlanetException //jpapersistencestrategy.java
    {
        |140. try {
            // ...
        }
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
135. public void flush() throws WebloggerException //jpapersistencestrategy.java
    | 136. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
268. private void upgradeTo130(Connection con, boolean runScripts) throws StartupException //databaseinstaller.java
    | ...
    | 270. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
320. private void upgradeTo200(Connection con, boolean runScripts) throws StartupException //databaseinstaller.java
    | ...
    | 322. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
408. private void upgradeTo210(Connection con, boolean runScripts) throws StartupException //databaseinstaller.java
    | ...
    | 410. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
579. private void upgradeTo300(Connection con, boolean runScripts) throws StartupException //databaseinstaller.java
    | ...
    | 581. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
689. private void upgradeTo400(Connection con, boolean runScripts) throws StartupException //databaseinstaller.java
    | ...
    | 714. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
689. private void upgradeTo400(Connection con, boolean runScripts) throws StartupException //databaseinstaller.java
    | ...
    | 772. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
689. private void upgradeTo400(Connection con, boolean runScripts) throws StartupException //databaseinstaller.java
    |
```

```
...
922. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
689. private void upgradeTo400(Connection con, boolean runScripts) throws StartupException //databaseinstaller.java
...
977. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 11:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
689. private void upgradeTo400(Connection con, boolean runScripts) throws StartupException //databaseinstaller.java
...
1024. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 12:

Query Name - **Improper_Transaction_Handling**

Severity -  **Warning**

```
119. public void runScript( //sqlscriptrunner.java
...
126. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Data Leaking Between Users	
CWE ID	488
Description	Data can "bleed" from one session to another through member variables of singleton objects, such as Servlets, and objects from a shared pool.
Demonstrative Examples	<div>The following Servlet stores the value of a request parameter in a member field and then later echoes the parameter value to the response output stream.</div> <div><pre>public class GuestBook extends HttpServlet { String name; protected void doPost (HttpServletRequest req, HttpServletResponse res) { name = req.getParameter("name"); ... out.println(name + ", thanks for visiting!"); } }</pre></div> <div>While this code will work perfectly in a single-user environment, if two users access the Servlet at approximately the same time, it is possible for the two request handler threads to interleave in the following way: Thread 1: assign "Dick" to name Thread 2: assign "Jane" to name Thread 1: print "Jane, thanks for visiting!" Thread 2: print "Jane, thanks for visiting!" Thereby showing the first user the second user's name.</div>
Context Notes	Many Servlet developers do not understand that, unless a Servlet implements the SingleThreadModel interface, the Servlet is a singleton; there is only one instance of the Servlet, and that single instance is used and re-used to handle multiple requests that are processed simultaneously by different threads. A common result of this misunderstanding is that developers use Servlet member fields in such a way that one user may inadvertently see another user's data. In other words, storing user data in Servlet member fields introduces a data access race condition.
Node Relationships	ChildOf - Encapsulation (485)
Source Taxonomies	7 Pernicious Kingdoms - Data Leaking Between Users
Applicable Platforms	All

Path 1:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
46. public class FeedServlet extends HttpServlet //feedservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
46. public class HomepageServlet extends HttpServlet //homepageservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
46. public class OpmlServlet extends HttpServlet //opmlservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
46. public class PageServlet extends HttpServlet //pageservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
44. public class CommentAuthenticatorServlet extends HttpServlet //commentauthenticatorservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
70. public class CommentServlet extends HttpServlet //commentservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
56. public class FeedServlet extends HttpServlet //feedservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
65. public class PageServlet extends HttpServlet //pageservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
54. public class PlanetFeedServlet extends HttpServlet //planetfeedservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
53. public class PreviewResourceServlet extends HttpServlet //previewresourceservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 11:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
57. public class PreviewServlet extends HttpServlet //previewervlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 12:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
44. public class PreviewThemeImageServlet extends HttpServlet //previewthemeimageservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 13:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
54. public class ResourceServlet extends HttpServlet //resourceservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 14:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
51. public class RSDServlet extends HttpServlet //rsdservlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 15:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
53. public class SearchServlet extends HttpServlet //searchervlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 16:

Query Name - **Singleton_HTTPServlet**

Severity -  **Warning**

```
57. public class TrackbackServlet extends HttpServlet //trackbackervlet.java
{
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 17:
Query Name - Singleton_HTTPServlet
Severity - ⚠️Warning

```
52. public class RedirectServlet extends HttpServlet //redirectservlet.java
    {
        ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 18:
Query Name - Singleton_HTTPServlet
Severity - ⚠️Warning

```
41. public class AdminServlet extends HttpServlet //adminservlet.java
    {
        ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 19:
Query Name - Singleton_HTTPServlet
Severity - ⚠️Warning

```
58. public class AtomServlet extends HttpServlet //atomservlet.java
    {
        ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Often Misused: Exception Handling	
CWE ID	248
Description	Failing to catch an exception thrown from a dangerous function, can potentially cause the program to crash.
Demonstrative Examples	<p>The <code>_alloca()</code> function allocates memory on the stack. If an allocation request is too large for the available stack space, <code>_alloca()</code> throws an exception. If the exception is not caught, the program will crash, potentially enabling a denial of service attack. <code>_alloca()</code> has been deprecated as of Microsoft Visual Studio 2005®. It has been replaced with the more secure <code>_alloca_s()</code>.</p> <p><code>EnterCriticalSection()</code> can raise an exception, potentially causing the program to crash. Under operating systems prior to Windows 2000, the <code>EnterCriticalSection()</code> function can raise an exception in low memory situations. If the exception is not caught, the program will crash, potentially enabling a denial of service attack.</p>
Node Relationships	ChildOf - API Abuse (227) Peer - Error Conditions, Return Values, Status Codes (389)
Source Taxonomies	7 Pernicious Kingdoms - Often Misused: Exception Handling
Applicable Platforms	C C++ Java .NET

Path 1:
Query Name - Improper_Exception_Handling
Severity - ⚠️Warning

```
77. public void printStackTrace(PrintWriter s) //fetcherexception.java
    {
        ...
        80. s.println(super.getMessage());
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:
Query Name - Improper_Exception_Handling
Severity - ⚠️Warning

```
77. public void printStackTrace(PrintWriter s) //fetcherexception.java
    {
        ...
        82. s.println("--- ROOT CAUSE ---");
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
242. private FeedFetcherCache getRomeFetcherCache() //romefeedfetcher.java
    ...
    254. "\\${user.home}",System.getProperty("user.home"));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
242. private FeedFetcherCache getRomeFetcherCache() //romefeedfetcher.java
    ...
    257. if (System.getProperty("catalina.home") != null) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
242. private FeedFetcherCache getRomeFetcherCache() //romefeedfetcher.java
    ...
    259. "\\${catalina.home}",System.getProperty("catalina.home"));
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
125. public int getSubscriptionCount() throws PlanetException //japlanetmanagerimpl.java
    ...
    127. return q.getResultList().size();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
138. public List getTopSubscriptions( //japlanetmanagerimpl.java
    ...
    147. result = q.getResultList();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
138. public List getTopSubscriptions( //japlanetmanagerimpl.java
    ...
    153. result = q.getResultList();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
193. public List getPlanets() throws PlanetException //japlanetmanagerimpl.java
    ...
    194. return (List)strategy.getNamedQuery("Planet.getAll").getResultList();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
207. public List getGroups(Planet planet) throws PlanetException //jpaplanetmanagerimpl.java
    |
    |...
    |210. return q.getResultList();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 11:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
239. public List getSubscriptions() throws PlanetException //jpaplanetmanagerimpl.java
    |
    |...
    |241. return q.getResultList();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 12:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
248. public List getEntries(Subscription sub, int offset, int len) throws PlanetException //jpaplanetmanagerimpl.java
    |
    |...
    |256. return q.getResultList();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 13:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
77. public Map getProperties() throws PlanetException //jpapropertiesmanagerimpl.java
    |
    |...
    |80. List list = strategy.getNamedQuery("RuntimeConfigProperty.getAll").getResultList();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 14:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
53. public Technorati() throws IOException //technorati.java
    |
    |...
    |56. mKey = br.readLine();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 15:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
356. public static void copyInputToOutput( //utilities.java
    |
    |...
    |364. count = in.read(buffer, 0, 8192);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 16:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
356. public static void copyInputToOutput( //utilities.java
    |
    |...
    |366. out.write(buffer, 0, count);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 17:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
36. public void testCache() throws Exception //diskfeedinfocachetest.java
    |
    |...
    |41. String buildDir = System.getProperty("ro.build");
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 18:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
54. public FileManagerImpl() //filemanagerimpl.java
    |
    |...
    |60. uploadDir = System.getProperty("user.home") + File.separator+"roller_data"+File.separator+"uploads";
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 19:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
75. public void removeAutoPing(PingTarget pingTarget, Weblog website) throws WebloggerException //jpaautoPingmanagerimpl.java
    |
    |...
    |79. q.executeUpdate();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 20:

Query Name - **Improper_Exception_Handling**

Severity -  **Warning**

```
86. public void removeAllAutoPings() throws WebloggerException //jpaautoPingmanagerimpl.java
    |
    |...
    |88. removeAutoPings(q.getResultList());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Unset Secure Attribute for Sensitive Cookies in HTTPS Session	
CWE ID	614
Description	If the Secure attribute for sensitive cookies in HTTPS sessions is not set, it could cause the user agent to send those cookies in plaintext over an HTTP session.
Observed Examples	- CVE-2004-0462
Node Relationships	ChildOf - Failure to Encrypt Data (311)
Source Taxonomies	Anonymous Tool Vendor (under NDA) -

Path 1:

Query Name - **Cookie_not_Sent_Over_SSL**

Severity -  **Warning**

```
1. <%-- //logout-redirect.jsp
    |
    |...
    |32. response.addCookie(terminate);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 1:

Query Name - **Thread_Safety_Violation_In_Action_Class**

Severity -  **Warning**

```
42. public class PlanetConfig extends PlanetUIAction implements ParameterAware //planetconfig.java
    |
    |...
    |47. private Map parameters = Collections.EMPTY_MAP;
```

Path 2:

Query Name - Thread_Safety_Violation_In_Action_Class

Severity -  Warning

```
42. public class PlanetConfig extends PlanetUIAction implements ParameterAware //planetconfig.java
    |
    |...
    |50. private Map properties = Collections.EMPTY_MAP;
```

Path 3:

Query Name - Thread_Safety_Violation_In_Action_Class

Severity -  Warning

```
42. public class PlanetConfig extends PlanetUIAction implements ParameterAware //planetconfig.java
    |
    |...
    |53. private ConfigDef globalConfigDef = null;
```

Path 4:

Query Name - Thread_Safety_Violation_In_Action_Class

Severity -  Warning

```
35. public class PlanetGroups extends PlanetUIAction //planetgroups.java
    |
    |...
    |40. private PlanetGroupsBean bean = new PlanetGroupsBean();
```

Path 5:

Query Name - Thread_Safety_Violation_In_Action_Class

Severity -  Warning

```
35. public class PlanetGroups extends PlanetUIAction //planetgroups.java
    |
    |...
    |43. private PlanetGroup group = null;
```

Path 6:

Query Name - Thread_Safety_Violation_In_Action_Class

Severity -  Warning

```
38. public class PlanetSubscriptions extends PlanetUIAction //planetsubscriptions.java
    |
    |...
    |43. private String groupHandle = null;
```

Path 7:


Query Name - Thread_Safety_Violation_In_Action_Class

Severity -  Warning

```
38. public class PlanetSubscriptions extends PlanetUIAction //planetsubscriptions.java
    |
    |...
    |46. private PlanetGroup group = null;
```

Path 8:

Query Name - Thread_Safety_Violation_In_Action_Class

Severity -  Warning

```
38. public class PlanetSubscriptions extends PlanetUIAction //planetsubscriptions.java
    |
    |...
    |49. private String subUrl = null;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Thread_Safety_Violation_In_Action_Class**

Severity -  **Warning**

```
30. public abstract class PlanetUIAction extends UIAction //planetuiaction.java
    |
    |...
    |37. private Planet planet = null;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Thread_Safety_Violation_In_Action_Class**

Severity -  **Warning**

```
32. public class CacheInfo extends UIAction //cacheinfo.java
    |
    |...
    |37. private Map stats = Collections.EMPTY_MAP;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 11:

Query Name - **Thread_Safety_Violation_In_Action_Class**

Severity -  **Warning**

```
32. public class CacheInfo extends UIAction //cacheinfo.java
    |
    |...
    |40. private String cache = null;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 12:

Query Name - **Thread_Safety_Violation_In_Action_Class**

Severity -  **Warning**

```
39. public class CreateUser extends UIAction //createuser.java
    |
    |...
    |44. private CreateUserBean bean = new CreateUserBean();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 13:

Query Name - **Thread_Safety_Violation_In_Action_Class**

Severity -  **Warning**

```
46. public class GlobalCommentManagement extends UIAction //globalcommentmanagement.java
    |
    |...
    |54. private GlobalCommentManagementBean bean = new GlobalCommentManagementBean();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 14:

Query Name - **Thread_Safety_Violation_In_Action_Class**

Severity -  **Warning**

```
46. public class GlobalCommentManagement extends UIAction //globalcommentmanagement.java
    |
    |...
    |57. private CommentsPager pager = null;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 15:

Query Name - **Thread_Safety_Violation_In_Action_Class**

Severity -  **Warning**

```
46. public class GlobalCommentManagement extends UIAction //globalcommentmanagement.java
    |
    |...
    |60. private WeblogEntryComment firstComment = null;
```

Path 16:

Query Name - Thread_Safety_Violation_In_Action_Class

Severity - ⚠️Warning

```
46. public class GlobalCommentManagement extends UIAction //globalcommentmanagement.java
    |
    |...
    |63. private WeblogEntryComment lastComment = null;
```

Path 17:

Query Name - Thread_Safety_Violation_In_Action_Class

Severity - ⚠️Warning

```
46. public class GlobalCommentManagement extends UIAction //globalcommentmanagement.java
    |
    |...
    |67. private int bulkDeleteCount = 0;
```

Path 18:

Query Name - Thread_Safety_Violation_In_Action_Class

Severity - ⚠️Warning

```
45. public class GlobalConfig extends UIAction implements ParameterAware //globalconfig.java
    |
    |...
    |50. private Map params = Collections.EMPTY_MAP;
```

Path 19:

Query Name - Thread_Safety_Violation_In_Action_Class

Severity - ⚠️Warning

```
45. public class GlobalConfig extends UIAction implements ParameterAware //globalconfig.java
    |
    |...
    |53. private Map properties = Collections.EMPTY_MAP;
```

Path 20:

Query Name - Thread_Safety_Violation_In_Action_Class

Severity - ⚠️Warning

```
45. public class GlobalConfig extends UIAction implements ParameterAware //globalconfig.java
    |
    |...
    |56. private ConfigDef globalConfigDef = null;
```

Unclosed objects	
CWE ID	10031
Description	Closing objects inside a try clause, without a matching Close statement in the finally section may lead to resouce leaking in case of exception.
Likelihood of Exploit	Low
Common Consequences	In case of exception, open objects may not be closed. Make sure that the marked Close statement, which is within a try clause, has a matching Close statement in the finally clause.
Potential Mitigations	Add a matching close statement to the finally clause.
Applicable Platforms	All

Path 1:

Query Name - Unclosed_Objects

Severity -  Information

```
83. public DatabaseProvider() throws StartupException //databaseprovider.java
    ...
    141. testcon.close();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Unclosed_Objects**

Severity -  Information

```
145. public static String getRuntimeConfigDefsAsString() //planetruntimeconfig.java
    ...
    159. reader.close();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Unclosed_Objects**

Severity -  Information

```
264. public static void copyFile(File from, File to) throws IOException //utilities.java
    ...
    282. in.close();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Unclosed_Objects**

Severity -  Information

```
300. public static void copyInputToOutput( //utilities.java
    ...
    321. in.close();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Unclosed_Objects**

Severity -  Information

```
300. public static void copyInputToOutput( //utilities.java
    ...
    322. out.close();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Unclosed_Objects**

Severity -  Information

```
300. public static void copyInputToOutput( //utilities.java
    ...
    338. in.close();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Unclosed_Objects**

Severity -  Information

```
300. public static void copyInputToOutput( //utilities.java
    ...
    339. out.close();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Unclosed_Objects**

Severity -  Information

```
300. public static void copyInputToOutput( //utilities.java
    |
    ...
    348. in.close();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Unclosed_Objects**

Severity -  Information

```
300. public static void copyInputToOutput( //utilities.java
    |
    ...
    349. out.close();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Unclosed_Objects**

Severity -  Information

```
356. public static void copyInputToOutput( //utilities.java
    |
    ...
    370. in.close();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Catch Without General Exception Handling	
CWE ID	10009
Description	Not handling general exceptions may cause the program to crash.
Likelihood of Exploit	Low
Common Consequences	Program crashing.
Potential Mitigations	Handle the case of general exception.
Demonstrative Examples	The following code catches specific exception but not the general one. <div><pre>... try { File.Copy(path, path2); } catch(IOException e) { Console.WriteLine(e.Message); } ...</pre></div>

Applicable Platforms All

Path 1:

Query Name - **Catch_Without_General_Exception_Handling**

Severity -  Information

```
83. public DatabaseProvider() throws StartupException //databaseprovider.java
    |
    ...
    104. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Catch_Without_General_Exception_Handling**

Severity -  Information

```
83. public DatabaseProvider() throws StartupException //databaseprovider.java
    |
    ...
    125. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Catch_Without_General_Exception_Handling**

Severity -  **Information**

```
83. public DatabaseProvider() throws StartupException //databaseprovider.java
    |
    |...
    |139. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Catch_Without_General_Exception_Handling**

Severity -  **Information**

```
43. public GuicePlanetProvider() //guiceplanetprovider.java
    |
    |...
    |50. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Catch_Without_General_Exception_Handling**

Severity -  **Information**

```
66. public GuicePlanetProvider(String moduleClassname) //guiceplanetprovider.java
    |
    |...
    |72. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Catch_Without_General_Exception_Handling**

Severity -  **Information**

```
148. public String encode(String str) //multiplaneturlstrategy.java
    |
    |...
    |150. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Catch_Without_General_Exception_Handling**

Severity -  **Information**

```
162. public String decode(String str) //multiplaneturlstrategy.java
    |
    |...
    |164. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Catch_Without_General_Exception_Handling**

Severity -  **Information**

```
73. public Subscription fetchSubscription(String feedURL, Date lastModified) //romefeedfetcher.java
    |
    |...
    |115. try {
```

[\[Table of Contents\]](#) [\[Description\]](#)


Path 9:

Query Name - **Catch_Without_General_Exception_Handling**

Severity -  **Information**

```
72. protected JPAPersistenceStrategy() throws PlanetException //jpapersistencestrategy.java
    |
    |...
    |77. try {
```


[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:
Query Name - **Catch_Without_General_Exception_Handling**
Severity -  **Information**

```
72. protected JPAPersistenceStrategy() throws PlanetException //jpaperistencestrategy.java
    ...
    125. try {
```


[\[Table of Contents\]](#) [\[Description\]](#)

Pages Without Global Error Handler	
CWE ID	10026
Description	Not handling global errors in a page may lead to error message information leaks.
Likelihood of Exploit	Low
Common Consequences	In case an unhandled exception was thrown in a page that doe's not have global error handler the error message information may reveal sensitive information which may be used for a later attack.
Potential Mitigations	Define global error handler.
Applicable Platforms	All

Path 1:
Query Name - **Pages_Without_Global_Error_Handler**
Severity -  **Information**


```
1. <%-- //index.jsp
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:
Query Name - **Pages_Without_Global_Error_Handler**
Severity -  **Information**


```
1. <% response.sendRedirect("admin/ConfigForm.action"); %> //index.jsp
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:
Query Name - **Pages_Without_Global_Error_Handler**
Severity -  **Information**


```
1. <%-- //login.jsp
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:
Query Name - **Pages_Without_Global_Error_Handler**
Severity -  **Information**

```
1. <% //logout.jsp
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:
Query Name - **Pages_Without_Global_Error_Handler**
Severity -  **Information**

```
1. <%-- //mainmenu.jsp
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - Pages_Without_Global_Error_Handler

Severity -  Information

```
1. <%-- //configform.jsp
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - Pages_Without_Global_Error_Handler

Severity -  Information

```
1. <%-- //menu.jsp
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - Pages_Without_Global_Error_Handler

Severity -  Information

```
1. <%-- //planetform.jsp
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - Pages_Without_Global_Error_Handler

Severity -  Information

```
1. <%-- //planetgroupform.jsp
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - Pages_Without_Global_Error_Handler

Severity -  Information

```
1. <%-- //planetslist.jsp
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Incorrect Block Delimitation	
CWE ID	483
Description	In some languages, forgetting to explicitly delimit a block can result in a logic error that can, in turn, have security implications.
Likelihood of Exploit	Low
Common Consequences	This is a general logic error -- with all the potential consequences that this entails.
Potential Mitigations	Implementation: Always use explicit block delimitation and use static-analysis technologies to enforce this practice.
Demonstrative Examples	In this example, when the condition is true, the intention may be that both x and y run. if (condition==true) x; y;
Context Notes	In many languages, braces are optional for blocks, and -- in a case where braces are omitted -- it is possible to insert a logic error where a statement is thought to be in a block but is not. This is a common and well known reliability error.
Node Relationships	ChildOf - Code Quality (398)
Source Taxonomies	CLASP - Incorrect block delimitation
Applicable Platforms	All

Path 1:

Query Name - Single_Line_If_Statement

Severity -  Information

```
83. public DatabaseProvider() throws StartupException //databaseprovider.java
    |...
```

```
117. if (getJdbcUsername() != null) props.put("user", getJdbcUsername());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Single_Line_If_Statement**

Severity -  **Information**

```
83. public DatabaseProvider() throws StartupException //databaseprovider.java
    ...
117. if (getJdbcUsername() != null) props.put("user", getJdbcUsername());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Single_Line_If_Statement**

Severity -  **Information**

```
83. public DatabaseProvider() throws StartupException //databaseprovider.java
    ...
118. if (getJdbcPassword() != null) props.put("password", getJdbcPassword());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Single_Line_If_Statement**

Severity -  **Information**

```
83. public DatabaseProvider() throws StartupException //databaseprovider.java
    ...
118. if (getJdbcPassword() != null) props.put("password", getJdbcPassword());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Single_Line_If_Statement**

Severity -  **Information**

```
242. private FeedFetcherCache getRomeFetcherCache() //romefeedfetcher.java
    ...
266. if (!cacheDir.exists()) cacheDir.mkdirs();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Single_Line_If_Statement**

Severity -  **Information**

```
242. private FeedFetcherCache getRomeFetcherCache() //romefeedfetcher.java
    ...
266. if (!cacheDir.exists()) cacheDir.mkdirs();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Single_Line_If_Statement**

Severity -  **Information**

```
138. public List getTopSubscriptions( //japlanetmanagerimpl.java
    ...
145. if (offset != 0) q.setFirstResult(offset);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Single_Line_If_Statement**

Severity -  **Information**

```
138. public List getTopSubscriptions( //japlanetmanagerimpl.java
    ...
145. if (offset != 0) q.setFirstResult(offset);
```


Path 9:
Query Name - **Single_Line_If_Statement**
Severity -  **Information**

```
138. public List getTopSubscriptions( //japlanetmanagerimpl.java
    ...
146. if (len != -1) q.setMaxResults(len);
```

Path 10:
Query Name - **Single_Line_If_Statement**
Severity -  **Information**

```
138. public List getTopSubscriptions( //japlanetmanagerimpl.java
    ...
146. if (len != -1) q.setMaxResults(len);
```

Improper Error Handling	
CWE ID	390
Description	Sometimes an error is detected, and bad or no action is taken.
Likelihood of Exploit	Medium
Potential Mitigations	Implementation: Properly handle each exception. This is the recommended solution. Ensure that all exceptions are handled in such a way that you can be sure of the state of your system at any given moment. Subject the software to extensive testing to discover some of the possible instances of where/how errors or return values are not handled. Consider testing techniques such as ad hoc, equivalence partitioning, robustness and fault tolerance, mutation, and fuzzing.
Demonstrative Examples	<div>C Example:<pre>foo=malloc(sizeof(char); //the next line checks to see if malloc failed if (foo==0) { //We do nothing so we just ignore the error. }</pre></div> <div>C++ Example:<pre>while (DoSomething()) { try { /* perform main loop here */ } catch (Exception e) { /* do nothing, but catch so it'll compile... */ } }</pre></div> <div>Java Example:<pre>while (DoSomething()) { try { /* perform main loop here */ } catch (Exception e) { /* do nothing, but catch so it'll compile... */ } }</pre></div>
Context Notes	If a function returns an error, it is important to either fix the problem and try again, alert the user that an error has happened and let the program continue, or alert the user and close and cleanup the program.
Node Relationships	ChildOf - Error Conditions , Return Values , Status Codes (389)
Source Taxonomies	CLASP - Improper error handling
Applicable Platforms	All

Path 1:
Query Name - **Empty_Catch**
Severity -  **Information**

```
148. public String encode(String str) //multiplaneturlstrategy.java
    |
    |...
    |152. } catch (UnsupportedEncodingException ex) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Empty_Catch**

Severity -  **Information**

```
162. public String decode(String str) //multiplaneturlstrategy.java
    |
    |...
    |166. } catch (UnsupportedEncodingException ex) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Empty_Catch**

Severity -  **Information**

```
73. public Subscription fetchSubscription(String feedURL, Date lastModified) //romefeedfetcher.java
    |
    |...
    |123. } catch (MalformedURLException ex) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Empty_Catch**

Severity -  **Information**

```
316. protected static Properties loadPropertiesFromResourceName( //jpapersistencestrategy.java
    |
    |...
    |335. } catch (IOException ioe) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Empty_Catch**

Severity -  **Information**

```
89. public void setRows(String rows) //propertydef.java
    |
    |...
    |94. } catch(Exception e) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Empty_Catch**

Severity -  **Information**

```
106. public void setCols(String cols) //propertydef.java
    |
    |...
    |111. } catch(Exception e) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Empty_Catch**

Severity -  **Information**

```
266. private boolean isPlanet(String planetHandle) //multiplanetrequestmapper.java
    |
    |...
    |279. } catch(Exception ex) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Empty_Catch**

Severity -  **Information**

```
48. public PlanetGroupPageRequest(HttpServletRequest request) //planetgrouppagerequest.java
    |
    |...
    |71. } catch(NumberFormatException e) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:
Query Name - **Empty_Catch**
Severity -  **Information**

```
141. public Weblog(Element elem) throws Exception //technorati.java
    |
    |...
    |150. } catch (Exception ignored) {}
```


[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:
Query Name - **Empty_Catch**
Severity -  **Information**

```
141. public Weblog(Element elem) throws Exception //technorati.java
    |
    |...
    |153. } catch (Exception ignored) {}
```


[\[Table of Contents\]](#) [\[Description\]](#)

Bad Practices: Threads	
CWE ID	383
Description	Thread management in a Web application is forbidden in some circumstances and is always highly error prone.
Affected Resource	System Process
Potential Mitigations	For EBJ, use framework approaches for parallel execution, instead of using threads.
Context Notes	Thread management in a web application is forbidden by the J2EE standard in some circumstances and is always highly error prone. Managing threads is difficult and is likely to interfere in unpredictable ways with the behavior of the application container. Even without interfering with the container, thread management usually leads to bugs that are hard to detect and diagnose like deadlock, race conditions, and other synchronization errors.
Node Relationships	ChildOf - J2EE Time and State Issues (381) ChildOf - Weaknesses that Affect System Processes (634) ParentOf - Use of Singleton Pattern in a Non-thread-safe Manner (543)
Source Taxonomies	7 Pernicious Kingdoms - J2EE Bad Practices: Threads
Applicable Platforms	Java .Net

Path 1:
Query Name - **Threads_in_WebApp**
Severity -  **Information**

```
28. public class ContinuousWorkerThread extends WorkerThread //continuousworkerthread.java
    |
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:
Query Name - **Threads_in_WebApp**
Severity -  **Information**

```
30. public class WorkerThread extends Thread //workerthread.java
    |
```


[\[Table of Contents\]](#) [\[Description\]](#)

Failure to Account for Default Case in Switch	
CWE ID	478
Description	The failure to account for the default case in switch statements may lead to complex logical errors and may aid in other, unexpected security-related conditions.
Common Consequences	Undefined: Depending on the logical circumstances involved, any consequences may result: e.g., issues of

	confidentiality, authentication, authorization, availability, integrity, accountability, or non-repudiation.
Potential Mitigations	Implementation: Ensure that there are no unaccounted for cases, when adjusting flow or values based on the value of a given variable. In switch statements, this can be accomplished through the use of the default label.
Demonstrative Examples	<div>In general, a safe switch statement has this form:</div> <div><pre>switch (value) { case 'A': printf("A!\n"); break; case 'B': printf("B!\n"); break; default: printf("Neither A nor B\n"); }</pre></div> <div>This is because the assumption cannot be made that all possible cases are accounted for. A good practice is to reserve the default case for error handling.</div>
Context Notes	This flaw represents a common problem in software development, in which not all possible values for a variable are considered or handled by a given process. Because of this, further decisions are made based on poor information, and cascading failure results. This cascading failure may result in any number of security issues, and constitutes a significant failure in the system. In the case of switch style statements, the very simple act of creating a default case can mitigate this situation, if done correctly. Often however, the default cause is used simply to represent an assumed option, as opposed to working as a sanity check. This is poor practice and in some cases is as bad as omitting a default case entirely.
Node Relationships	ChildOf - Code Quality (398)
Source Taxonomies	CLASP - Failure to account for default case in switch
Applicable Platforms	C C++ Java .NET

Path 1:

Query Name - No_Default_Case


Severity -  Information

```
810. public static String stripInvalidTagCharacters(String tag) //utilities.java  
    ...  
    820. switch (c) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - No_Default_Case

Severity -  Information


```
888. public static String stripInvalidTagCharacters(String tag) //utilities.java  
    ...  
    898. switch (c) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Magic Numbers	
CWE ID	10017
Description	Using literals in the code makes the code less readable and maintainable.
Likelihood of Exploit	Low
Common Consequences	Code become less readable and maintainable.
Potential Mitigations	Define constants with meaningful names and use them instead.
Applicable Platforms	All

Path 1:

Query Name - Magic_Numbers

Severity -  Information

```
32. public class PropertyDef //propertydef.java
```

```
...
38. private int rows = 5;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Magic_Numbers**

Severity -  **Information**

```
32. public class PropertyDef //propertydef.java
...
39. private int cols = 25;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Magic_Numbers**

Severity -  **Information**

```
35. public class PlanetGroup implements Serializable, Comparable //planetgroup.java
...
44. private int maxPageEntries = 45;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Magic_Numbers**

Severity -  **Information**

```
35. public class PlanetGroup implements Serializable, Comparable //planetgroup.java
...
45. private int maxFeedEntries = 45;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Magic_Numbers**

Severity -  **Information**

```
43. public class HitCountQueue //hitcountqueue.java
...
50. private int sleepTime = 180000;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Magic_Numbers**

Severity -  **Information**

```
40. public class PingQueueTask extends RollerTaskWithLeasing //pingqueuetask.java
...
52. private int interval = 5;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Magic_Numbers**

Severity -  **Information**

```
40. public class PingQueueTask extends RollerTaskWithLeasing //pingqueuetask.java
...
55. private int leaseTime = 30;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Magic_Numbers**

Severity -  **Information**

```
57. public class ReferrerQueueManagerImpl implements ReferrerQueueManager //referrerqueuemanagerimpl.java
```



```
...
65. private int sleepTime = 10000;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Magic_Numbers**

Severity -  **Information**

```
28. public class ContinuousWorkerThread extends WorkerThread //continuousworkerthread.java
...
33. long sleepTime = 10000;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Magic_Numbers**

Severity -  **Information**

```
33. public class ResetHitCountsTask extends RollerTaskWithLeasing //resethitcountstask.java
...
45. private int interval = 1440;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Confusing naming	
CWE ID	10032
Description	Naming a method and field with the same name, may be confusing to developers
Likelihood of Exploit	Low
Common Consequences	Difficulty to maintain the code
Potential Mitigations	Renaming methods and fields
Applicable Platforms	All

Path 1:

Query Name - **Confusing_Naming**

Severity -  **Information**

```
39. public class Collection //collection.java
...
46. private List accepts = new ArrayList(); // of Strings
```

[\[Table of Contents\]](#) [\[Description\]](#)

Leftover Debug Code	
CWE ID	489
Description	Debug code can create unintended entry points in an application.
Common Consequences	The severity of the exposed debug application will depend on the particular instance. At the least, it will give an attacker sensitive information about the settings and mechanics of web applications on the server. At worst, as is often the case, the debug application will allow an attacker complete control over the web application and server, as well as confidential information that either of these access.
Demonstrative Examples	Debug code can be used to bypass authentication. For example, suppose an application has a login script that receives a username and a password. Assume also that a third, optional, parameter, called "debug", is interpreted by the script as requesting a switch to debug mode, and that when this parameter is given the username and password are not checked. In such a case, it is very simple to bypass the authentication process if the special behavior of the application regarding the debug parameter is known. In a case where the form is:

```
<FORM ACTION="/authenticate_login.cgi">
  <INPUT TYPE=TEXT name=username>
  <INPUT TYPE=PASSWORD name=password>
  <INPUT TYPE=SUBMIT>
</FORM>
```

Then a conforming link will look like:

```
http://TARGET/authenticate_login.cgi?username=...&password=...
```

An attacker can change this to:

```
http://TARGET/authenticate_login.cgi?username=&password=&debug=1
```

Which will grant the attacker access to the site, bypassing the authentication process.

Context Notes	<p>A common development practice is to add "back door" code specifically designed for debugging or testing purposes that is not intended to be shipped or deployed with the application. In web-based applications, debug code is used to test and modify web application properties, configuration information, and functions. If a debug application is left on a production server, an attacker may be able to use it to perform these tasks. When this sort of debug code is left in the application, the application is open to unintended modes of interaction. These back door entry points create security risks because they are not considered during design or testing and fall outside of the expected operating conditions of the application.</p> <p>While it is possible to leave debug code in an application in any language, in J2EE a main method may be a good indicator that debug code has been left in the application, although there may not be any direct security impact.</p>
Node Relationships	ChildOf - Encapsulation (485) ChildOf - Weaknesses Examined by SAMATE (630)
Source Taxonomies	7 Pernicious Kingdoms - Leftover Debug Code
Applicable Platforms	All

Path 1:

Query Name - **Leftover_Debug_Code**

Severity -  **Information**

```
120. public static void main(String[] args) throws Exception //generateplanettask.java
    {
        ;
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Leftover_Debug_Code**

Severity -  **Information**

```
55. public static void main(String[] args) throws Exception //refreshplanettask.java
    {
        ;
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Leftover_Debug_Code**

Severity -  **Information**

```
137. public static void main(String[] args) throws Exception //technoratirankingstask.java
    {
        ;
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Leftover_Debug_Code**

Severity -  **Information**

```
33. public static void main(String[] args) throws Exception //taskrunner.java
    {
        ;
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Leftover_Debug_Code**

Severity -  **Information**

```
32. public static void main(String[] args) //diskfeedinfocachetest.java
    {
        ;
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Leftover_Debug_Code**

Severity -  **Information**

```
153. public static void main(String[] args) throws Exception //pingqueuetask.java
    {
        // ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:
Query Name - **Leftover_Debug_Code**
Severity -  **Information**

```
144. public static void main(String[] args) throws Exception //resethitcountstask.java
    {
        // ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:
Query Name - **Leftover_Debug_Code**
Severity -  **Information**

```
196. public static void main(String[] args) throws Exception //scheduledentriestask.java
    {
        // ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:
Query Name - **Leftover_Debug_Code**
Severity -  **Information**

```
30. public static void main(String[] args) throws Exception //taskrunner.java
    {
        // ...
    }
}
```


[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:
Query Name - **Leftover_Debug_Code**
Severity -  **Information**

```
144. public static void main(String[] args) throws Exception //turnoverrefererstask.java
    {
        // ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Missing Catch Block	
CWE ID	600
Description	If a Servlet fails to catch all exceptions, it may reveal debugging information that will help an adversary form a plan of attack.
Demonstrative Examples	<div>In the following method a DNS lookup failure will cause the Servlet to throw an exception.<pre>protected void doPost (HttpServletRequest req, HttpServletResponse res) throws IOException { String ip = req.getRemoteAddr(); InetAddress addr = InetAddress.getByName(ip); ... out.println("hello " + addr.getHostName()); }</pre></div>
Context Notes	When a Servlet throws an exception, the default error response the Servlet container sends back to the user typically includes debugging information. This information is of great value to an attacker. For example, a stack trace might show the attacker a malformed SQL query string, the type of database being used, and the version of the application container. This information enables the attacker to target known vulnerabilities in these components.
Node Relationships	ChildOf - Error Handling (388) ResultsIn - Error Message Information Leaks (209) Peer - Improper Error Handling (390)

Path 1:
Query Name - **Missing_Catch_Block**
Severity -  **Information**

```
65. public void doGet(HttpServletRequest request, HttpServletResponse response) //feedservlet.java
    {
        // ...
    }
}
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - **Missing_Catch_Block**

Severity -  **Information**

```
65. public void doGet(HttpServletRequest request, HttpServletResponse response) //homepageservlet.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Missing_Catch_Block**

Severity -  **Information**

```
65. public void doGet(HttpServletRequest request, HttpServletResponse response) //opmlservlet.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Missing_Catch_Block**

Severity -  **Information**

```
65. public void doGet(HttpServletRequest request, HttpServletResponse response) //pageservlet.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Missing_Catch_Block**

Severity -  **Information**

```
57. public void doGet(HttpServletRequest request, HttpServletResponse response) //commentauthenticatorservlet.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Missing_Catch_Block**

Severity -  **Information**

```
145. public void doGet(HttpServletRequest request, HttpServletResponse response) //commentservlet.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Missing_Catch_Block**

Severity -  **Information**

```
156. public void doPost(HttpServletRequest request, HttpServletResponse response) //commentservlet.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Missing_Catch_Block**

Severity -  **Information**

```
84. public void doGet(HttpServletRequest request, HttpServletResponse response) //feedservlet.java
    {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Missing_Catch_Block**

Severity -  **Information**

```
114. public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException //pageservlet.java
    {
```

Path 10:

Query Name - **Missing_Catch_Block**

Severity -  **Information**

```
491. public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    IOException //pageservlet.java
    {
        ...
    }
```

Overly-Broad Catch Block	
CWE ID	396
Description	Catching overly broad exceptions promotes complex error handling code that is more likely to contain security vulnerabilities.
Demonstrative Examples	<div>The following code excerpt handles three types of exceptions in an identical fashion.</div> <div><pre>try { doExchange(); } catch (IOException e) { logger.error("doExchange failed", e); } catch (InvocationTargetException e) { logger.error("doExchange failed", e); } catch (SQLException e) { logger.error("doExchange failed", e); }</pre></div> <div>At first blush, it may seem preferable to deal with these exceptions in a single catch block, as follows: try { doExchange(); } catch (Exception e) { logger.error("doExchange failed", e); } However, if doExchange() is modified to throw a new type of exception that should be handled in some different kind of way, the broad catch block will prevent the compiler from pointing out the situation. Further, the new catch block will now also handle exceptions derived from RuntimeException such as ClassCastException, and NullPointerException, which is not the programmer's intent.</div>
Context Notes	Multiple catch blocks can get ugly and repetitive, but "condensing" catch blocks by catching a high-level class like Exception can obscure exceptions that deserve special treatment or that should not be caught at this point in the program. Catching an overly broad exception essentially defeats the purpose of Java's typed exceptions, and can become particularly dangerous if the program grows and begins to throw new types of exceptions. The new exception types will not receive any attention.
Node Relationships	ChildOf - Error Conditions , Return Values , Status Codes (389)
Source Taxonomies	7 Pernicious Kingdoms - Overly-Broad Catch Block
Applicable Platforms	C C++ Java .NET

Path 1:

Query Name - **Overly_Broad_Catch**

Severity -  **Information**

```
75. public static final void bootstrap() throws BootstrapException //planetfactory.java
    {
        ...
        89. } catch (Exception ex) {
            ...
        }
```

Path 2:

Query Name - **Overly_Broad_Catch**

Severity -  **Information**

```
73. public Subscription fetchSubscription(String feedURL, Date lastModified) //romefeedfetcher.java
    {
        ...
        88. } catch (Exception ex) {
            ...
        }
```

Path 3:

Query Name - **Overly_Broad_Catch**

Severity -  **Information**

```
242. private FeedFetcherCache getRomeFetcherCache() //romefeedfetcher.java
    |
    |...
    |267. } catch (Exception e) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Overly_Broad_Catch**

Severity -  **Information**

```
125. public void initialize() throws InitializationException //japropertiesmanagerimpl.java
    |
    |...
    |135. } catch (Exception e) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Overly_Broad_Catch**

Severity -  **Information**

```
34. public class PlanetConfig //planetconfig.java
    |
    |...
    |124. } catch (Exception e) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Overly_Broad_Catch**

Severity -  **Information**

```
60. public static String getProperty(String name) //planetruntimeconfig.java
    |
    |...
    |67. } catch(Exception e) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Overly_Broad_Catch**

Severity -  **Information**

```
95. public static int getIntProperty(String name) //planetruntimeconfig.java
    |
    |...
    |106. } catch(Exception e) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Overly_Broad_Catch**

Severity -  **Information**

```
114. public static RuntimeConfigDefs getRuntimeConfigDefs() //planetruntimeconfig.java
    |
    |...
    |126. } catch(Exception e) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Overly_Broad_Catch**

Severity -  **Information**

```
145. public static String getRuntimeConfigDefsAsString() //planetruntimeconfig.java
    |
    |...
    |162. } catch(Exception e) {
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Overly_Broad_Catch**

```
89. public void setRows(String rows) //propertydef.java
    |
    |...
    |94. } catch(Exception e) {
```

[\[Table of Contents\]](#)

[\[Description\]](#)

Overly-Broad Throws Declaration	
CWE ID	397
Description	Throwing overly broad exceptions promotes complex error handling code that is more likely to contain security vulnerabilities.
Demonstrative Examples	<div>The following method throws three types of exceptions.<pre>public void doExchange() throws IOException, InvocationTargetException, SQLException { ... }</pre></div> <div>While it might seem tidier to write <code>public void doExchange() throws Exception { ... }</code> doing so hampers the caller's ability to understand and handle the exceptions that occur. Further, if a later revision of <code>doExchange()</code> introduces a new type of exception that should be treated differently than previous exceptions, there is no easy way to enforce this requirement.</div>
Context Notes	Declaring a method to throw <code>Exception</code> or <code>Throwable</code> makes it difficult for callers to do good error handling and error recovery. Java's exception mechanism is set up to make it easy for callers to anticipate what can go wrong and write code to handle each specific exceptional circumstance. Declaring that a method throws a generic form of exception defeats this system.
Node Relationships	ChildOf - Error Conditions, Return Values, Status Codes (389)
Source Taxonomies	7 Pernicious Kingdoms - Overly-Broad Throws Declaration
Applicable Platforms	C C++ Java .NET

Path 1:

Query Name - Overly_Broad_Throws

Severity

-  Information

```
120. public static void main(String[] args) throws Exception //generateplanettask.java
    |
```

[\[Table of Contents\]](#)

[\[Description\]](#)

Path 2:

Query Name - Overly_Broad_Throws

Severity

-  Information

```
35. public void initialize() throws Exception //planettask.java
    |
```

[\[Table of Contents\]](#)

[\[Description\]](#)

Path 3:

Query Name - Overly_Broad_Throws

Severity

-  Information

```
55. public static void main(String[] args) throws Exception //refreshplanettask.java
    |
```

[\[Table of Contents\]](#)

[\[Description\]](#)

Path 4:

Query Name - Overly_Broad_Throws

Severity

-  Information

```
52. public Subscription getSubscription(String feedUrl) throws Exception //staticplanetmodel.java
    |
```

Path 5:
Query Name - **Overly_Broad_Throws**
Severity -  **Information**


```
57. public List getFeedEntries(String feedUrl, int maxEntries) throws Exception //staticplanetmodel.java
    {
        // ...
    }
}
```

Path 6:
Query Name - **Overly_Broad_Throws**
Severity -  **Information**


```
137. public static void main(String[] args) throws Exception //technoratirankingtask.java
    {
        // ...
    }
}
```

Path 7:
Query Name - **Overly_Broad_Throws**
Severity -  **Information**


```
54. public void prepare() throws Exception //configform.java
    {
        // ...
    }
}
```

Path 8:
Query Name - **Overly_Broad_Throws**
Severity -  **Information**

```
55. public void prepare() throws Exception //planetform.java
    {
        // ...
    }
}
```

Path 9:
Query Name - **Overly_Broad_Throws**
Severity -  **Information**

```
59. public void prepare() throws Exception //planetgroupform.java
    {
        // ...
    }
}
```

Path 10:
Query Name - **Overly_Broad_Throws**
Severity -  **Information**

```
54. public void prepare() throws Exception //planetsubscriptionform.java
    {
        // ...
    }
}
```

Use of system output stream	
CWE ID	10033
Description	A dedicated logging class should be used, rather than using default output stream, to ease the monitoring process
Likelihood of Exploit	Low
Common Consequences	Difficulty to maintain the application and to follow log files
Potential Mitigations	Using dedicated logging class
Applicable Platforms	All

Path 1:

Query Name - Use_of_System_Output_Stream

Severity -  Information

```
45. public void printStackTrace() //fetcherexception.java
    |
    |...
    |48. System.out.println(super.getMessage());
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - Use_of_System_Output_Stream

Severity -  Information

```
45. public void printStackTrace() //fetcherexception.java
    |
    |...
    |50. System.out.println("--- ROOT CAUSE ---");
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - Use_of_System_Output_Stream

Severity -  Information

```
33. public static void main(String[] args) throws Exception //taskrunner.java
    |
    |...
    |35. System.err.println("USAGE: java -cp roller-planet.jar TaskRunner WEBAPPPDIR JARSDIR CLASSNAME");
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - Use_of_System_Output_Stream

Severity -  Information

```
33. public static void main(String[] args) throws Exception //taskrunner.java
    |
    |...
    |36. System.err.println("WEBAPPPDIR: The directory path to the web application ");
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - Use_of_System_Output_Stream

Severity -  Information

```
33. public static void main(String[] args) throws Exception //taskrunner.java
    |
    |...
    |37. System.err.println(" (e.g. $CATALINA_HOME/webapps/roller)");
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - Use_of_System_Output_Stream

Severity -  Information

```
33. public static void main(String[] args) throws Exception //taskrunner.java
    |
    |...
    |38. System.err.println("JARSDIR: The directory path to the additional jars ");
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - Use_of_System_Output_Stream

Severity -  Information

```
33. public static void main(String[] args) throws Exception //taskrunner.java
    |
    |...
    |39. System.err.println(" directory (e.g. $CATALINA_HOME/common/lib)");
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - Use_of_System_Output_Stream

Severity -  Information

```
33. public static void main(String[] args) throws Exception //taskrunner.java
    ...
    40. System.err.println("CLASSNAME: The name of the class to be executed by TaskRunner ");
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - Use_of_System_Output_Stream

Severity -  Information

```
33. public static void main(String[] args) throws Exception //taskrunner.java
    ...
    46. System.out.println("WEBAPPPDIR = " + webappDir);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - Use_of_System_Output_Stream

Severity -  Information

```
33. public static void main(String[] args) throws Exception //taskrunner.java
    ...
    47. System.out.println("JARSDIR = " + jarsDir);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Visible Fields	
CWE ID	10003
Description	Visible fields make code change more difficult.
Likelihood of Exploit	Low
Common Consequences	Changing code is more difficult.
Potential Mitigations	Make the field private or internal and expose it using an externally visible property.
Demonstrative Examples	<div>The following code declares public and protected fields. C# Example: <pre>... public class MyClass { public int field1; protected string field2; } ...</pre></div>
Applicable Platforms	All

Path 1:

Query Name - Visible_Fields

Severity -  Information

```
51. public class JPAPersistenceStrategy //jpaperistencestrategy.java
    ...
    65. protected EntityManagerFactory emf;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 2:

Query Name - Visible_Fields

Severity -  Information

```
48. public class JPAPlanetManagerImpl extends AbstractManagerImpl implements PlanetManager //japplanetmanagerimpl.java
    ...
    55. protected Map lastUpdatedByGroup = new HashMap();
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 3:

Query Name - **Visible_Fields**

Severity -  **Information**

```
49. public class PlanetRequest extends ParsedRequest //planetrequest.java
    ...
    55. protected String pathInfo = null;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 4:

Query Name - **Visible_Fields**

Severity -  **Information**

```
37. public class DiskFeedInfoCache implements FeedFetcherCache //diskfeedinfocache.java
    ...
    42. protected String cachePath = null;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 5:

Query Name - **Visible_Fields**

Severity -  **Information**

```
36. public class StandaloneWebappClassLoader extends URLClassLoader //standalonewebappclassloader.java
    37. public static String FS = File.separator;
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 6:

Query Name - **Visible_Fields**

Severity -  **Information**

```
33. public class EntryFunctionalTests extends TestCase //entryfunctionaltests.java
    ...
    35. public static Log log = LogFactory.getLog(EntryFunctionalTests.class);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 7:

Query Name - **Visible_Fields**

Severity -  **Information**

```
30. public class PlanetBasicTests extends TestCase //planetbasictests.java
    ...
    32. public static Log log = LogFactory.getLog(PlanetBasicTests.class);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 8:

Query Name - **Visible_Fields**

Severity -  **Information**

```
30. public class PlanetFunctionalTests extends TestCase //planetfunctionaltests.java
    ...
    32. public static Log log = LogFactory.getLog(PlanetFunctionalTests.class);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 9:

Query Name - **Visible_Fields**

Severity -  **Information**

```
36. public class PropertiesTest extends TestCase //propertiestest.java
    ...
    38. public static Log log = LogFactory.getLog(PropertiesTest.class);
```

[\[Table of Contents\]](#) [\[Description\]](#)

Path 10:

Query Name - **Visible_Fields**

Severity -  **Information**

```
33. public class RomeFeedFetcherTest extends TestCase //romefeedfetchertest.java
    |
    |...
35. public static Log log = LoggerFactory.getLog(RomeFeedFetcherTest.class);
```

[\[Table of Contents\]](#) [\[Description\]](#)
