"In the name of God"

Author: Soroush Dalili
Email Address: irsdl {a7} yahoo {d07} com
Website: Soroush.SecProject.com
Title of Report: Finding vulnerabilities of YaFtp 1.0.14 (a client-side FTP application)
Application Website: http://sourceforge.net/projects/yaftp/files/
Language: Java
Purpose: Educational Purpose (Secure Programming 2008/09 Exercise)

Abstract

In this report we are going to find the vulnerabilities of YaFtp program, a client-side FTP application, and we are also going to suggest some mitigation methods. This process will be performed by using a specific plan which plays an important role in finding the security issues and analyzing the program. First of all we must understand the problem and gather the information which is related to this program. In fact, gathering the information is the most important phase in finding the vulnerabilities which clears the problem for us. In the next phase, model of the application will be drawn. Then, possible vulnerabilities will be discussed and we will draw two possible attack trees for YaFtp program. Finally, by using some automation tools and also manually, we will find the vulnerable candidate points, and we will investigate them to find the vulnerabilities. To summarize, 9 important vulnerabilities were found in this report. And, there are some solutions and suggestions in the last section of this report for developers of this application.

1 Introduction

YaFtp (Yet Another File Transfer Program) is a stand-alone application which is used to connect to the FTP server. So, this program is a client-side FTP application. At first sight, it seems very difficult to find vulnerability in this program. Because it is used on the client-side and it does not listen to any port by default, and also it does not use any databases; moreover, the language of this program is Java. So, there is not any general vulnerability such as a buffer overflow or a SQL injection in it. However, by having critical and precise look, we can understand that this program is also vulnerable to some kinds of attacks. And in this report, we want to discuss about finding and preventing from these security issues. In the next chapter, the plan of performing the security analysis will be described.

2 General Plan

Having a specific plan plays an important role in finding the vulnerabilities and security analyses of a program. The plan which is used in this report is based on the [1] and also experience of the author about reporting the security vulnerabilities:

- Understanding the problem (basic information)
- Information gathering
    - View the related documents and RFCs about the technology which is used
    - View the application's programming language security guidelines

    - View documents of the program
    - View the older vulnerabilities in the program and also the similar applications.
    - Files and directories
        - Directory structure
        - Find important files and folders which are contain sensitive data
    - Source code of the program
        - General idea of the program according to the source code
        - Executable modules
        - External libraries or packages which are used
        - Imports and exports any external resources such as the connection, files, and so on (entry and exit points of the program)
- Modeling the application
- Possible Vulnerabilities - Drawing the Attack Trees
- Finding the vulnerabilities (Detailed security audit)

Following chapters follow this plan to get the best result.

3 Understanding the Problem (Basic Information)

We are going to find the vulnerabilities of YaFtp program. YaFtp is a client-side application which is run on the pc of a user. This program is based on the FTP RFCs and its language is Java.

4 Information Gathering

The specific information of the program which is going to be analyzed in this report is in appendix A.
The useful RFCs to study for FTP are [2], [3], and [4]. Moreover [7], [8], [9], and [10] are some good documents about the FTP and its functionality.
Useful security guidelines of Java are in [5] and [6].The documents of the program are in "/doc" folder.
There was not any vulnerability for this application to the date of this report in public. But, there were some vulnerabilities in some other FTP clients and their links are in the appendix B.
Directory structure of the application is coming in appendix C.
Important files and directories are:

- "ftpTraces.trc" which is in "/init" or "/" directories. (Contains trace information).
- "Ftp.properties" which is in "/init" folder. (Contains application settings and configuration)
- Ftp work directory which is set in "/init/Ftp.properties". (Contains downloaded files)

There are 6 packages in this application:

- com.yaftp.ftp (Contains main FTP classes)
- com.yaftp.ftp.gui (Contains GUI classes)
- com.yaftp.ftp.gui.images (Contains GUI images + 1 old .trc file)
- com.yaftp.ftp.mvsjobs (Contains MVS idiosyncrasy classes)
- com.yaftp.utils (Contains general and utility functions)
- com.yaftp.utils.images (Contains utilities images)

This program uses Swing and also AWT in some places.
The explanation of the methods and classes are in the source code and also JavaDocs of the program.
There is some more functionality for IBM OS390 FTP server in comparison with the Linux and Windows FTP which must be tested as well.

5 Modeling the Application

As YaFtp is a FTP client and we want to audit its security, so the trust boundary zone is around the client. And, server and other users can assume as external entities (or attackers). The simplest shape of the modeling of this application is in figure 1:

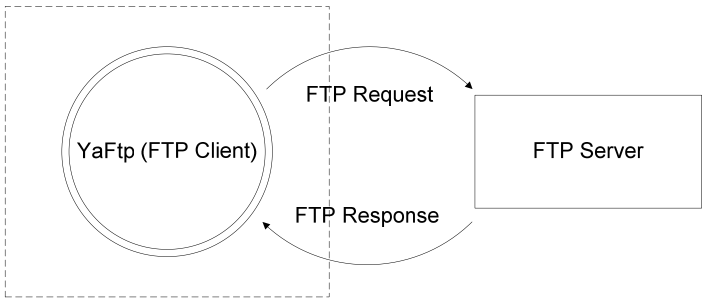


Figure 1 - Simplest DFD context diagram

The details of FTP transactions itself are in [8].

YaFtp send a FTP request and receive a FTP response and by using this response decide what it should to do next. For instance, by sending a CWD command (for changing the directory) to the server, it receives some information and if there is not an error from the server, it will show the results of directories for the user otherwise shows that error. Other information of this program is achieved in the section 4 and also by looking at its JavaDocs.

6 Possible Vulnerabilities - Drawing the Attack Trees

According to [1], we must start writing the attack trees and possible vulnerabilities before entering to the finding section. Reference [1] defines vulnerability as *a bug which allows attackers to do something they normally wouldn't be able to do*. And, everything that threat the CIA triangle (Confidentiality, Integrity, and Availability) can be assumed as vulnerability or a security flaw. Since our scope is bounded to the client, server can be also assumed as an attacker.
There are some possible vulnerabilities for a client FTP:

- Compromising the FTP confidential information. (For Ex. disclosure of username and password of the FTP server)
- Compromising the client confidential information. (For Ex. disclosure of the open ports)
- Malicious file manipulation on the client. (For Ex. copy a file on the startup folder)
- Executing arbitrary commands on the client. (For Ex. executing another part of the program)
- Attacking a FTP server to another system by using the FTP client application. (For Ex. force the client to send username and password of another open FTP server to the attacker FTP server)
- Exploit the internal modules of the FTP client to perform some other kinds of attacks. (For Ex. performing a cross site scripting attack in the internal html browser of the FTP client)
- Performing a denial of service on the client.

Although the above list may not be a complete list, it helps us to understand how to draw the attack trees.
The attack tree for “compromising the user confidential information” is in figure 2. In fact, this application stores username and password of the user on the memory and also on the hard disk if the

"trace" option of the program has been selected. So, this attack tree contains the threat about "information disclosure" and also "information leakage".
Another attack tree for "changing the normal work of the application to do some malicious things" is in figure 3. This attack tree includes all those types of attacks which are not related to the information disclosure.
After designing the attack trees, we must start finding these vulnerabilities at the target program which is YaFtp. Furthermore, by using some methods and techniques in finding the security flaws in some cases, we can complete and precise our attack trees more than before.

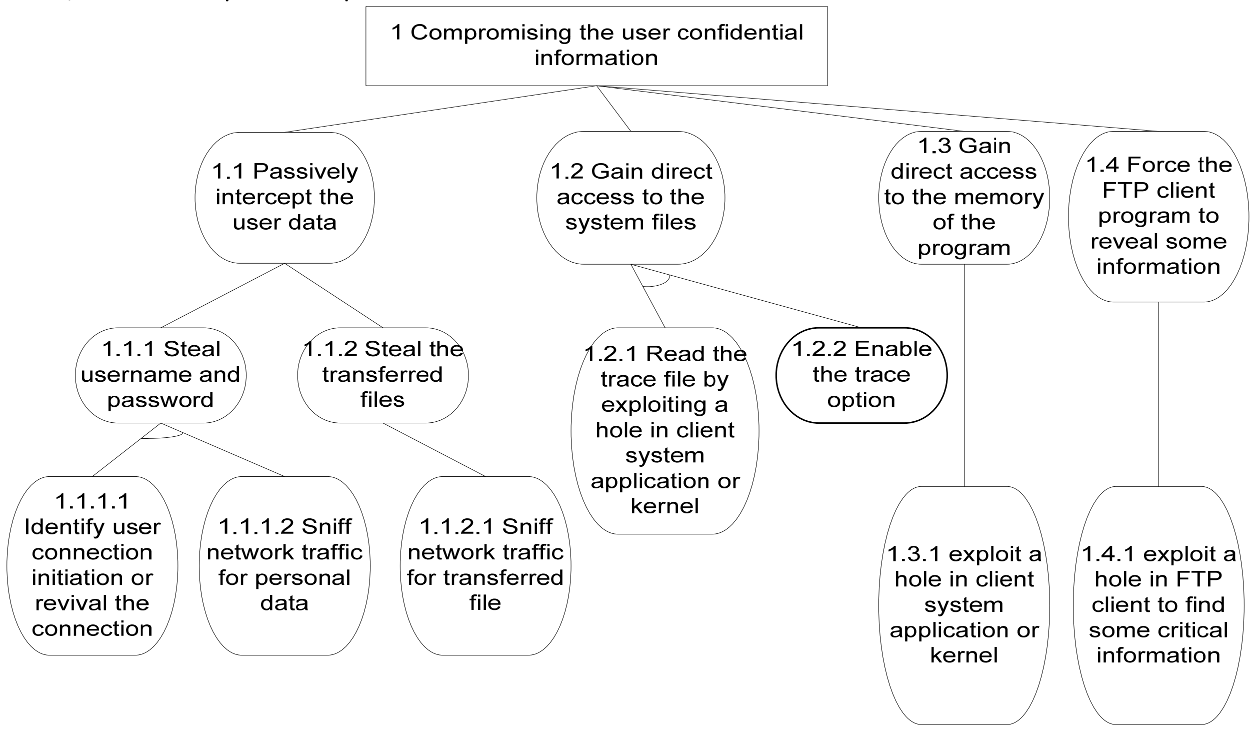
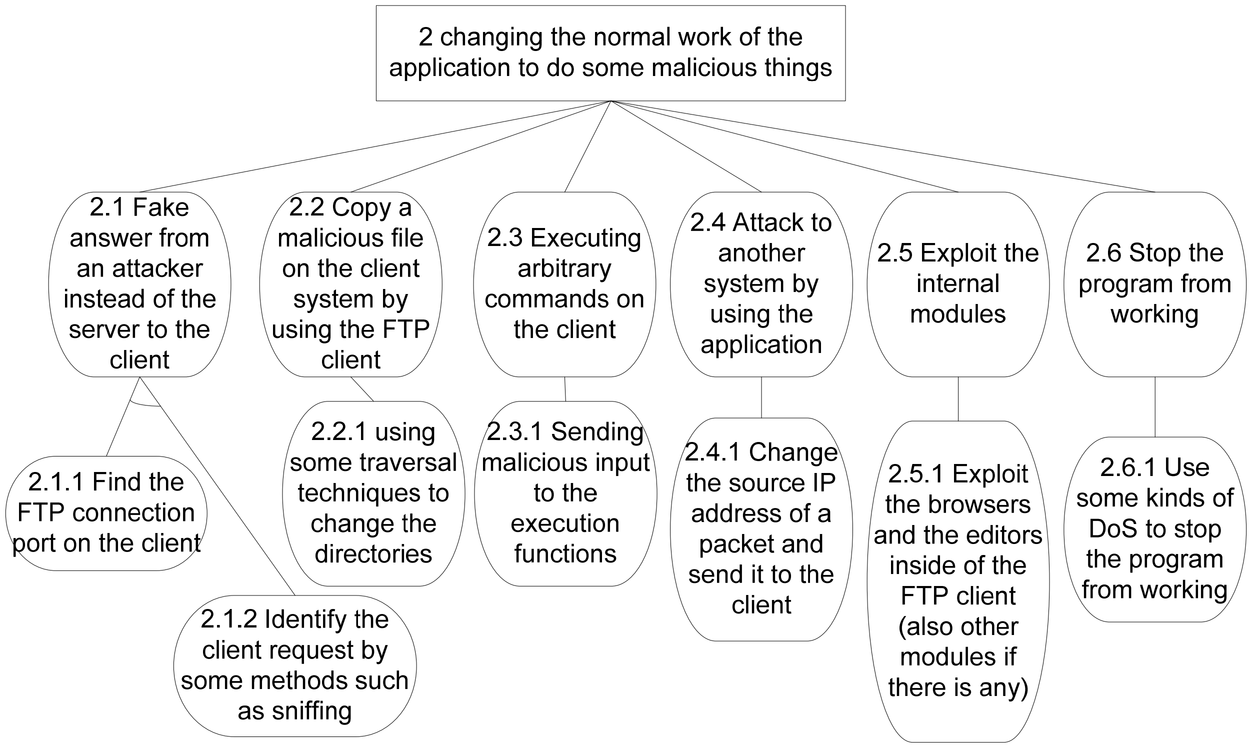

1 Compromising the user confidential information
1.1 Passively intercept the user data
1.2 Gain direct access to the system files
1.3 Gain direct access to the memory of the program
1.4 Force the FTP client program to reveal some information
1.1.1 Steal username and password
1.1.2 Steal the transferred files
1.2.1 Read the trace file by exploiting a hole in client system application or kernel
1.2.2 Enable the trace option
1.1.1.1 Identify user connection initiation or revival the connection
1.1.1.2 Sniff network traffic for personal data
1.1.2.1 Sniff network traffic for transferred file
1.3.1 exploit a hole in client system application or kernel
1.4.1 exploit a hole in FTP client to find some critical information

Figure 2 – Attack tree for "compromising the user confidential information"



2 changing the normal work of the application to do some malicious things
2.1 Fake answer from an attacker instead of the server to the client
2.2 Copy a malicious file on the client system by using the FTP client
2.3 Executing arbitrary commands on the client
2.4 Attack to another system by using the application
2.5 Exploit the internal modules
2.6 Stop the program from working
2.1.1 Find the FTP connection port on the client
2.1.2 Identify the client request by some methods such as sniffing
2.2.1 using some traversal techniques to change the directories
2.3.1 Sending malicious input to the execution functions
2.4.1 Change the source IP address of a packet and send it to the client
2.5.1 Exploit the browsers and the editors inside of the FTP client (also other modules if there is any)
2.6.1 Use some kinds of DoS to stop the program from working

Figure 3 – Attack tree for "changing the normal work of the application to do some malicious things"

7 Finding the Vulnerabilities (Detailed Security Audit)

In this section, finding the vulnerabilities is performed in three stages:

- Using automation tools to find the candidate points for the security flaws and check them manually.
- Searching for dangerous functions or methods.
- Looking for some scenarios which are written in the attack trees and are not found by two previous stages.

Note1: To examine the security vulnerabilities and simulating the FTP server, "jftpd" [11] is used which is free and open source Java application.
Note2: In this Java application, executable modules which contain "main" method are not really important to check as no one can run them directly. But perhaps in some rare situation these become important. For instance, assume a situation in which an attacker only can run a java applet or a java class on the victim system, and he or she can point to these executable and vulnerable modules to exploit them. So, if it is an important function, it must be private to prevent from a security flaw otherwise we can assume it as vulnerability.

7.1 Using Automation Tools

Three automation tools are used in this section:

- Klocwork [12]
- Findbugs [13]

- PMD [14]

Pictures of these programs are in appendix D.

7.1.1 Using Klocwork

In the severity of “Review”, “Style”, and “Suggestion” there are some points about the performance of the program and suggest replacing some Java codes to have better performance and improvement. Because of the fact that these kinds of performance problems never make a security issue such as a denial of service for a client, there is not any important thing related to the security flaws in them.

There is a warning message on “utils.ProcessLauncher. setCommand” which does not use any exception for “java.lang.NullPointerException”. So, there will be an error when this module is run without any input. And, it really works in practice. So, if we could find a way to run this method without any input, this would be a denial of service flaw. Another important thing about this function is: This function executes some commands on the operating system, so this is really important to check this function.
Now, we have two things to investigate them about this function:

- Arbitrary code execution by using this function:
  Base on note1 in section 7.1, if we can run arbitrary commands on the client system by this module directly we assume it as a security flaw in some situation. So, in this place we have a security vulnerability which is obvious by:
  java -cp %CLASSPATH%;../lib/yaftp.jar com.yaftp.utils.ProcessLauncher "Arbitrary Command"
  To prevent this vulnerability this function must be private. This vulnerability will be listed as the 1st vulnerability in section 7.4. Now, we want to check that whether it is exploitable by the remote attacker or not. So, we follow all the methods which call this function with an input.
  “OsEditorPanePanel” (line 173) creates an object from this method, and call “setCommand” at line 180 with “argString” as an input (See figure E.1 in appendix E). “argString” gets its value at line 175 or/and 178. “_editorEnvironmentString” and “_browsingEnvironmentString” are set in “/init/Ftp.properties” file and the default value for them is nothing. And, “myWkFile” is an input file which comes from “FtpSwingFListPanel. getTextEditor” which is private itself. The important point is: If the file name (“myWkFile”) was not having any filter to be validated, an attacker can execute a command by using a file name such as:
  “*test.html " | Another_Arbitrary_OS_Command | foo*”
  or
  “*test.html " & Another_Arbitrary_OS_Command | foo*”
  The double quotation in this filename is necessary to execute the command. Although Windows prevent from double quotation and “|” in the name of files, it is very easy to make a fake FTP server which send these filenames as an input to the client. Also, this is obvious that Linux does not have any problem by double quotation.
  There are several evidences which show that the file name does not have any filter as an input and this dangerous function is called from some classes such as “FtpJobSubmissionThread” from “FtpSwingFListPanel”. So, we have another vulnerability which is caused by the filename itself which remote server as an attacker can run an arbitrary command on the client.
  To have a successful attack, “Ftp.editorClassName” and at least one of “Ftp.externalEditorInitArgs” or “Ftp.externalBrowserInitArgs” in FTP properties file must be selected. To prevent from performing this kind of attack, file input names must be validated before passing to the other functions, and dangerous and forbidden characters must be deleted from the filenames. This vulnerability will be listed as the 2nd vulnerability in section 7.4 (attack tree – 2.3).

- Denial of service situation:
  Actually in this program, Null Pointer Exception is not really important. There is another error handling function which prevents from crashing in case of errors.

Other warnings of Klocwork are not a security issue. Also, there are some situations for race conditions which can lead to some security problems in some special cases, but here we could not find any important effect to show that it is a real vulnerability.
In "error" severity of Klocwork there are some messages on "ftp.FtpClientSession" which show that there are some inputs without any validation. In fact, all of these inputs are from the servers and they must have some validation. It is clear in this file that an attacker can send some malicious data from the server to the client to perform his/her purposes. These malicious can be everything depend on its functionality. For instance, attacker can send some crafted header packets to the client to stop YaFtp from working. In practice, by using the "jftpd" FTP server (this is mentioned in section 7 - note1) it is observable. So it would be the 3rd vulnerability.
Some candidate points, which Klocwork specifies, show that there are some important objects without any close section. And, this can cause denial of service for the FTP client during the time, but actually these are only some bugs because there is not any attacker which forces the YaFtp to do that.

7.1.2 Using Findbugs

There are some similar selected points as we had them by using Klocwork. So, only new things will be written here.
There are several modules which FindBugs say that "*This code stores a reference to an externally mutable object into the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Storing a copy of the object is better approach in many situations.".* These modules are:
"ftp.FtpBytesListener.set_dataList" , "ftp.gui.SwingFtpTable.setData", "ftp.SwingFtpTable.setData", "utils.SwingAboutBox.set_productInfos", "ftp.FtpOsFile.get_detailled", "yaftp.utils.CommandArgs", "utils.DataStructure", "utils.SwingStateButton", "com.yaftp.ftp.FtpBytesListener.get_dataList", "ftp.FtpOsFile.get_detailled", "ftp.gui.FtpSwingSelectedFiles.get_selected", "ftp.LocalFileConnection.get_column_names", "ftp.MVSftp.get_Column_Names", "ftp.UNIXftp.get_Column_Names", "utils.EbcdicTable.get_tAscii2Ebcdic", "utils.EbcdicTable.get_tEbcdic2Ascii"
However, we could not find any evidence to prove the FindBugs idea in these modules.

7.1.3 Using PMD

There is not any newer vulnerability by using PMD.

7.2 Searching For Dangerous Functions Manually

In this section we will look for some important Java functions manually.

These functions are:

System.exit → which leads to fast closing of the application. (Useful for performing DoS)

We ignore the normal situation to exit by clicking on the exit button. Others are:

"FtpCustomizer.main()" → no call to this method from the main program

"Notepad"→ no call to this method from the main program

getRuntime() →which is an important function to have some interaction with the OS.

A vulnerability for "Runtime.getRuntime().exec" in "utils.ProcessLauncher" module has been detected in the 7.1.1 section.

getClass().getName().equals → which is unsafe method to identify a class

No match files were found.

7.3 Looking For Some Security Scenarios

In this section according to the attack tree we will find the vulnerabilities.
Reference [1] says that "*From a design perspective, TELNET arguably has a vulnerability in that it relies on unencrypted communication."*. So, we can assume the same for this YaFtp client which relies on unencrypted communication. So this would be the 4th vulnerability which leads to information disclosure by sniffing operation (attack tree - 1.1). Mitigation is in section 7.4.
Now, we want to investigate the vulnerability of achieving username and password. YaFtp stores username and password and also server information in a .trc file by using the trace option without any encryption. An attacker can achieve the user connection information by accessing to this file. So, it is the 5th vulnerability of this program (attack tree - 1.2) (Mitigation is in section 7.4.). Furthermore, this application store the username and password in the memory without any encryption. And, in some rare situation that attacker has an access to the memory, he or she can dump the memory to find user connection information. So, there is another vulnerability – the 6th- for this program which stores confidential information of the user in memory in plaintext (attack tree - 1.3). Mitigation is in section 7.4.
At this point, we cannot speak about the (1.4) of attack tree which needs more study.
Now assume that a fake FTP server wants to attack to the client by using a malicious filename. For instance this filename can be look like this: "../../../../../users/root/startup/maliciousfile.src". Without any input validation on the FTP client, by downloading this file, it will be copied to the startup folder. In case of YaFtp, there is not any input validation on anything, so this program will be vulnerable against this attack as well as the 7th vulnerability (attack tree 2.2). Mitigation is in section 7.4.

We test an application for timeout operation but it froze completely. The scenario is this: client sends a request to connect to the FTP server, FTP server responses to the client and waits for receiving username and password. Now, client sends username and password for the server, but server does not response to the client. In this situation, YaFtp program will be frozen. So, we can assume this situation as a denial of service vulnerability (the 8th vulnerability). Mitigation is in section 7.4.

We also check the HTML browser and also the file viewer box of the program in practical and by using source code. There are not some kinds of attacks such as XSS and also it can load the huge files correctly. However, these functions need to have more precise investigation to check all the situations.

Now, assume that the program uses the passive (PASV) mode to connect to the FTP server. FTP server must send an IP address plus the connection port for the client. But, this malicious FTP server sends another computer IP address and port to the client. If the FTP client works with this new IP, it will send some packets to special port of this new computer which can lead to some security effects or information disclosure. Assume that there is an attacker which is trying to send a lot of PASV response packets to the FTP client port in order to fool the client to connect to his/her FTP server. The solution is that the FTP client must discard the IP address when using the passive mode. YaFtp uses "ftp.FtpClientSession.buildPasvSocket" method to manage the PASV connection. It is obvious that this function always work with the IP address which server sends to it. So, it is vulnerable to this kind of attack and we mention it as the 9th vulnerability (attack tree - 2.1 and 2.4).

7.4 Threat Summary

No:	**1**
Threat:	Local attacker can execute an arbitrary command on the system
Impact (1-10):	10
Probability:	2
Affected Component:	utils.ProcessLauncher
Description:	This component is a public executable class which executes the input argument as an operating system command. More details are in section 7.1.1
Result:	Because of note2 in section 7, an attacker can execute arbitrary commands on the system by using this module.
Mitigation Strategies:	Convert “public” method to “private” in this important class, and prevent from direct execution by changing the class structure.

No:	**2**
Threat:	Remote attacker can execute an arbitrary command on the system
Impact (1-10):	10
Probability:	5
Affected Component:	utils.ProcessLauncher & filename reader methods
Description:	In the situation that client uses “utils.OsEditorPanePanel” instead of “utils.SwingEditorPanePanel” and an external editor, attacker can send a malicious filename to execute an arbitrary code on the client system. This malicious filename uses the “multiple command execution” feature of the OS to execute arbitrary commands on the system. More details are in section 7.1.1
Result:	An attacker can execute arbitrary commands on the system by sending a malicious file name.
Mitigation Strategies:	Use input validation technique to filter bad characters from the filenames. Using white list is not recommended in the case that we may have some Unicode filenames.

No:	**3**
Threat:	Server can send malicious data by the header.
Impact (1-10):	5
Probability:	8
Affected Component:	ftp.FtpClientSession
Description:	This module does not check the input data which are inserted from the server. Specially “getStatus()” method does not have any protection against the malicious input and may cause some “null point exception” too.
Result:	Although it can cause some problem such as denial of service for the client, the real result is unexpected.
Mitigation Strategies:	Validate all server header according to the RFC.

No:	**4**
Threat:	Design vulnerability: Unencrypted communication
Impact (1-10):	Except the local usage is 10
Probability:	8
Affected Component:	Design
Description:	According to the reference [1], which assumes a design vulnerability for TELNET, this FTP client must use some encryption techniques to prevent information disclosure.
Result:	Attacker can find user confidential information such as username and password.
Mitigation Strategies:	Using some encryption techniques based on RFCs and standard cryptography.

No:	**5**
Threat:	Insecure method to save user connection information in a file
Impact (1-10):	8
Probability:	4
Affected Component:	Trace option
Description:	By using the trace option, YaFtp stores the user connection information (such as username, password, and server address) in a .trc file without any encryption.
Result:	Attacker can steal the confidential information of the user to connect to the target server.
Mitigation Strategies:	It should not store the password in .trc file, and it must use some strong technique to set some permission on the .trc file.

No:	**6**
Threat:	Insecure method to save user connection information in the memory
Impact (1-10):	8
Probability:	1
Affected Component:	Login and Swing - login section
Description:	If an attacker has access to the memory, he/she can dump the YaFtp memory section to achieve user connection information (such as username, password, and server address).
Result:	Attacker can steal the confidential information of the user to connect to the target server.
Mitigation Strategies:	Although this program needs to store the connection information in memory in order to reconnect to an FTP server, the way of storing these data must be safe with at least some standard encryption and randomization techniques.

No:	7
Threat:	Directory traversal on the client
Impact (1-10):	10
Probability:	8
Affected Component:	Directory name and Filename input modules
Description:	Since there is not any input validation in this FTP client, a fake FTP server can send some dangerous filename to the client in order to do the directory traversal attack.
Result:	Attacker can copy an arbitrary file on the client system in order to do some further attack.
Mitigation Strategies:	Input validation on the file and directory name to omit the harmful characters or using white list in non-Unicode situation.

No:	8
Threat:	Denial of service
Impact (1-10):	4
Probability:	7
Affected Component:	Listening modules
Description:	The program will be frozen in the situation that server suddenly stop sending the information to the client.
Result:	The program will be frozen and does not response to anything.
Mitigation Strategies:	Use some timeout method to predict this situation and free the resources when it does not need them.

No:	9
Threat:	Redirect YaFtp to another server
Impact (1-10):	8
Probability:	4
Affected Component:	ftp.FtpClientSession.buildPasvSocket
Description:	The program uses the received IP address (from FTP server) during the passive (PASV) mode.
Result:	Attacker can redirect the FTP packets of the client to another server to do some further attack or steal the user's information.
Mitigation Strategies:	Discard the new IP address in PASV mode and use the default IP address.

8 Conclusions

In summary this application is not secure enough and needs to have some security corrections. The 9 vulnerabilities of this application are mentioned in section 7.4. According to this security flaws, the main problems of the application were because of:

- Lack of input validation
- Lack of the encryption techniques

Which can lead to create some more vulnerabilities in the future.
For input validation, using the "white list" methods is suggested. "Discarding the harmful characters" is still useful in situations where the application cannot use the white list validation.
Nowadays, for encryption techniques to protect the confidential information, there are some standard ways which most of the servers support them as well.
Other vulnerabilities are related to the logic of the program which must be reviewed and improved by the time. Using the software testing black box techniques helps to find the most of these logical vulnerabilities which lead to denial of service.
Although in this process 9 vulnerabilities were found, there can be more vulnerabilities according to the attack trees.

Appendix

A - Information of YaFtp application:

Name	YaFtp (Yet another FTP)
Version	1.0.14
Programmer	Jean-Yves
Homepage	http://sf.net/projects/YaFtp
License	GNU

B – Links of some vulnerabilities in some other client FTP applications:

- In SecurityFocus.com:
URL:
http://search.securityfocus.com/swsearch?query=ftp+client&sbm=bid&submit=Search!&metaname=alldoc&sort=swishlastmodified

1. Google Chrome FTP Client PASV Port Scan Information Disclosure Vulnerability
URL: http://www.securityfocus.com/bid/33112/info
2. BulletProof FTP Client Bookmark File Heap Buffer Overflow Vulnerability (Vulnerabilities)
URL: http://www.securityfocus.com/bid/33007
3. Multiple Vendor Web Browser FTP Client Cross Site Scripting Vulnerability (Vulnerabilities)
URL: http://www.securityfocus.com/bid/31855
4. Ipswitch WS_FTP Client Format String Vulnerability (Vulnerabilities)
URL: http://www.securityfocus.com/bid/30720
5. WISE-FTP FTP Client 'LIST' Command Directory Traversal Vulnerability (Vulnerabilities)
URL: http://www.securityfocus.com/bid/29844
6. net2ftp FTP Client Request Handling Unspecified Security Vulnerability (Vulnerabilities)
URL: http://www.securityfocus.com/bid/29664
7. ALFTP FTP Client 'LIST' Command Directory Traversal Vulnerability (Vulnerabilities)
URL: http://www.securityfocus.com/bid/29585
8. FileZilla FTP Client Hard-Coded Cipher Key Vulnerability (Vulnerabilities)
URL: http://www.securityfocus.com/bid/14730
9. PeerFTP_5 Insecure Password Storage Vulnerability (Vulnerabilities)
URL: http://www.securityfocus.com/bid/12670
10. Junkie FTP Client Server Response Download Filename File Corruption Vulnerability (Vulnerabilities)
URL: http://www.securityfocus.com/bid/12011
11. Junkie FTP Client Server Response Download Filename Command Execution Vulnerability (Vulnerabilities)
URL: http://www.securityfocus.com/bid/11978
12. IglooFTP File Upload Insecure Temporary File Vulnerability (Vulnerabilities)
URL: http://www.securityfocus.com/bid/11961

And etc.

- In cve.mitre.org:
URL: http://www.google.com/custom?hl=en&q=client+ftp&sitesearch=cve.mitre.org

C – Directory Structure:

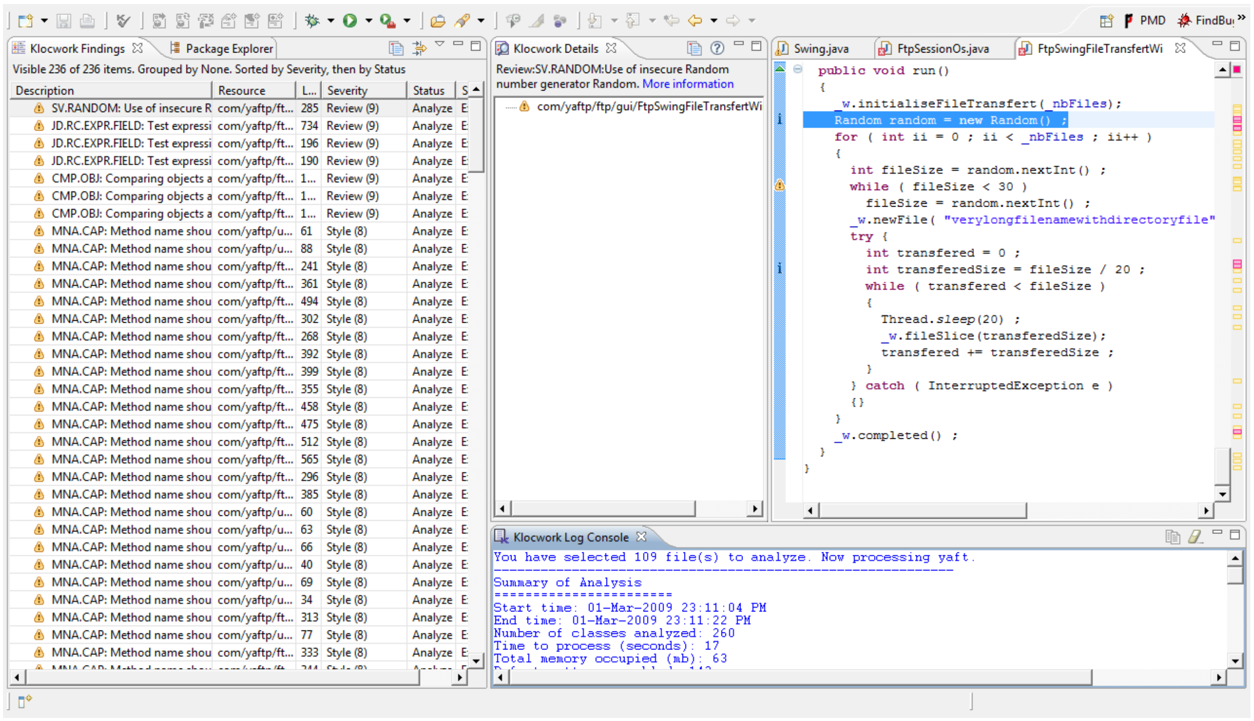
In executable mode (not source code):

/doc (contains document of the program)

/init (contains settings and important files of the program)

/lib (contains class files and jar file)

D – Automation Tools Pictures



Visible 236 of 236 items. Grouped by None. Sorted by Severity, then by Status

Description	Resource	L...	Severity	Status	S
SV.RANDOM: Use of insecure R	com/yaftp/ft...	285	Review (9)	Analyze	E
JD.RC.EXPR.FIELD: Test expressi	com/yaftp/ft...	734	Review (9)	Analyze	E
JD.RC.EXPR.FIELD: Test expressi	com/yaftp/ft...	196	Review (9)	Analyze	E
JD.RC.EXPR.FIELD: Test expressi	com/yaftp/ft...	190	Review (9)	Analyze	E
CMP.OBJ: Comparing objects a	com/yaftp/ft...	1...	Review (9)	Analyze	E
CMP.OBJ: Comparing objects a	com/yaftp/ft...	1...	Review (9)	Analyze	E
CMP.OBJ: Comparing objects a	com/yaftp/ft...	1...	Review (9)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/u...	61	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/u...	88	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	241	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	361	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	494	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	302	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	268	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	392	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	399	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	355	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	458	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	475	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	512	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	565	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	296	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	385	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/u...	60	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/u...	63	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/u...	66	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/u...	40	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/u...	69	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/u...	34	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	313	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/u...	77	Style (8)	Analyze	E
MNA.CAP: Method name shou	com/yaftp/ft...	333	Style (8)	Analyze	E

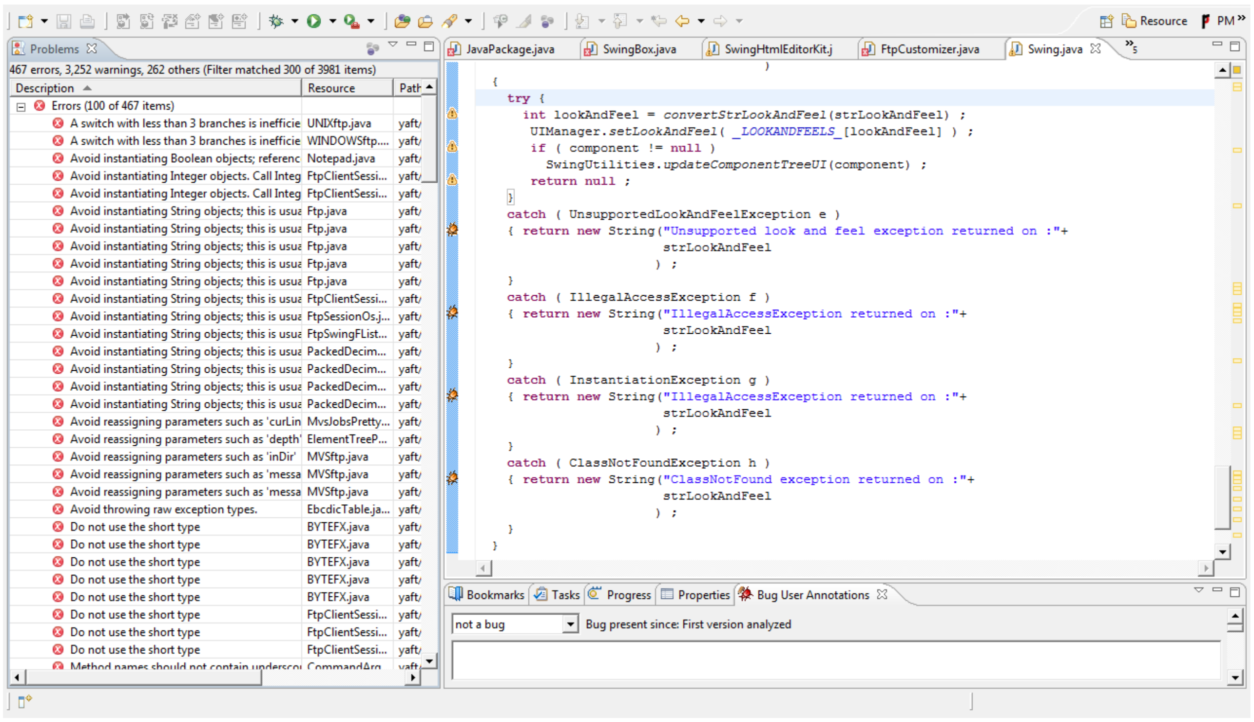
Review:SV.RANDOM:Use of insecure Random number generator Random. More information

com/yaftp/ftp/gui/FtpSwingFileTransfertWi

```
public void run()
{
  _w.initialiseFileTransfert(_nbFiles);
  Random random = new Random() ;
  for ( int ii = 0 ; ii < _nbFiles ; ii++ )
  {
    int fileSize = random.nextInt() ;
    while ( fileSize < 30 )
      fileSize = random.nextInt() ;
    _w.newFile( "verylongfilenamewithdirectoryfile"
    try {
      int transfered = 0 ;
      int transferedSize = fileSize / 20 ;
      while ( transfered < fileSize )
      {
        Thread.sleep(20) ;
        _w.fileSlice(transferedSize);
        transfered += transferedSize ;
      }
    } catch ( InterruptedException e )
    {}
  }
  _w.completed() ;
 }
}
```

```
You have selected 109 file(s) to analyze. Now processing yaft.
------------------------------------------------------------
Summary of Analysis
=======================
Start time: 01-Mar-2009 23:11:04 PM
End time: 01-Mar-2009 23:11:22 PM
Number of classes analyzed: 260
Time to process (seconds): 17
Total memory occupied (mb): 63
```

Figure D.1 - Klocwork



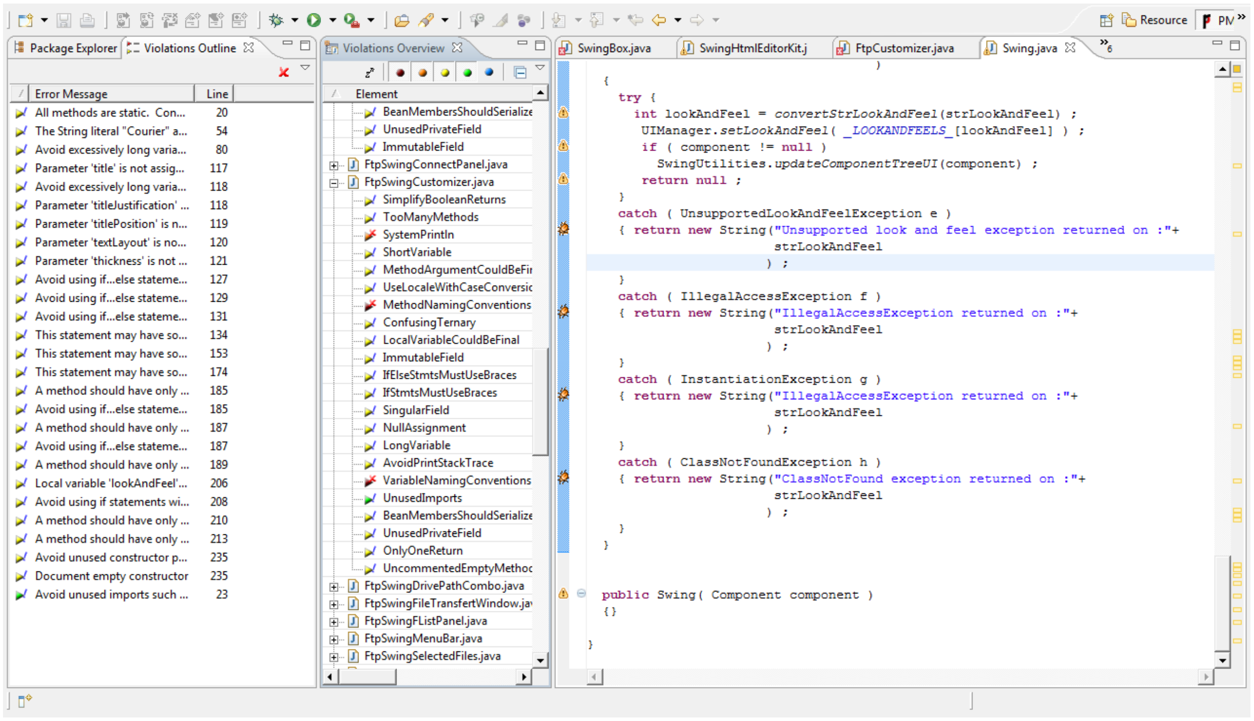
467 errors, 3,252 warnings, 262 others (Filter matched 300 of 3981 items)

Description	Resource	Path
Errors (100 of 467 items)		
A switch with less than 3 branches is inefficie	UNIXftp.java	yaft/
A switch with less than 3 branches is inefficie	WINDOWSftp....	yaft/
Avoid instantiating Boolean objects; referenc	Notepad.java	yaft/
Avoid instantiating Integer objects. Call Integ	FtpClientSessi...	yaft/
Avoid instantiating Integer objects. Call Integ	FtpClientSessi...	yaft/
Avoid instantiating String objects; this is usua	Ftp.java	yaft/
Avoid instantiating String objects; this is usua	Ftp.java	yaft/
Avoid instantiating String objects; this is usua	Ftp.java	yaft/
Avoid instantiating String objects; this is usua	Ftp.java	yaft/
Avoid instantiating String objects; this is usua	Ftp.java	yaft/
Avoid instantiating String objects; this is usua	FtpClientSessi...	yaft/
Avoid instantiating String objects; this is usua	FtpSessionOs.j...	yaft/
Avoid instantiating String objects; this is usua	FtpSwingFList...	yaft/
Avoid instantiating String objects; this is usua	PackedDecim...	yaft/
Avoid instantiating String objects; this is usua	PackedDecim...	yaft/
Avoid instantiating String objects; this is usua	PackedDecim...	yaft/
Avoid instantiating String objects; this is usua	PackedDecim...	yaft/
Avoid reassigning parameters such as 'curLin	MvsJobsPretty...	yaft/
Avoid reassigning parameters such as 'depth'	ElementTreeP...	yaft/
Avoid reassigning parameters such as 'inDir'	MVSftp.java	yaft/
Avoid reassigning parameters such as 'messa	MVSftp.java	yaft/
Avoid reassigning parameters such as 'messa	MVSftp.java	yaft/
Avoid throwing raw exception types.	EbcdicTable.ja...	yaft/
Do not use the short type	BYTEFX.java	yaft/
Do not use the short type	BYTEFX.java	yaft/
Do not use the short type	BYTEFX.java	yaft/
Do not use the short type	BYTEFX.java	yaft/
Do not use the short type	BYTEFX.java	yaft/
Do not use the short type	FtpClientSessi...	yaft/
Do not use the short type	FtpClientSessi...	yaft/
Do not use the short type	FtpClientSessi...	yaft/
Method names should not contain undersco	CommandArg	yaft/

```
                                 )
    {
      try {
        int lookAndFeel = convertStrLookAndFeel(strLookAndFeel) ;
         UIManager.setLookAndFeel( _LOOKANDFEELS_[lookAndFeel] ) ;
         if ( component != null )
           SwingUtilities.updateComponentTreeUI(component) ;
         return null ;
      }
      catch ( UnsupportedLookAndFeelException e )
      { return new String("Unsupported look and feel exception returned on :"+
                          strLookAndFeel
                         ) ;
      }
      catch ( IllegalAccessException f )
      { return new String("IllegalAccessException returned on :"+
                          strLookAndFeel
                         ) ;
      }
      catch ( InstantiationException g )
      { return new String("IllegalAccessException returned on :"+
                          strLookAndFeel
                         ) ;
      }
      catch ( ClassNotFoundException h )
      { return new String("ClassNotFound exception returned on :"+
                          strLookAndFeel
                         ) ;
      }
    }
```

not a bug   Bug present since: First version analyzed

Figure D.2 - FindBugs



Error Message	Line
All methods are static. Con...	20
The String literal "Courier" a...	54
Avoid excessively long varia...	80
Parameter 'title' is not assig...	117
Avoid excessively long varia...	118
Parameter 'titleJustification' ...	118
Parameter 'titlePosition' is n...	119
Parameter 'textLayout' is no...	120
Parameter 'thickness' is not ...	121
Avoid using if...else stateme...	127
Avoid using if...else stateme...	129
Avoid using if...else stateme...	131
This statement may have so...	134
This statement may have so...	153
This statement may have so...	174
A method should have only ...	185
Avoid using if...else stateme...	185
A method should have only ...	187
Avoid using if...else stateme...	187
A method should have only ...	189
Local variable 'lookAndFeel'...	206
Avoid using if statements wi...	208
A method should have only ...	210
A method should have only ...	213
Avoid unused constructor p...	235
Document empty constructor	235
Avoid unused imports such ...	23

```
Element
    BeanMembersShouldSerialize
    UnusedPrivateField
    ImmutableField
+ FtpSwingConnectPanel.java
- FtpSwingCustomizer.java
    SimplifyBooleanReturns
    TooManyMethods
    SystemPrintln
    ShortVariable
    MethodArgumentCouldBeFi
    UseLocaleWithCaseConversic
    MethodNamingConventions
    ConfusingTernary
    LocalVariableCouldBeFinal
    ImmutableField
    IfElseStmtsMustUseBraces
    IfStmtsMustUseBraces
    SingularField
    NullAssignment
    LongVariable
    AvoidPrintStackTrace
    VariableNamingConventions
    UnusedImports
    BeanMembersShouldSerialize
    UnusedPrivateField
    OnlyOneReturn
    UncommentedEmptyMethoc
+ FtpSwingDrivePathCombo.java
+ FtpSwingFileTransfertWindow.ja
+ FtpSwingFListPanel.java
+ FtpSwingMenuBar.java
+ FtpSwingSelectedFiles.java
```

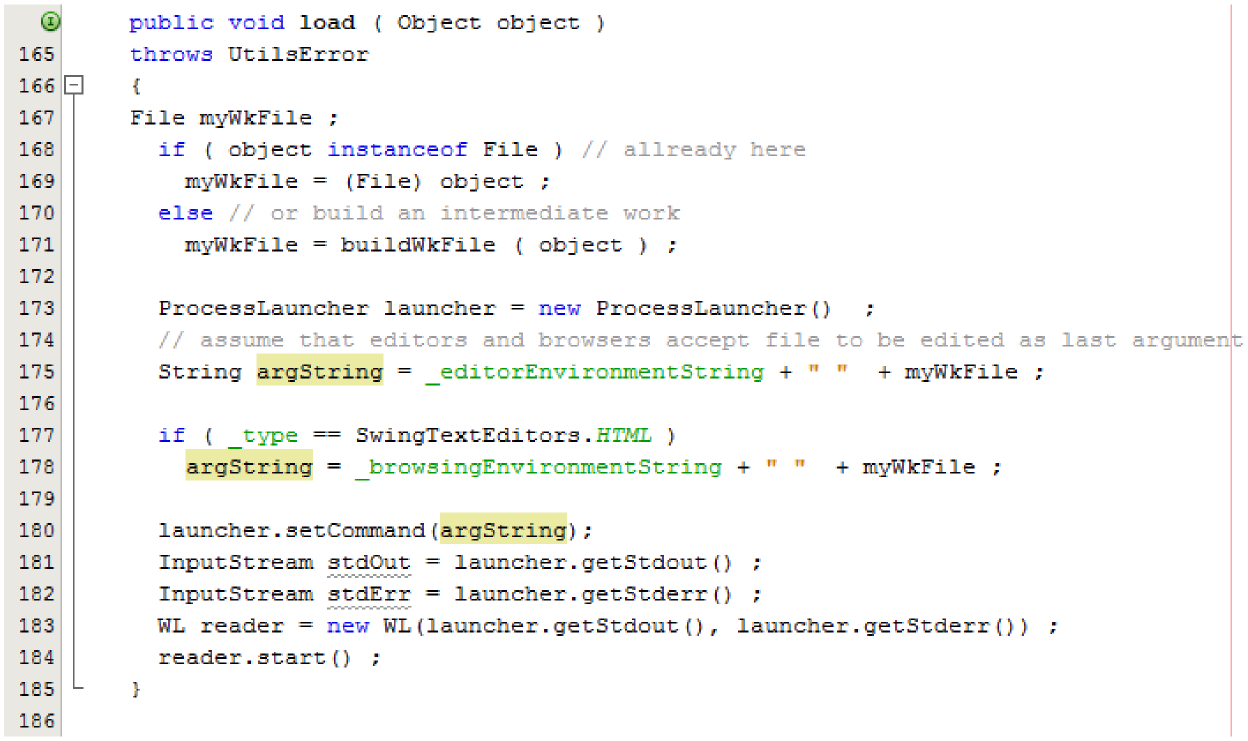
```
                                 )
    {
      try {
        int lookAndFeel = convertStrLookAndFeel(strLookAndFeel) ;
         UIManager.setLookAndFeel( _LOOKANDFEELS_[lookAndFeel] ) ;
         if ( component != null )
           SwingUtilities.updateComponentTreeUI(component) ;
         return null ;
      }
      catch ( UnsupportedLookAndFeelException e )
      { return new String("Unsupported look and feel exception returned on :"+
                          strLookAndFeel
                         ) ;
      }
      catch ( IllegalAccessException f )
      { return new String("IllegalAccessException returned on :"+
                          strLookAndFeel
                         ) ;
      }
      catch ( InstantiationException g )
      { return new String("IllegalAccessException returned on :"+
                          strLookAndFeel
                         ) ;
      }
      catch ( ClassNotFoundException h )
      { return new String("ClassNotFound exception returned on :"+
                          strLookAndFeel
                         ) ;
      }
    }


  public Swing( Component component )
  {}

}
```

Figure D.3 - PMD

E – YaFtp Codes:



```
public void load ( Object object )
throws UtilsError
{
File myWkFile ;
  if ( object instanceof File ) // allready here
    myWkFile = (File) object ;
  else // or build an intermediate work
    myWkFile = buildWkFile ( object ) ;

  ProcessLauncher launcher = new ProcessLauncher()  ;
  // assume that editors and browsers accept file to be edited as last argument
  String argString = _editorEnvironmentString + " "  + myWkFile ;

  if ( _type == SwingTextEditors.HTML )
    argString = _browsingEnvironmentString + " "  + myWkFile ;

  launcher.setCommand(argString);
  InputStream stdOut = launcher.getStdout() ;
  InputStream stdErr = launcher.getStderr() ;
  WL reader = new WL(launcher.getStdout(), launcher.getStderr()) ;
  reader.start() ;
}
```

Figure E.1 – "OsEditorPanePanel" which call "setCommand" method

References

[1] Mark Dowd, John McDonald, and Justin Schuh. The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities. Addison Wesley; 2006.
[2] J. Postel, J. Reynolds. RFC959 - FILE TRANSFER PROTOCOL (FTP). ISI; October 1985.
[3] M. Allman, S. Ostermann. RFC2577 - FTP Security Considerations; May 1999.
[4] M. Horowitz, S. Lunt. RFC2228 - FTP Security Extensions; October 1997.
[5] Joseph A. Bank. Java Security; December 1995.
[6] Sun Microsystems, Inc. Secure Coding Guidelines for the Java Programming Language, version 2.0; 2007.
[7] File Transfer Protocol. Wikipedia (http://en.wikipedia.org/wiki/File_Transfer_Protocol).
[8] FTP Sequence Diagram. EventHelix co. (http://www.eventhelix.com/RealtimeMantra/Networking/FTP.pdf).
[9] Chris Grant. FTP Reviewed; 1998.
[10] List of FTP commands. Wikipedia (http://en.wikipedia.org/wiki/List_of_FTP_commands).
[11] Ryan Heise. jftpd - 0.3. (http://www.ryanheise.com/software/jftpd/)
[12] Klocwork - Static source code analysis. (http://www.klocwork.com/)
[13] Find Bugs - Static Java source code analysis. (http://findbugs.sourceforge.net/)
[14] PMD - Static Java source code analysis. (http://pmd.sourceforge.net/)