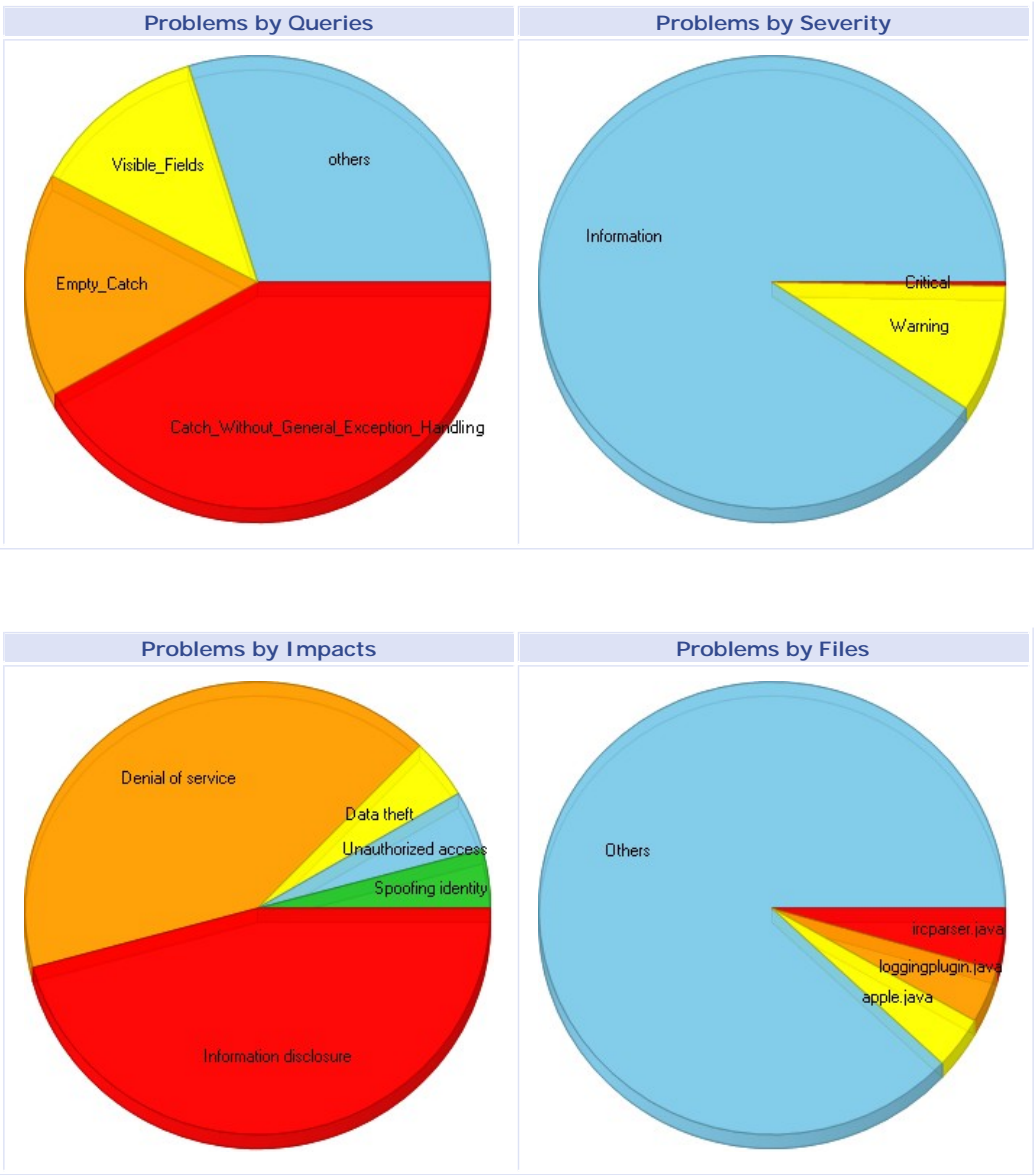

















Project Name: c:\documents and settings\sate\desktop\sate\_2009\dmdirc  
Load Time: 02/09/2009 18:04:00  
CxSuite Version: 2.7.5.0



## Summary

Query	Group	Problems found	Severity
<a href="#">_Command Injection</a>	Java High Risk	2	
<a href="#">_DoS by Unreleased Resources</a>	Java Low Visibility	5	
<a href="#">_Improper Exception Handling</a>	Java Low Visibility	60	
<a href="#">_Weak Cryptographic Algorithm</a>	Java Low Visibility	1	
<a href="#">_Unclosed Objects</a>	Java Best Coding Practice	23	
<a href="#">_Catch Without General Exception Handling</a>	Java Best Coding Practice	303	
<a href="#">_Single Line If Statement</a>	Java Best Coding Practice	16	
<a href="#">_Empty Catch</a>	Java Best Coding Practice	116	
<a href="#">_No Default Case</a>	Java Best Coding Practice	6	
<a href="#">_Call to Thread run</a>	Java Best Coding Practice	1	
<a href="#">_Erroneous String Compare</a>	Java Best Coding Practice	2	
<a href="#">_Magic Numbers</a>	Java Best Coding Practice	22	
<a href="#">_Confusing Naming</a>	Java Best Coding Practice	7	
<a href="#">_Overly Broad Catch</a>	Java Best Coding Practice	16	
<a href="#">_Overly Broad Throws</a>	Java Best Coding Practice	55	
<a href="#">_Visible Fields</a>	Java Best Coding Practice	90	

## Top 10 Files

File name	Problems found
client-0.6.3m1\client-0.6.3m1\src\com\dmdirc\parser\irc\ircparser.java	32
client-0.6.3m1\client-0.6.3m1\src\com\dmdirc\addons\logging\loggingplugin.java	28
client-0.6.3m1\client-0.6.3m1\src\com\dmdirc\addons\ui_swing\apple.java	27
client-0.6.3m1\client-0.6.3m1\src\com\dmdirc\plugins\plugininfo.java	20
client-0.6.3m1\client-0.6.3m1\src\com\dmdirc\addons\dcc\dcc.java	20
client-0.6.3m1\client-0.6.3m1\src\net\miginfocom\base64.java	19
client-0.6.3m1\client-0.6.3m1\src\com\dmdirc\addons\ui_swing\components\frames\textframe.java	18
client-0.6.3m1\client-0.6.3m1\src\com\dmdirc\installer\linuxinstaller.java	18
client-0.6.3m1\client-0.6.3m1\src\com\dmdirc\addons\dcc\dccsend.java	17
client-0.6.3m1\client-0.6.3m1\src\com\dmdirc\parser\irc\logging.java	14

## Command Injection

<b>CWE ID</b>	77
<b>Description</b>	Command injection problems are a subset of injection problem, in which the process can be tricked into calling external processes of an attackers choice through the injection of command syntax into the data plane.
<b>Likelihood of Exploit</b>	Very High
<b>Weakness Ordinality</b>	Primary (Weakness exists independent of other weaknesses)
<b>Causal Nature</b>	Explicit (This is an explicit weakness resulting from behavior of the developer)
<b>Common Consequences</b>	Access control: Command injection allows for the execution of arbitrary commands and code by the attacker.
<b>Potential Mitigations</b>	Design: If at all possible, use library calls rather than external processes to recreate the desired functionality Implementation: Ensure that all external commands called from the program are statically created, or -- if they must take input from a user -- that the input and final line generated are vigorously white-list checked. Run time: Run time policy enforcement may be used in a white-list fashion to prevent use of any non-sanctioned commands. Assign permissions to the software system that prevents the user from accessing/opening privileged files.
<b>Demonstrative Examples</b>	The following simple program accepts a filename as a command line argument and displays the contents of the file back to the user. The program is installed setuid root because it is intended for use as a learning tool to allow system administrators in-training to inspect privileged system files without giving them the ability to modify them or damage the system.

```
int main(char* argc, char** argv) {  
    char cmd[CMD_MAX] = "/usr/bin/cat ";  
    strcat(cmd, argv[1]);  
    system(cmd);  
}
```

Because the program runs with root privileges, the call to `system()` also executes with root privileges. If a user specifies a standard filename, the call works as expected. However, if an attacker passes a string of the form `";rm -rf /"`, then the call to `system()` fails to execute `cat` due to a lack of arguments and then plows on to recursively delete the contents of the root partition.

The following code is from an administrative web application designed allow users to kick off a backup of an Oracle database using a batch-file wrapper around the `rman` utility and then run a `cleanup.bat` script to delete some temporary files. The script `rmanDB.bat` accepts a single command line parameter, which specifies what type of backup to perform. Because access to the database is restricted, the application runs the backup as a privileged user.

```
...  
String btype = request.getParameter("backuptype");  
String cmd = new String("cmd.exe /K  
\\c:\\util\\rmanDB.bat " + btype + "&&c:\\util\\cleanup.bat")  
System.Runtime.getRuntime().exec(cmd);  
...
```

The problem here is that the program does not do any validation on the `backuptype` parameter read from the user. Typically the `Runtime.exec()` function will not execute multiple commands, but in this case the program first runs the `cmd.exe` shell in order to run multiple commands with a single call to `Runtime.exec()`. Once the shell is invoked, it will happily execute multiple commands separated by two ampersands. If an attacker passes a string of the form `"& del c:\\dbms\\*. *"`, then the application will execute this command along with the others specified by the program. Because of the nature of the application, it runs with the privileges necessary to interact with the database, which means whatever command the attacker injects will run with those privileges as well.

The following code from a system utility uses the system property `APPHOME` to determine the directory in which it is installed and then executes an initialization script based on a relative path from the specified directory.

```
...  
String home = System.getProperty("APPHOME");  
String cmd = home + INITCMD;  
java.lang.Runtime.getRuntime().exec(cmd);  
...
```

The code above allows an attacker to execute arbitrary commands with the elevated privilege of the application by modifying the system property `APPHOME` to point to a different path containing a malicious version of `INITCMD`. Because the program does not validate the value read from the environment, if an attacker can control the value of the system property `APPHOME`, then they can fool the application into running malicious code and take control of the system.

The following code is from a web application that allows users access to an interface through which they can update their password on the system. Part of the process for updating passwords in certain network environments is to run a `make` command in the `/var/yp` directory, the code for which is shown below.

```
...  
System.Runtime.getRuntime().exec("make");
```

The problem here is that the program does not specify an absolute path for make and fails to clean its environment prior to executing the call to Runtime.exec(). If an attacker can modify the \$PATH variable to point to a malicious binary called make and cause the program to be executed in their environment, then the malicious binary will be loaded instead of the one intended. Because of the nature of the application, it runs with the privileges necessary to perform system operations, which means the attacker's make will now be run with these privileges, possibly giving the attacker complete control of the system.

The following code is a wrapper around the UNIX command cat which prints the contents of a file to standard out. It is also injectable:

C Example:

```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv) {
    char cat[] = "cat ";
    char *command;
    size_t commandLength;

    commandLength = strlen(cat) + strlen(argv[1]) + 1;
    command = (char *) malloc(commandLength);
    strncpy(command, cat, commandLength);
    strncat(command, argv[1], (commandLength - strlen(cat)) );

    system(command);
    return (0);
}
```

Used normally, the output is simply the contents of the file requested: \$ ./catWrapper Story.txt When last we left our heroes... However, if we add a semicolon and another command to the end of this line, the command is executed by catWrapper with no complaint: \$ ./catWrapper Story.txt; ls When last we left our heroes... Story.txt

doubFree.c nullpointer.c unstosig.c www\* a.out\* format.c strlen.c useFree\* catWrapper\* misnull.c strlen.c useFree.c commandinjection.c nodefault.c trunc.c writeWhatWhere.c If catWrapper had been set to have a higher privilege level than the standard user, arbitrary commands could be executed with that higher privilege.

Context Notes	Command injection is a common problem with wrapper programs. Often, parts of the command to be run are controllable by the end user. If a malicious user injects a character (such as a semi-colon) that delimits the end of one command and the beginning of another, he may then be able to insert an entirely new and unrelated command to do whatever he pleases. The most effective way to deter such an attack is to ensure that the input provided by the user adheres to strict rules as to what characters are acceptable. As always, white-list style checking is far preferable to black-list style checking.
	Dynamically generating operating system commands that include user input as parameters can lead to command injection attacks. An attacker can insert operating system commands or modifiers in the user input that can cause the request to behave in an unsafe manner. Such vulnerabilities can be very dangerous and lead to data and system compromise. If no validation of the parameter to the exec command exists, an attacker can execute any command on the system the application has the privilege to access.
	Command injection vulnerabilities take two forms: * An attacker can change the command that the program executes: the attacker explicitly controls what the command is. * An attacker can change the environment in which the command executes: the attacker implicitly controls what the command means. In this case we are primarily concerned with the first scenario, in which an attacker explicitly controls the command that is executed. Command injection vulnerabilities of this type occur when: 1. Data enters the application from an untrusted source. 2. The data is part of a string that is executed as a command by the application. 3. By executing the command, the application gives an attacker a privilege or capability that the attacker would not otherwise have.
References	G. Hoglund and G. McGraw. "Exploiting Software: How to Break Code". Addison-Wesley. February 2004.
Node Relationships	ChildOf - <a href="#">Injection</a> (74) ChildOf - <a href="#">Weaknesses in OWASP Top Ten</a> (629) ParentOf - <a href="#">Executable Regular Expression Error</a> (624) ParentOf - <a href="#">OS Command Injection</a> (78)
Source Taxonomies	7 Pernicious Kingdoms - Command Injection CLASP - Command injection
Applicable Platforms	All

Path 1:  
Query Name - Command\_Injection  
Severity - Critical

```
73. public void hyperlinkUpdate(final HyperlinkEvent e) //aboutpanel.java
    ...
75. URLHandler.getURLHandler().launchApp(e.getURL());

118. public void launchApp(final URI uri) //urlhandler.java
    ...
137. final String command = config.getOption("protocol", uri.getScheme());
    ...
154. Main.getUI().getStatusBar().setMessage("Opening: " + uri.toString());
155. execApp(substituteParams(uri, command));
```

167. public String substituteParams(final URI url, final String command) //urlhandler.java

```
168. final String userInfo = url.getUserInfo();
...
209. password = userInfo.substring(pos + 1);
...
221. newCommand = newCommand.replaceAll("\\$password", password);
...
223. return newCommand;
```

118. public void launchApp(final URI uri) //urlhandler.java

```
...
155. execApp(substituteParams(uri, command));
```

231. private void execApp(final String command) //urlhandler.java

```
...
233. Runtime.getRuntime().exec(parseArguments(command));
```

248. protected static String[] parseArguments(final String command) //urlhandler.java

```
...
266. builder.append(word.substring(1));
...
278. return args.toArray(new String[args.size()]);
```

231. private void execApp(final String command) //urlhandler.java

```
...
233. Runtime.getRuntime().exec(parseArguments(command));
```

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

## Path 2:

Query Name - **Command\_Injection**

Severity -  **Critical**

93. public void hyperlinkUpdate(final HyperlinkEvent e) //creditspanel.java

```
...
95. URLHandler.getURLHandler().launchApp(e.getURL());
```

118. public void launchApp(final URI uri) //urlhandler.java

```
...
137. final String command = config.getOption("protocol", uri.getScheme());
...
154. Main.getUI().getStatusBar().setMessage("Opening: " + uri.toString());
155. execApp(substituteParams(uri, command));
```

167. public String substituteParams(final URI url, final String command) //urlhandler.java

```
168. final String userInfo = url.getUserInfo();
...
209. password = userInfo.substring(pos + 1);
...
221. newCommand = newCommand.replaceAll("\\$password", password);
...
223. return newCommand;
```

118. public void launchApp(final URI uri) //urlhandler.java

```
...
155. execApp(substituteParams(uri, command));
```

231. private void execApp(final String command) //urlhandler.java

```
...
233. Runtime.getRuntime().exec(parseArguments(command));
```

248. protected static String[] parseArguments(final String command) //urlhandler.java

```
...
266. builder.append(word.substring(1));
...
278. return args.toArray(new String[args.size()]);
```

231. private void execApp(final String command) //urlhandler.java

```
...
233. Runtime.getRuntime().exec(parseArguments(command));
```

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;
}
```

Observed Examples	CVE-1999-1127 - Does not shut down named pipe connections if malformed data is sent. 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. 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 - <a href="#">Resource Management Errors</a> (399) Peer - <a href="#">Memory Leak</a> (401) Peer - <a href="#">Asymmetric Resource Consumption (Amplification)</a> (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

```
35. public class DCCChat extends DCC //dccchat.java
    |
    |...
    |41. private BufferedReader in;
```

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

### Path 2:

Query Name - DoS\_by\_Unreleased\_Resources

Severity -  Warning

```
77. public void run() //streamreader.java
    |
    |78. BufferedReader reader = new BufferedReader(new InputStreamReader(stream));
```

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

### Path 3:

Query Name - DoS\_by\_Unreleased\_Resources

Severity -  Warning

```
777. protected boolean showHistory(final InputWindow target) //loggingplugin.java
    |
    |...
```




```
| 802. ReverseFileReader reader;
```

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

Path 4:  
Query Name - **DoS\_by\_Unreleased\_Resources**  
Severity -  **Warning**

```
39. public class ReverseFileReader //reversefilereader.java
    {
```

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

Path 5:  
Query Name - **DoS\_by\_Unreleased\_Resources**  
Severity -  **Warning**

```
60. public class IRCParse implements Runnable //ircparser.java
    {
    ...
    220. private BufferedReader in;
```

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

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 - <a href="#">API Abuse</a> (227) Peer - <a href="#">Error Conditions, Return Values, Status Codes</a> (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**


```
287. public static String getConfigDir() //main.java
    {
    ...
    289. final String fs = System.getProperty("file.separator");
```

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

Path 2:  
Query Name - **Improper\_Exception\_Handling**  
Severity -  **Warning**

```
287. public static String getConfigDir() //main.java
    {
    ...
    290. final String osName = System.getProperty("os.name");
```

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

Path 3:  
Query Name - **Improper\_Exception\_Handling**  
Severity -  **Warning**

```
287. public static String getConfigDir() //main.java
    |
    |...
    |292. configdir = System.getProperty("user.home") + fs + "Library"
```

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

---

#### Path 4:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
287. public static String getConfigDir() //main.java
    |
    |...
    |296. configdir = System.getProperty("user.home") + fs + "DMDirc" + fs;
```

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

---

#### Path 5:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
287. public static String getConfigDir() //main.java
    |
    |301. configdir = System.getProperty("user.home") + fs + ".DMDirc" + fs;
```

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

---

#### Path 6:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
406. public static String getDirectory() //actionmanager.java
    |
    |407. return Main.getConfigDir() + "actions" + System.getProperty("file.separator");
```

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

---

#### Path 7:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
126. public void sendLine(final String line) //dccchat.java
    |
    |...
    |128. out.printf("%s\r\n", line);
```

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

---

#### Path 8:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
456. public void domainUpdated() //dccplugin.java
    |
    |460. Main.getConfigDir() + "downloads" + System.getProperty("file.separator");
```

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

---

#### Path 9:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
105. protected String getIdentResponse(final String input, final ConfigManager config) //identclient.java
    |
    |...
    |133. final String osName = System.getProperty("os.name").toLowerCase();
```

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

---

#### Path 10:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
105. protected String getIdentResponse(final String input, final ConfigManager config) //identclient.java
    |
```

```
...
162. username = System.getProperty("user.name");
```

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

---

#### Path 11:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
112. public void showConfig(final PreferencesManager manager) //identdplugin.java
...
126. "" + System.getProperty("user.name") + "');
```

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

---

#### Path 12:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
112. public void showConfig(final PreferencesManager manager) //identdplugin.java
...
131. "" + System.getProperty("user.name") + "');
```

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

---

#### Path 13:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
104. public void domainUpdated() //loggingplugin.java
...
107. + System.getProperty("file.separator");
```

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

---

#### Path 14:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
661. protected void addNetworkDir(final StringBuffer directory, final StringBuffer file, final String
networkName) //loggingplugin.java
...
671. final File dir = new File(directory.toString()+network+System.getProperty("file.separator"));
```

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

---

#### Path 15:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
661. protected void addNetworkDir(final StringBuffer directory, final StringBuffer file, final String
networkName) //loggingplugin.java
...
686. directory.append(System.getProperty("file.separator"));
```

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

---

#### Path 16:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
200. public static boolean isApple() //apple.java
...
201. return (System.getProperty("mrj.version") != null);
```

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

---

#### Path 17:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
133. protected final JButton getLeftButton() //standarddialog.java
```

```
134. if (System.getProperty("os.name").toLowerCase().startsWith("win")) {
```

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

Path 18:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
145. protected final JButton getRightButton() //standarddialog.java
    |
    |146. if (System.getProperty("os.name").toLowerCase().startsWith("win")) {
```

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

Path 19:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
159. protected final void orderButtons(final JButton leftButton, //standarddialog.java
    |
    |...
    |161. if (System.getProperty("os.name").toLowerCase().startsWith("win")) {
```

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

Path 20:

Query Name - **Improper\_Exception\_Handling**

Severity -  **Warning**

```
260. private void addProfile() //profilemanagerdialog.java
    |
    |261. final String nick = System.getProperty("user.name").replace(' ', '_');
```

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

Using a Broken or Risky Cryptographic Algorithm	
CWE ID	327
Description	The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the disclosure of sensitive information.
Likelihood of Exploit	Medium to High
Common Consequences	Confidentiality: The confidentiality of sensitive data may be compromised by the use of a broken or risky cryptographic algorithm. Integrity: The integrity of sensitive data may be compromised by the use of a broken or risky cryptographic algorithm. Accountability: Any accountability to message content preserved by cryptography may be subject to attack.
Potential Mitigations	Design: Use a cryptographic algorithm that is currently considered to be strong by experts in the field. You should choose a tested and widely used implementation. As with all cryptographic mechanisms, the source code should be available for analysis.
Demonstrative Examples	<b>C/C++ Example:</b> <pre>EVP_des_ecb();</pre> <b>Java Example:</b> <pre>Cipher des=Cipher.getInstance("DES..."); des.initEncrypt(key2);</pre>
Context Notes	<p>Cryptographic algorithms are the methods by which data is scrambled. There are a small number of well understood and heavily studied algorithms that should be used by most applications. It is quite difficult to produce a secure algorithm, and even high profile algorithms by accomplished cryptographic experts have been broken. The use of a non-standard algorithm is dangerous because a determined attacker may be able to break the algorithm and compromise whatever data has been protected.</p> <p>Since the state of cryptography advances so rapidly, it is common to find algorithms, which previously were considered to be safe, currently considered unsafe. In some cases, things are discovered, or processing speed increases to the degree that the cryptographic algorithm provides little more benefit than the use of no cryptography at all.</p>
Node Relationships	ChildOf - <a href="#">Weak Encryption</a> (326) Peer - <a href="#">Failure to Encrypt Data</a> (311)
Source Taxonomies	CLASP - Using a broken or risky cryptographic algorithm
Applicable Platforms	All

Path 1:  
Query Name - Weak\_Cryptographic\_Algorithm  
Severity - ⚠️Warning

```
707. protected static String md5(final String string) //loggingplugin.java
    |
    |...
709. final MessageDigest m = MessageDigest.getInstance("MD5");
```

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

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

```
157. public void run() //dcc.java
    |
    |...
178. serverSocket.close();
```

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

Path 2:  
Query Name - Unclosed\_Objects  
Severity - ⓘInformation

```
199. protected void close() //dcc.java
    |
    |...
213. serverSocket.close();
```

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

Path 3:  
Query Name - Unclosed\_Objects  
Severity - ⓘInformation

```
199. protected void close() //dcc.java
    |
    |...
228. socket.close();
```

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

Path 4:  
Query Name - Unclosed\_Objects  
Severity - ⓘInformation

```
327. protected void socketClosed() //dcssend.java
    |
    |...
329. if (out != null) { try { out.close(); } catch (IOException e) { } }
```

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

Path 5:  
Query Name - Unclosed\_Objects  
Severity - ⓘInformation

```
327. protected void socketClosed() //dcssend.java
```

```
...
330. if (in != null) { try { in.close(); } catch (IOException e) { } }
```

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

Path 6:  
Query Name - **Unclosed\_Objects**  
Severity -  **Information**

```
364. protected boolean handleReceive() //dccsend.java
...
377. fileOut.close();
```

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

Path 7:  
Query Name - **Unclosed\_Objects**  
Severity -  **Information**

```
364. protected boolean handleReceive() //dccsend.java
...
383. fileOut.close();
```

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

Path 8:  
Query Name - **Unclosed\_Objects**  
Severity -  **Information**

```
398. protected boolean handleSend() //dccsend.java
...
418. fileIn.close();
```

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

Path 9:  
Query Name - **Unclosed\_Objects**  
Severity -  **Information**

```
398. protected boolean handleSend() //dccsend.java
...
441. fileIn.close();
```

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

Path 10:  
Query Name - **Unclosed\_Objects**  
Severity -  **Information**


```
54. public static List<String> getDcopResult(final String command) //dcopplugin.java
...
74. input.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>... try { File.Copy(path, path2);</div>


```
    }
    catch(IOException e)
    {
        Console.WriteLine(e.Message);
    }
    ...
}
```

Applicable Platforms      All

Path 1:  
Query Name - **Catch\_Without\_General\_Exception\_Handling**  
Severity      -  **Information**


```
113. protected void loadTrustedCAs() //certificatemanager.java
    |
    |...
    |116. try {
```

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

Path 2:  
Query Name - **Catch\_Without\_General\_Exception\_Handling**  
Severity      -  **Information**


```
113. protected void loadTrustedCAs() //certificatemanager.java
    |
    |...
    |139. try {
```

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

Path 3:  
Query Name - **Catch\_Without\_General\_Exception\_Handling**  
Severity      -  **Information**


```
154. public KeyManager[] getKeyManager() //certificatemanager.java
    |
    |...
    |157. try {
```

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

Path 4:  
Query Name - **Catch\_Without\_General\_Exception\_Handling**  
Severity      -  **Information**


```
154. public KeyManager[] getKeyManager() //certificatemanager.java
    |
    |...
    |187. try {
```

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

Path 5:  
Query Name - **Catch\_Without\_General\_Exception\_Handling**  
Severity      -  **Information**

```
238. public boolean isValidHost(final X509Certificate certificate) //certificatemanager.java
    |
    |...
    |244. try {
```

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

Path 6:  
Query Name - **Catch\_Without\_General\_Exception\_Handling**  
Severity      -  **Information**

```
264. public void checkServerTrusted(final X509Certificate[] chain, final String authType) //certificatemanager.java
    |
    |...
    |284. try {
```

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

Path 7:  
Query Name - **Catch\_Without\_General\_Exception\_Handling**

Severity -  Information

```
264. public void checkServerTrusted(final X509Certificate[] chain, final String authType) //certificatemanager.java
    |
    |...
    |306. try {
```

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

Path 8:

Query Name - **Catch\_Without\_General\_Exception\_Handling**

Severity -  Information

```
357. public static Map<String, String> getDNFieldsFromCert(final X509Certificate cert) //certificatemanager.java
    |
    |...
    |360. try {
```

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

Path 9:

Query Name - **Catch\_Without\_General\_Exception\_Handling**

Severity -  Information

```
46. public void registerCallbacks() //eventhandler.java
    |
    |...
    |49. try {
```

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

Path 10:

Query Name - **Catch\_Without\_General\_Exception\_Handling**

Severity -  Information

```
68. public boolean canConvert() //ignorelist.java
    |
    |69. try {
```

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

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 - <a href="#">Code Quality</a> (398)
Source Taxonomies	CLASP - Incorrect block delimitation
Applicable Platforms	All

Path 1:

Query Name - **Single\_Line\_If\_Statement**

Severity -  Information

```
95. public final static char[] encodeToChar(byte[] sArr, boolean lineSep) //base64.java
    |
    |...
    |99. if (sLen == 0)
```

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

Path 2:

Query Name - **Single\_Line\_If\_Statement**



Severity -  Information

```
147. public final static byte[] decode(char[] sArr) //base64.java
    ...
    151. if (sLen == 0)
```

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

Path 3:

Query Name - **Single\_Line\_If\_Statement**

Severity -  Information

```
147. public final static byte[] decode(char[] sArr) //base64.java
    ...
    162. if ((sLen - sepCnt) % 4 != 0)
```

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

Path 4:

Query Name - **Single\_Line\_If\_Statement**

Severity -  Information

```
147. public final static byte[] decode(char[] sArr) //base64.java
    ...
    179. if (c >= 0)
```

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

Path 5:

Query Name - **Single\_Line\_If\_Statement**

Severity -  Information

```
147. public final static byte[] decode(char[] sArr) //base64.java
    ...
    188. if (d < len)
```

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

Path 6:

Query Name - **Single\_Line\_If\_Statement**

Severity -  Information

```
204. public final static byte[] decodeFast(char[] sArr) //base64.java
    ...
    208. if (sLen == 0)
```

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

Path 7:

Query Name - **Single\_Line\_If\_Statement**

Severity -  Information

```
271. public final static byte[] encodeToByte(byte[] sArr, boolean lineSep) //base64.java
    ...
    275. if (sLen == 0)
```

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

Path 8:

Query Name - **Single\_Line\_If\_Statement**

Severity -  Information

```
323. public final static byte[] decode(byte[] sArr) //base64.java
    ...
    336. if ((sLen - sepCnt) % 4 != 0)
```

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

Path 9:

Query Name - **Single\_Line\_If\_Statement**

```
323. public final static byte[] decode(byte[] sArr) //base64.java
    |
    |...
353. if (c >= 0)
```

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

[\[Description\]](#)

Path 10:

Query Name - **Single\_Line\_If\_Statement**

Severity -  Information

```
323. public final static byte[] decode(byte[] sArr) //base64.java
    |
    |...
363. if (d < len)
```


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

[\[Description\]](#)

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><b>C Example:</b><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><b>C++ Example:</b><pre>while (DoSomething()) {     try {         /* perform main loop here */     }     catch (Exception e) {         /* do nothing, but catch so it'll compile... */     } }</pre></div> <div><b>Java Example:</b><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 - <a href="#">Error Conditions</a> , <a href="#">Return Values</a> , <a href="#">Status Codes</a> (389)
Source Taxonomies	CLASP - Improper error handling
Applicable Platforms	All

Path 1:

Query Name - **Empty\_Catch**

Severity -  Information

```
113. protected void loadTrustedCAs() //certificatemanager.java
    |
    |...
141. } catch (IOException ex) {
```

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

[\[Description\]](#)

---

## Path 2:

Query Name - **Empty\_Catch**

Severity -  **Information**

```
154. public KeyManager[] getKeyManager() //certificatemanager.java
    |
    |...
    |189. } catch (IOException ex) {
```

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

---

## Path 3:

Query Name - **Empty\_Catch**

Severity -  **Information**

```
357. public static Map<String, String> getDNFieldsFromCert(final X509Certificate cert) //certificatemanager.java
    |
    |...
    |365. } catch (InvalidNameException ex) {
```

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

---

## Path 4:

Query Name - **Empty\_Catch**

Severity -  **Information**

```
212. private void checkMetaData() //action.java
    |
    |...
    |228. } catch (NumberFormatException ex) {
```

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

---

## Path 5:

Query Name - **Empty\_Catch**

Severity -  **Information**

```
212. private void checkMetaData() //action.java
    |
    |...
    |236. } catch (NumberFormatException ex) {
```

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

---

## Path 6:

Query Name - **Empty\_Catch**

Severity -  **Information**

```
106. public BrowserWindow() //browserwindow.java
    |
    |...
    |151. } catch (IOException ex) {
```

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

---

## Path 7:

Query Name - **Empty\_Catch**

Severity -  **Information**

```
106. public BrowserWindow() //browserwindow.java
    |
    |...
    |152. } catch (InvalidConfigFileException ex) {
```

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

---

## Path 8:


Query Name - **Empty\_Catch**

Severity -  **Information**

```
52. public void run() //downloaderwindow.java
    |
    |...
    |57. } catch (IOException ex) {
```


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

---

Path 9:  
Query Name - **Empty\_Catch**  
Severity -  **Information**

```
115. private void playWav() //audioplayer.java
    |
    |...
    |119. } catch (MalformedURLException e) { }
```

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

Path 10:  
Query Name - **Empty\_Catch**  
Severity -  **Information**

```
129. public void listen(final int startPort, final int endPort) throws IOException //dcc.java
    |
    |...
    |139. } catch (IOException ioe) { }
```

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

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:<pre>switch (value) {   case 'A':     printf("A!\n");     break;   case 'B':     printf("B!\n");     break;   default:     printf("Neither A nor B\n"); }</pre></div> <p>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.</p>
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 - <a href="#">Code Quality</a> (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**

```
264. public void checkServerTrusted(final X509Certificate[] chain, final String authType) //certificatemanager.java
    |
    |...
    |312. switch (action) { }
```

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

Path 2:

Query Name - No\_Default\_Case

Severity -  Information

```
59. public void actionPerformed(final(ActionEvent) e) //installlistener.java
    |
    |...
    |67. switch (info.getType()) {
```

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

Path 3:

Query Name - No\_Default\_Case

Severity -  Information

```
242. protected void handleQueryEvent(final CoreActionType type, final StringBuffer format, final Object...
arguments) //loggingplugin.java
    |
    |...
    |268. switch (type) {
```

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

Path 4:

Query Name - No\_Default\_Case

Severity -  Information

```
313. protected void handleChannelEvent(final CoreActionType type, final StringBuffer format, final Object...
arguments) //loggingplugin.java
    |
    |...
    |323. switch (type) {
```

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

Path 5:

Query Name - No\_Default\_Case

Severity -  Information

```
231. public void tableChanged(final TableModelEvent e) //errorlistdialog.java
    |
    |232. switch (e.getType()) {
```

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

Path 6:

Query Name - No\_Default\_Case

Severity -  Information

```
76. public void processEvent(final ActionType type, final StringBuffer format, //userlevelplugin.java
    |
    |...
    |78. switch ((CoreActionType) type) {
```

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


Call to Thread.run()	
CWE ID	572
Description	The program calls a thread's run() method instead of calling start()
Affected Resource	System Process
Demonstrative Examples	<div>The following excerpt from a Java program mistakenly calls run() instead of start().<pre>Thread thr = new Thread() {     public void run() {         ...     } };  thr.run();</pre></div>
Context Notes	In most cases a direct call to a Thread object's run() method is a bug. The programmer intended to begin a new thread of control, but accidentally called run() instead of start(), so the run() method will execute in the caller's thread of control.
Node Relationships	ChildOf - <a href="#">Concurrency Issues</a> (557) ChildOf - <a href="#">Race Condition within a Thread</a> (366) ChildOf - <a href="#">Weaknesses that Affect System Processes</a> (634)
Applicable Platforms	Java

Path 1:  
Query Name - **Call\_to\_Thread\_run**  
Severity -  **Information**

```
190. public static <T> T invokeAndWait(final ReturnableThread<T> returnable) //uiutilities.java
    |
    |...
    |192. returnable.run();
```


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

Erroneous String Compare	
CWE ID	597
Description	Strings should be compared with the equals() method, not == or !=
Demonstrative Examples	<div>The following branch will never be taken.<pre>if (args[0] == STRING_CONSTANT) {     logger.info("miracle"); }</pre></div>
Context Notes	Using == or != to compare two strings for equality actually compares two objects for equality, not their values. Chances are good that the two references will never be equal.
Node Relationships	ChildOf - <a href="#">Incorrect Object Comparison: Syntactic</a> (595) ChildOf - <a href="#">Incorrect Object Comparison: Syntactic</a> (595) ChildOf - <a href="#">String Errors</a> (133)

Path 1:  
Query Name - **Erroneous\_String\_Compare**  
Severity -  **Information**

```
70. public boolean equals(final Object obj) //previouscommand.java
    |
    |...
    |80. if (this.line != other.line
```

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

Path 2:  
Query Name - **Erroneous\_String\_Compare**  
Severity -  **Information**

```
70. public boolean equals(final Object obj) //previouscommand.java
    |
    |...
    |80. if (this.line != other.line
```

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

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**

```
53. public final class LagDisplayPlugin extends Plugin implements ActionListener, ConfigChangeListener //lagdisplayplugin.java
    |
    |...
    |72. private int historySize = 100;
```

Path 2:

Query Name - **Magic\_Numbers**

Severity     -    **Information**

```
128. public void paint(final Graphics g) //colourpickerpanel.java
    |
    | 129. int offset = 20;
```

Path 3:

Query Name - **Magic\_Numbers**

Severity     -    **Information**

```
59. public final class SwingPreferencesDialog extends StandardDialog implements //swingpreferencesdialog.java
    |
    | ...
    | 73. public static int CLIENT_HEIGHT = 375;
```

Path 4:

Query Name - **Magic\_Numbers**

Severity     -    **Information**

```
362. public int hashCode() //profile.java
    |
    | 363. int hash = 5;
```

Path 5:

Query Name - **Magic\_Numbers**

Severity     -    **Information**

```
90. public int hashCode() //previouscommand.java
    |
    | 91. int hash = 5;
```

Path 6:

Query Name - **Magic\_Numbers**

Severity     -    **Information**

```
52. public void execute(final InputWindow origin, final boolean isSilent, //newserver.java
    |
    | ...
    | 62. int port = 6667;
```

Path 7:

Query Name - **Magic\_Numbers**

Severity     -    **Information**

```
51. public void execute(final InputWindow origin, final Server server, //changeserver.java
    |
    | ...
    | 61. int port = 6667;
```

Path 8:

Query Name - **Magic\_Numbers**

Severity     -    **Information**

```
34. public class ConfigTarget implements Comparable, Serializable //configtarget.java
    |
    | ...
    | 70. protected int order = 50000;
```

Path 9:  
Query Name - Magic\_Numbers  
Severity - Information

```
416. public int hashCode() //programerror.java
    |
    |417. int hash = 7;
```

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

Path 10:  
Query Name - Magic\_Numbers  
Severity - Information

```
60. public class IRCParser implements Runnable //ircparser.java
    |
    |...
    |99. private long pingTimerLength = 10000;
```

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

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

```
36. public abstract class DCC implements Runnable //dcc.java
    |
    |...
    |48. protected boolean listen = false;
```

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

Path 2:  
Query Name - Confusing\_Naming  
Severity - Information

```
49. public abstract class DCCFrame extends WritableFrameContainer //dccframe.java
    |
    |...
    |116. private boolean windowClosing = false;
```

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

Path 3:  
Query Name - Confusing\_Naming  
Severity - Information

```
33. public class PluginInfoToggle //plugininfotoggle.java
    |
    |...
    |39. private boolean toggle = false;
```

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

Path 4:  
Query Name - Confusing\_Naming  
Severity - Information

```
36. public class PreferencesCategory //preferencescategory.java
    |
    |...
    |52. private boolean isInline = false;
```

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



Path 5:

Query Name - Confusing\_Naming

Severity - Information

```
40. public class ChannelInfo //channelinfo.java
    ...
    88. private boolean hasGotListModes = false;
```

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

Path 6:

Query Name - Confusing\_Naming

Severity - Information

```
31. public class Logging //logging.java
    ...
    80. private Object log = null;
```

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

Path 7:

Query Name - Confusing\_Naming

Severity - Information

```
35. public class TestSSLCertificateDialogModel extends SSLCertificateDialogModel //testsslcertificatedialogmodel.java
    ...
    38. public boolean needsResponse = true;
```

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

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> <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>

Path 1:


Query Name - Overly\_Broad\_Catch

Severity - Information

```
214. public boolean isTrusted(final X509Certificate certificate) //certificatemanager.java
    |
    |...
    |231. } catch (Exception ex) {
```

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

---

Path 2:  
Query Name - **Overly\_Broad\_Catch**  
Severity     -    **Information**

```
243. public int showOpenDialog(final Component parent) throws HeadlessException //kfilechooser.java
    |
    |...
    |277. } catch (Exception e) {
```

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


---

Path 3:  
Query Name - **Overly\_Broad\_Catch**  
Severity     -    **Information**

```
300. public int showSaveDialog(final Component parent) throws HeadlessException //kfilechooser.java
    |
    |...
    |323. } catch (Exception e) {
```

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


---

Path 4:  
Query Name - **Overly\_Broad\_Catch**  
Severity     -    **Information**

```
349. public void run() //mainframe.java
    |
    |...
    |357. } catch (Exception ex) {
```

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


---

Path 5:  
Query Name - **Overly\_Broad\_Catch**  
Severity     -    **Information**

```
34. public ValidationResponse validate(final String object) //regexvalidator.java
    |
    |...
    |41. } catch (Exception ex) {
```

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

---

Path 6:  
Query Name - **Overly\_Broad\_Catch**  
Severity     -    **Information**

```
984. protected void processLine(final String line) //ircparser.java
    |
    |...
    |1051. } catch (Exception e) {
```

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


---

Path 7:  
Query Name - **Overly\_Broad\_Catch**  
Severity     -    **Information**

```
139. public boolean call(final Object ... args) //callbackobject.java
    |
    |...
    |153. } catch (Exception e) {
```


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

---

Path 8:  
Query Name - **Overly\_Broad\_Catch**  
Severity     -    **Information**


```
132. public boolean call(final Object ... args) //callbackobjectspecific.java
    |
    |...
    |157. } catch (Exception e) {
```

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

Path 9:  
Query Name - **Overly\_Broad\_Catch**  
Severity -  **Information**

```
1042. private void unloadPlugin(final boolean parentUnloading) //plugininfo.java
    |
    |...
    |1061. } catch (Exception e) {
```

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

Path 10:  
Query Name - **Overly\_Broad\_Catch**  
Severity -  **Information**

```
32. public void testConstructorNegativeID() //programerrortest.java
    |
    |...
    |37. } catch (Exception ex) {
```

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

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 - <a href="#">Error Conditions, Return Values, Status Codes</a> (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**

```
499. protected Object doInBackground() throws Exception //mainframe.java
    |
```

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

Path 2:  
Query Name - **Overly\_Broad\_Throws**  
Severity -  **Information**

```
152. protected Object doInBackground() throws Exception //swinginputhandler.java
    |
```

Path 3:


Query Name - **Overly\_Broad\_Throws**

Severity -  **Information**

```
190. protected Object doInBackground() throws Exception //swinginputhandler.java
    |
```

Path 4:

Query Name - **Overly\_Broad\_Throws**

Severity -  **Information**

```
452. protected Object doInBackground() throws Exception //textframe.java
    |
```

Path 5:

Query Name - **Overly\_Broad\_Throws**

Severity -  **Information**

```
646. protected Object doInBackground() throws Exception //textframe.java
    |
```

Path 6:


Query Name - **Overly\_Broad\_Throws**

Severity -  **Information**

```
664. protected Object doInBackground() throws Exception //textframe.java
    |
```

Path 7:


Query Name - **Overly\_Broad\_Throws**

Severity -  **Information**

```
682. protected Object doInBackground() throws Exception //textframe.java
    |
```

Path 8:


Query Name - **Overly\_Broad\_Throws**

Severity -  **Information**

```
731. protected Object doInBackground() throws Exception //textframe.java
    |
```

Path 9:


Query Name - **Overly\_Broad\_Throws**

Severity -  **Information**

```
754. protected Object doInBackground() throws Exception //textframe.java
    |
```

Path 10:

Query Name - **Overly\_Broad\_Throws**


Severity -  **Information**

```
375. protected Object doInBackground() throws Exception //feedbackdialog.java
    |
```

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

```
44. public abstract class FrameContainer //framecontainer.java
    |
    | ...
    | 51. protected Color notification = Color.BLACK;
```

Path 2:


Query Name - Visible\_Fields

Severity -  Information

```
34. public class ServerStatus //serverstatus.java
    |
    | ...
    | 37. protected ServerState state = ServerState.DISCONNECTED;
```

Path 3:


Query Name - Visible\_Fields

Severity -  Information

```
34. public class ServerStatus //serverstatus.java
    |
    | ...
    | 40. protected RollingList<String> history = new RollingList<String>(10);
```

Path 4:

Query Name - Visible\_Fields

Severity -  Information

```
49. public class Action extends ActionModel implements Serializable //action.java
    |
    | ...
    | 73. protected ConfigFile config;
```

Path 5:

Query Name - Visible\_Fields

Severity -  Information

```
45. public class ActionModel //actionmodel.java
    ...
    48. protected String group;
```

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

---

#### Path 6:

Query Name - **Visible\_Fields**

Severity -  **Information**

```
45. public class ActionModel //actionmodel.java
    ...
    51. protected String name;
```

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

---

#### Path 7:

Query Name - **Visible\_Fields**

Severity -  **Information**

```
45. public class ActionModel //actionmodel.java
    ...
    54. protected ActionType[] triggers;
```

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

---

#### Path 8:

Query Name - **Visible\_Fields**

Severity -  **Information**

```
45. public class ActionModel //actionmodel.java
    ...
    57. protected String[] response;
```

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

---

#### Path 9:

Query Name - **Visible\_Fields**

Severity -  **Information**

```
45. public class ActionModel //actionmodel.java
    ...
    60. protected String newFormat;
```

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

---

#### Path 10:

Query Name - **Visible\_Fields**

Severity -  **Information**

```
45. public class ActionModel //actionmodel.java
    ...
    63. protected List<ActionCondition> conditions = new ArrayList<ActionCondition>();
```

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

---