UNIVERSITÉ Concordia UNIVERSITY

TOOLS3: Revival of the Ftklipse project with MARFCAT and MARFPCAT plug-ins, and possibly distributed system evaluation integration

Abdulazeez A Olukoso 40111182
Alavi Tabassum Sristi 40104468
Daniel(Aliasghar Zakerifar40054463
Mohammed Moazzem Hossain Khan 40104656
Omar Bakhache 40157025

August 2020

Contents

List of Figures

A Abstract

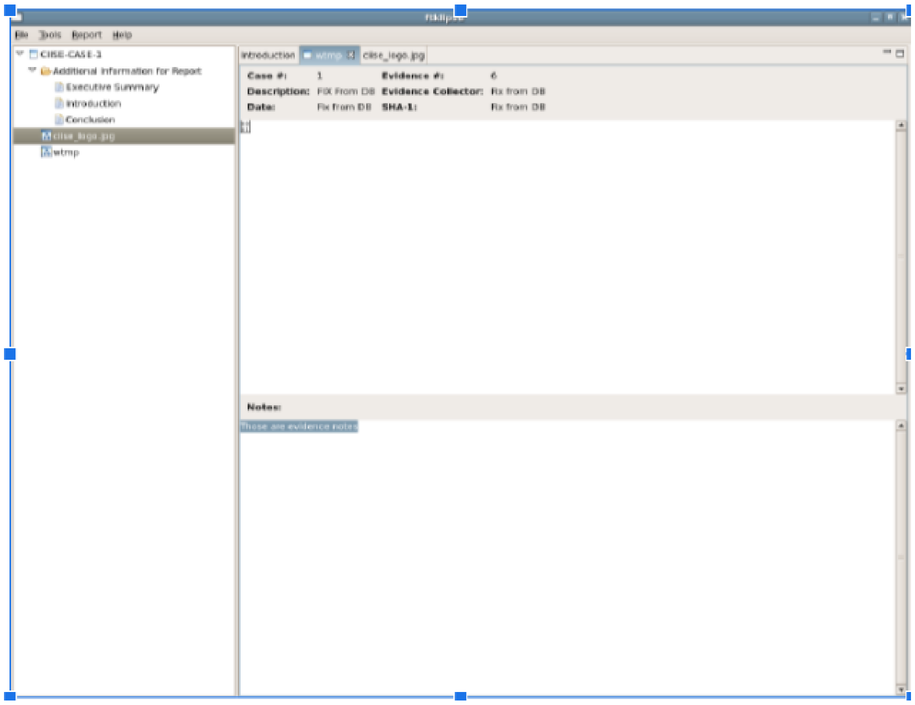
As the cyber world evolves daily with technological advancements to fix traditional issues, so do the threats that come with it. However, when it gets to the point of critically investigating the crimes being committed, investigators sort out for the best computer forensic tools to allow them go through with each case. Having in mind the steps of investigations going from gathering evidences to generating reports, Ftklipse as a forensic tools helps to make their work easier in a cost effective manner. After revival of this project, we further added tools to help investigators widen their reach into proper evidence analysis.

B Introduction

The internet is an interconnection of computer networks, connecting multiple computing devices and networks for communication[6]. It is used to perform activities in business, communications and information interchange globally. However this interconnection and wide range of application has given room for networks and personal computers to be susceptible to threats and attacks from cyber-criminals.
Computer crimes also referred to as cyber-crime are criminal acts committed through digital knowledge of offenders stored in their computer systems. This act comes in the form of trade secrets theft, theft of or destruction of intellectual property, cyber obscenity in form of child pornography and fraud. In turn, the criminal activities being performed by these criminals give room for them to gain access to enterprise information systems, this is in turn exploited for financial gain[2].
Cyber-criminals get involved with one or all of these crimes. They do so by discovering vulnerabilities in victims operating systems, applications or services that are running on a computer that is connected to the internet [cite here]. These vulnerabilities are exploited to allow the criminal view and store sensitive information on any storage media like hard-drives or removable floppy disks, zip drives, memory sticks or CDs [2]. At this point where the crime has been committed, there is a call for legal intervention to prevent them from causing further harm and their apprehension. Therefore, these storage mediums can be used as evidence against them. As such, computer forensic specialists (CFS) are tasked to investigate the digital crime scene by unprejudiced scrutinizing of digital evidence that is either involved or thought to be involved in the crime. At the end of the investigation, they ultimately produce a single document of report highlighting a summary of the contents of the pieces of evidence.
Investigating this computer-based crime involves a specialization called forensic computing. However, the process of computer investigation and analysis technique to gather evidence in a manner that is legally acceptable sometimes is a hectic process, hence this is where the Ftklipse tool comes in.

Ftklipse is a java based tool that implements graphical user interfaces in its evidence-center for forensics investigation. It is used by investigators to collect and preserve evidence, analyze it, record notes for each piece of evidence, record additional reporting information, and report on it[7].

It allows the implementation of other configurable tools and plugins which in turn assists to facilitate and accelerate the investigation process. In terms of hardware, any computer that can run the Eclipse IDE can be used to run Ftklipse, however, in terms of software, the investigators' computer must support JVM version 5 or higher LATEX2e but preferably pdflatex[18]see fig 1.



Case #:	1	Evidence #:	6
Description:	FIX From DB	Evidence Collector:	Fix from DB
Date:	Fix from DB	SHA-1:	Fix from DB

These are evidence notes

Figure 1: Ftklipse GUI Showing the Evidence Information and Notes

B.1 Goal and Methodology

The goal of every good forensic tool is to maintain integrity and credibility at a cost effective manner. Ftklipse like any other forensic tool follows the same methodology for their investigations. This methodology includes [2]:

- Protecting the confististicated computer system during the forensic examination from any possible alteration, damage, data corruption or virus infection.
- Discovering all files on the confiscated computer system and storage mediums which includes existing normal files, deleted yet remaining files, hidden files, password-protected files and encrypted files.

- Recovering of files that are discovered to be deleted to the best extent possible.
- Revealing the contents of hidden files as well as temporary files used by application programs and the operating system.
- Accessing all files either protected or hidden if possible and legally appropriate.
- Analyzing all relevant data found in special areas of the disk.
- Printing of an overall analysis of the confisticated computer system.

B.2 Features of Ftklipse and workability

- Creation of cases: This is a critical feature of the software that lets Ftklipse create cases with their associated metadata. No technical issues are recorded here as well as no dependencies with Other Requirements
- Evidence gathering: This is another critical feature that allows multiple tools to run simultaneously to perform evidence collection on a live system. In terms of technicalities, only the useful aspects of investigations should be considered for the redirection of the tool's standard input and output.
- Evidence import: This feature lets investigators import all pieces of evidence gathered and it has to be accompanied by Sha-1 digest.
- Plugin integration: This feature works with evidence gathering, evidence importing, and hash function for its integrity as specialized tools are used to perform these tasks.
- Evidence analysis: This feature lets investigators perform a critical analysis of the evidence gathered.
- Evidence Integrity Validation: SHA-1 signature of every piece of evidence is recorded on this software to ascertain that the evidence is kept correct and undistorted during the investigation.
- Evidence Display - This lets the pieces of evidence be displayed visually in read-only mode either in ASCII, Unicode, Hex formats and/or Forensic lucid.
- Logging: All operations performed can be logged on this software. Also, all operations related to a given piece of evidence are logged for that evidence specifically.
- Evidence Extraction: During a case, this feature lets the investigator extract a specific piece of evidence to aid in presenting his case.
- Evidence Cloning: This feature lets investigators make a copy of any piece of evidence that had been gotten and logged[7].

Other features offered by this software are but not limited to Report Generation, Database management system for storage and retrieval of all pieces of evidence, Access Control Management to view, read , write to evidence and information tracking on the database.

B.3 MARFCAT and MARFPCAT plugins and Forensic Lucid

MARF is an asynchronous framework that consists of a collection of algorithms for audio recognition and natural language text analysis implemented in Java. It was developed with the ability to subsist with manipulated code or codes already in use by other projects [11]. Similarly, MARFCAT is a tool that is utilized for quick static code analysis for looking through defects and vulnerabilities identified with security, programming development and others utilizing the open source MARF structure. MARFCAT doesn't rely on the language utilized for investigation, source code and binary[12].

B.3.1 Limitation of MARFCAT and MARFPCAT

MARFCAT has no path tracing because it has multiple algorithms with different applications and the existence of those applications brings about complexity in the framework.

B.3.2 Forensic lucid

Lucis is a purely declarative programming language which is problem oriented. It is based on intensional logic. This logic mainly deals with assertions and expressions whose meanings depend on an implicit/unclear context. Natural language proliferate with these assertions as we use natural language all the time to solve problems[3][4]. The goal of Forensic Lucid in the cyber forensic analysis is to be enable investigators express the encoding of the evidence in a program form, witness stories, as well as evidential statements, that can be tested against claims. This lets them see any possible sequence or multiple sequences of events that make this more explicit enough for a given story. The implementing system (i.e. GIPSY) has to backtrace intermediate results to provide an existing corresponding event reconstruction path. The result of the expression in its basic form is either true or false based on context per explanation with the backtrace. There can be multiple backtraces, that correspond to the explanation of the evidence (or lack thereof)[10].GIPSY provides an integrated framework for compiling programs written in theoretically all variants of Lucid[8].

C Background

In order to further understand Ftklipse better, we need to include a review of the existing area of research leading up to the existing system. This includes the use case and architecture that brought about this tool. Ftklipse is a thick-client solution for forensics investigation. It allows us to collect and preserve evidence, to analyze it and to report on it.

- a singular product depending on a variety of standard tools organized as plug-ins
- extendable using plug-ins that will add evidence gathering and analysis properties [7]

The use case model for Ftklipse is illustrated in the fig2:

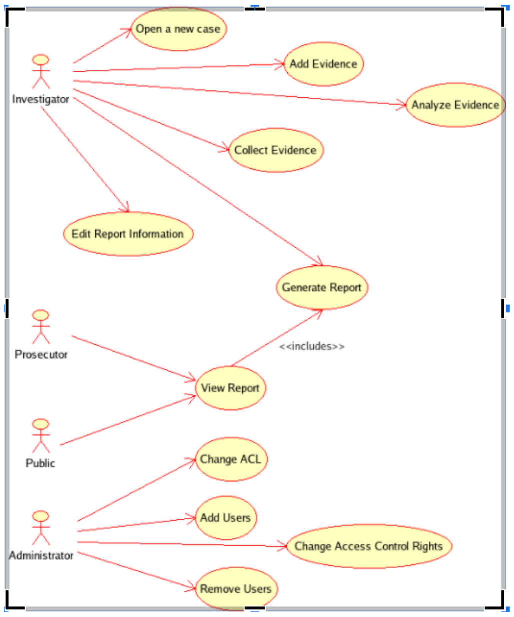


Figure 2: Use Case Diagram for Ftklipse

A user interface need identified itself to aid visualize forensic cases, digital evidence, as well as related specification components .However this revolves around providing usability enhancements to aid the investigators.
Ftklipse allows investigators to run different tools in order to perform evidence collection on a live system. MARF and MARFCAT can be integrated with the Forensic Lucic model and GIPSY. The overall deployment architecture is as fig3

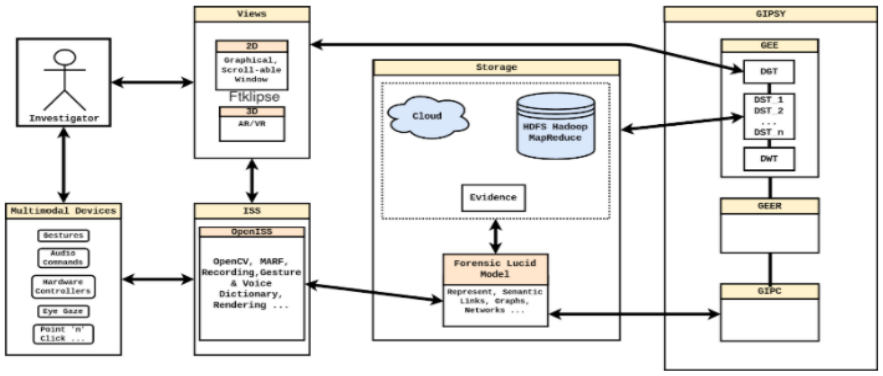


Figure 3: Development Architecture for Ftklipse

C.1 Related Tools

- Access Data Forensic Toolkit-It is used to scan hard drives to look for information like deleted texts, emails and other information. The toolkit also includes FTK Imager that previews and images data for electronic evidence and validates any further investigation needed. FTK Imager confirms the integrity of the data and saves the result in various formats. FTK Imager can also create perfect copies (forensic images) of computer data without making changes to the original evidence[5].

- Autopsy used with the Sleuth Kit- Autopsy is a digital forensics platform that provides the GUI to many forensic tools including the Sleuth Kit. Autopsy helps analyze hard drives and smartphones while the Sleuth Kit allows us to analyze disk images and recover files from them. Since Autopsy has a plug-in architecture, it allows the Sleuth Kit to be run in the background as well. This tool is more expensive and tasking to maintain as compared to the FTklipse tool[1][5].

- Bulk Extractor- This is another tool that scans a disk image, file, or directory of files to extract information like credit card numbers, domains, e-mail addresses, URLs, and ZIP files. The output of the extracted information is a series of text files that is further analyzed either manually or by using more forensic tools[9].

D Ftklipse tool

D.1 Functionality requirements

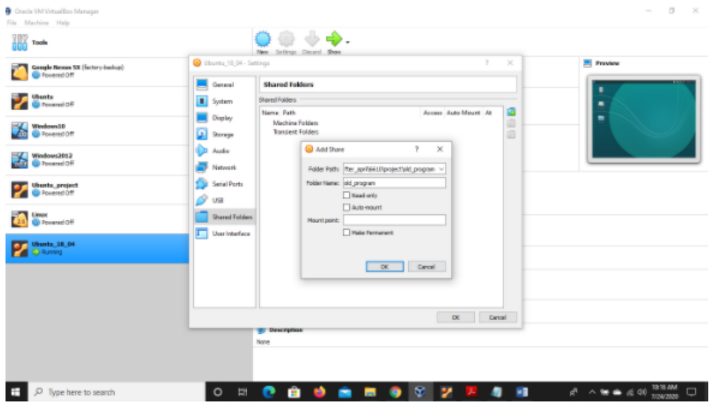
In order to successfully run the Ftklipse tool, we require the below software requirements.

- Operating System - Linux (Xubuntu-18.04-desktop (32 bit))
- JVM - Java for linux (jdk-7u80-linux-i586 (32 bit))
- Java IDE - Eclipse

D.2 Steps to Implementing Ftklipse source code

Stage 1 - Virtual Box, Operating system and Java Installation

1. Install Ubuntu Xubuntu-18.04-desktop-i386.iso (32 bit)
2. Shared folder from virtual box see fig 4



Shared Folders

Folder Path: fter_aprl6610\project\old_program

Folder Name: old_program

Read-only [ ]

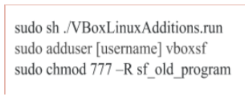
Auto-mount [ ]

Mount point:

Make Permanent [ ]

Figure 4: Virtual Box

3. Perform the following activity from Ubuntu machine
   Go to the /media/sf_old _program and run prompt a in fig 6



```
sudo sh ./VBoxLinuxAdditions.run
sudo adduser [username] vboxsf
sudo chmod 777 –R sf_old_program
```

Figure 6: prompt a

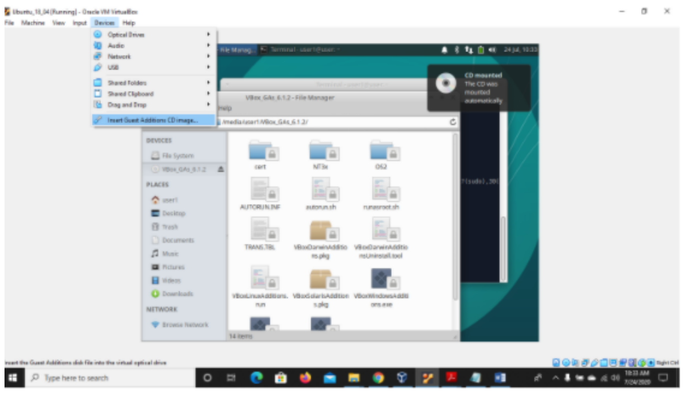
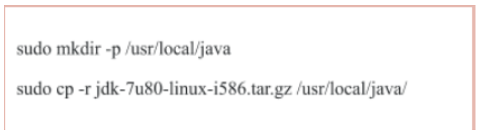


Figure 5: Ubuntu machine

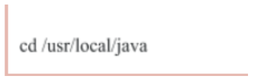
4. Install java:jdk-7u80-linux-i586.tar.gz (32 bit)
5. Create a directory in /usr/local where java will reside and copy tarball there see fig 7



```
sudo mkdir -p /usr/local/java
sudo cp -r jdk-7u80-linux-i586.tar.gz /usr/local/java/
```

Figure 7: prompt b

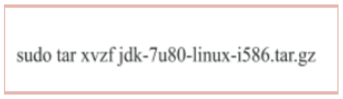
6. Navigate to /usr/local/java with command in fig8



```
cd /usr/local/java
```

Figure 8: prompt c

7. Extract the tarball see fig 9

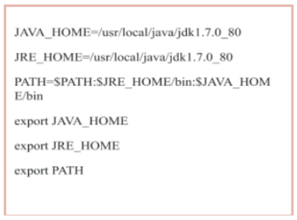


```
sudo tar xvzf jdk-7u80-linux-i586.tar.gz
```

Figure 9: prompt d

8. Confirm that tarball has been successfully extracted

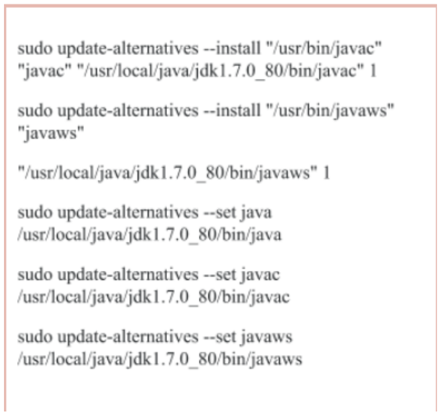
9. Open /etc/profile with Sudo privileges with command Sudo nano /etc/profile
10. Scroll down to the end of the file using arrow keys and add the following lines below to the end of /etc/profile file: see fig 10



```
JAVA_HOME=/usr/local/java/jdk1.7.0_80
JRE_HOME=/usr/local/java/jdk1.7.0_80
PATH=$PATH:$JRE_HOME/bin:$JAVA_HOM
E/bin
export JAVA_HOME
export JRE_HOME
export PATH
```

Figure 10: prompt e

11. Update alternatives see fig 11



```
sudo update-alternatives --install "/usr/bin/javac"
"javac" "/usr/local/java/jdk1.7.0_80/bin/javac" 1

sudo update-alternatives --install "/usr/bin/javaws"
"javaws"

"/usr/local/java/jdk1.7.0_80/bin/javaws" 1

sudo update-alternatives --set java
/usr/local/java/jdk1.7.0_80/bin/java

sudo update-alternatives --set javac
/usr/local/java/jdk1.7.0_80/bin/javac

sudo update-alternatives --set javaws
/usr/local/java/jdk1.7.0_80/bin/javaws
```

Figure 11: prompt f

12. Reload profile: source /etc/profile
13. Verify installation java version
    You should receive a message which displays:
    java version ”1.7.0_80”
    Java(TM) SE Runtime Environment (build 1.7.0_80-b15)
    Java HotSpot(TM) 64-Bit Server VM (build 24.80-b11, mixed mode

Stage 2 - Eclipse IDE Installation and running the Ftklipse src code

1. Install eclipse ”eclipse-jee-kepler-SR2-linux-gtk.tar.gz” within ubuntu (32 bit)

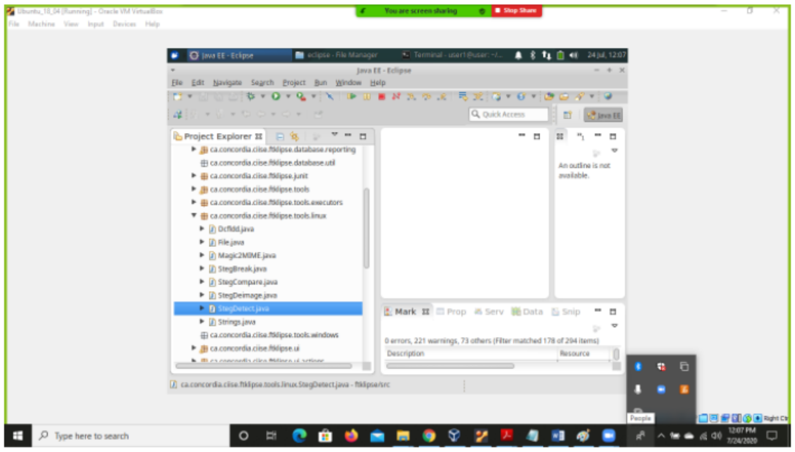
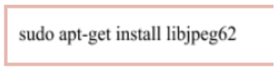

Java EE - Eclipse
File Edit Navigate Search Project Run Window Help
Project Explorer
ca.concordia.ciise.ftklipse.database.reporting
ca.concordia.ciise.ftklipse.database.util
ca.concordia.ciise.ftklipse.junit
ca.concordia.ciise.ftklipse.tools
ca.concordia.ciise.ftklipse.tools.executors
ca.concordia.ciise.ftklipse.tools.linux
Dcfldd.java
File.java
Magic2MIME.java
StegBreak.java
StegCompare.java
StegDeimage.java
StegDetect.java
Strings.java
ca.concordia.ciise.ftklipse.tools.windows
ca.concordia.ciise.ftklipse.ui
An outline is not available.
0 errors, 221 warnings, 73 others (Filter matched 178 of 294 items)
ca.concordia.ciise.ftklipse.tools.linux.StegDetect.java - ftklipse/src
Type here to search

Figure 12: Eclipse IDE

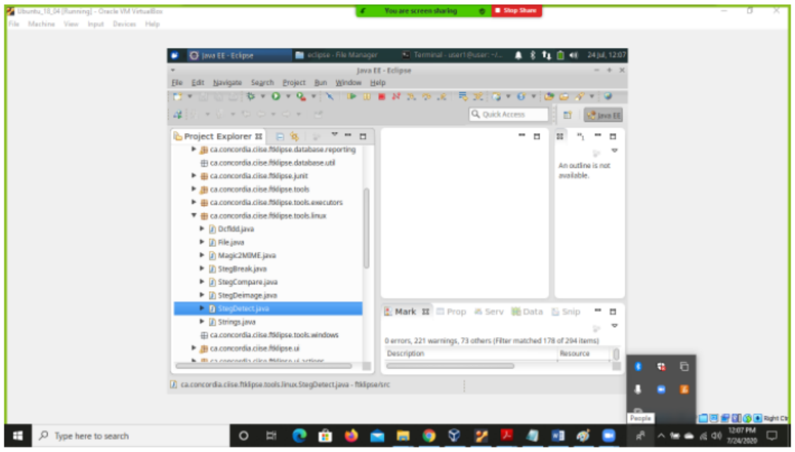
2. Import "ciisesec-code-r398-forensics.zip" within eclipse
3. Import "ciisesec-code-r398-forensics.zip" within eclipse see fig 13



```
sudo apt-get install libjpeg62
```

Figure 13: prompt g

4. Changes in the "StegDetect.java" and "Tool.java" according to the docutment give execute permission to /home/user1/workspace/ftklipse/tools/linux



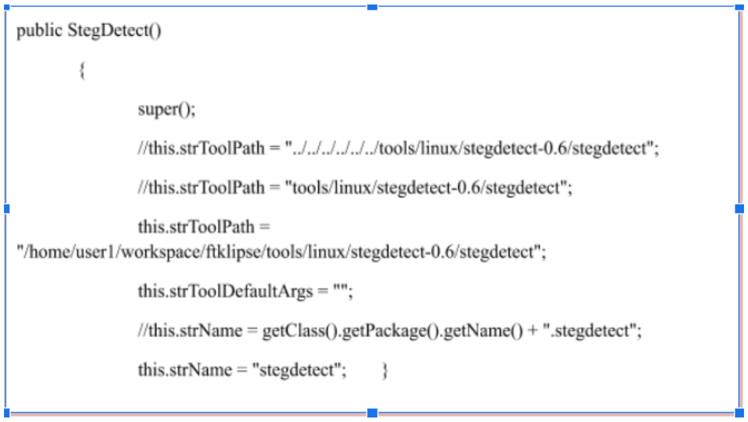
```
▸ ca.concordia.ciise.ftklipse.database.reporting
  ca.concordia.ciise.ftklipse.database.util
▸ ca.concordia.ciise.ftklipse.junit
▸ ca.concordia.ciise.ftklipse.tools
▸ ca.concordia.ciise.ftklipse.tools.executors
▾ ca.concordia.ciise.ftklipse.tools.linux
    ▸ Dcfldd.java
    ▸ File.java
    ▸ Magic2MIME.java
    ▸ StegBreak.java
    ▸ StegCompare.java
    ▸ StegDeimage.java
    ▸ StegDetect.java
    ▸ Strings.java
  ca.concordia.ciise.ftklipse.tools.windows
▸ ca.concordia.ciise.ftklipse.ui
```

```
0 errors, 221 warnings, 73 others (Filter matched 178 of 294 items)
```

```
ca.concordia.ciise.ftklipse.tools.linux.StegDetect.java - ftklipse/src
```

Figure 14: Eclipse IDE

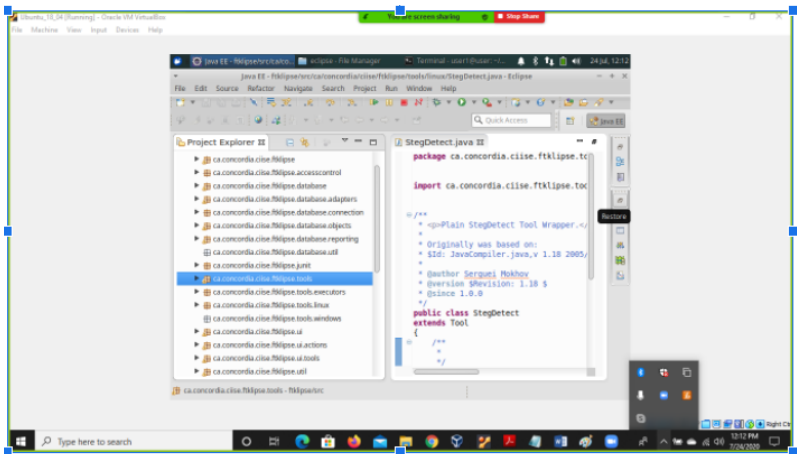
5. Edit the line in ca.concordia.ciise.ftklipse.tools.linux as follows see fig 15



```
public StegDetect()
        {
                super();
                //this.strToolPath = "../../../../../../tools/linux/stegdetect-0.6/stegdetect";
                //this.strToolPath = "tools/linux/stegdetect-0.6/stegdetect";
                this.strToolPath =
"/home/user1/workspace/ftklipse/tools/linux/stegdetect-0.6/stegdetect";
                this.strToolDefaultArgs = "";
                //this.strName = getClass().getPackage().getName() + ".stegdetect";
                this.strName = "stegdetect";        }
```

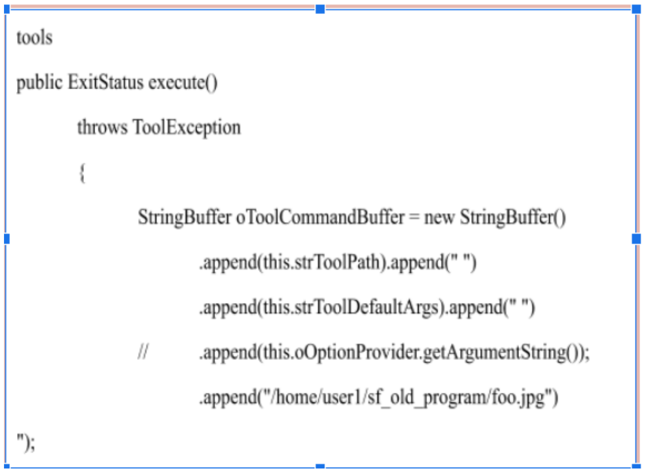
Figure 15: Eclipse IDE

6. Edit the line in ca.concordia.ciise.ftklipse.tools as follows see fig 17


Java EE - ftklipse/src/ca/concordia/ciise/ftklipse/tools/linux/StegDetect.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
ca.concordia.ciise.ftklipse
ca.concordia.ciise.ftklipse.accesscontrol
ca.concordia.ciise.ftklipse.database
ca.concordia.ciise.ftklipse.database.adapters
ca.concordia.ciise.ftklipse.database.connection
ca.concordia.ciise.ftklipse.database.objects
ca.concordia.ciise.ftklipse.database.reporting
ca.concordia.ciise.ftklipse.database.util
ca.concordia.ciise.ftklipse.junit
ca.concordia.ciise.ftklipse.tools
ca.concordia.ciise.ftklipse.tools.executors
ca.concordia.ciise.ftklipse.tools.linux
ca.concordia.ciise.ftklipse.tools.windows
ca.concordia.ciise.ftklipse.ui
ca.concordia.ciise.ftklipse.ui.actions
ca.concordia.ciise.ftklipse.ui.tools
ca.concordia.ciise.ftklipse.util
ca.concordia.ciise.ftklipse.tools - ftklipse/src
StegDetect.java
Restore
Type here to search

```
package ca.concordia.ciise.ftklipse.to

import ca.concordia.ciise.ftklipse.too

/**
 * <p>Plain StegDetect Tool Wrapper.</
 *
 * Originally was based on:
 * $Id: JavaCompiler.java,v 1.18 2005/
 *
 * @author Serguei Mokhov
 * @version $Revision: 1.18 $
 * @since 1.0.0
 */
public class StegDetect
extends Tool
{
	/**
	 *
	 */
```

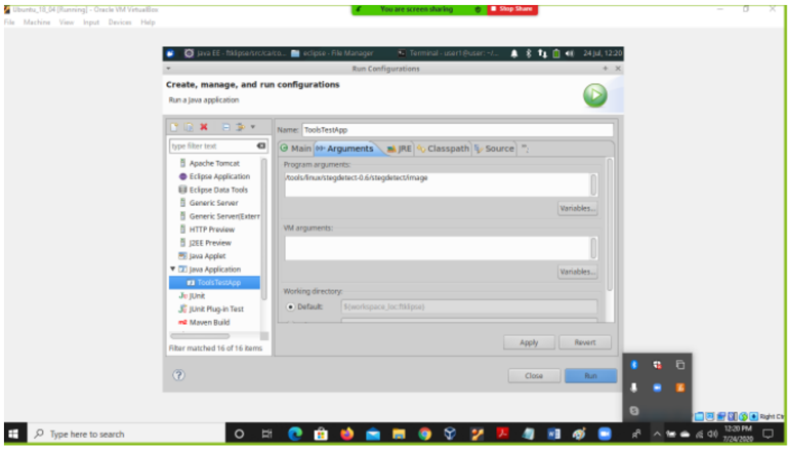
Figure 16: Ftklipse source code on Eclipse



```
tools
public ExitStatus execute()
		throws ToolException
		{
			StringBuffer oToolCommandBuffer = new StringBuffer()
				.append(this.strToolPath).append(" ")
				.append(this.strToolDefaultArgs).append(" ")
			//	.append(this.oOptionProvider.getArgumentString());
				.append("/home/user1/sf_old_program/foo.jpg")
");
```

Figure 17:

7. Compile the projects. Clean the projects.
8. Run Java Application.
9. Run and config the argument (Run Configuration) and Put the line in Arguments /tools/linux/stgedetect6.0/stgdetect/image



Run Configurations

Create, manage, and run configurations

Run a Java application

Name: ToolsTestApp

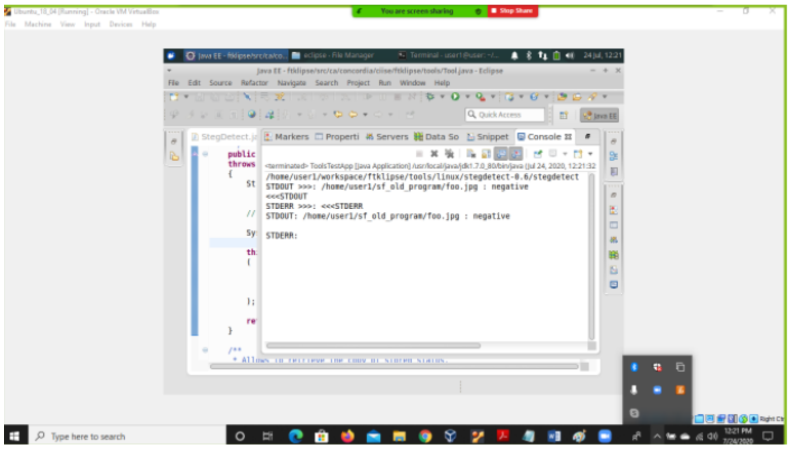
Program arguments:

```
/tools/linux/stegdetect-0.6/stegdetect/image
```

Working directory: Default: ${workspace_loc:ftklipse}

Figure 18: Ftklipse source code on Eclipse

10. Run the project



Java EE - ftklipse/src/ca/concordia/ciise/ftklipse/tools/Tool.java - Eclipse

```
<terminated> ToolsTestApp [Java Application] /usr/local/java/jdk1.7.0_80/bin/java (Jul 24, 2020, 12:21:32
/home/user1/workspace/ftklipse/tools/linux/stegdetect-0.6/stegdetect
STDOUT >>>: /home/user1/sf_old_program/foo.jpg : negative
<<<STDOUT
STDERR >>>: <<<STDERR
STDOUT: /home/user1/sf_old_program/foo.jpg : negative

STDERR:
```

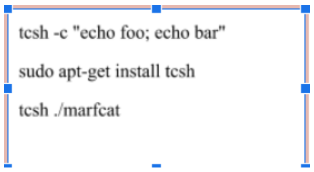
Figure 19: Eclipse IDE

D.3 Integrating MARFCAT

see fig 20

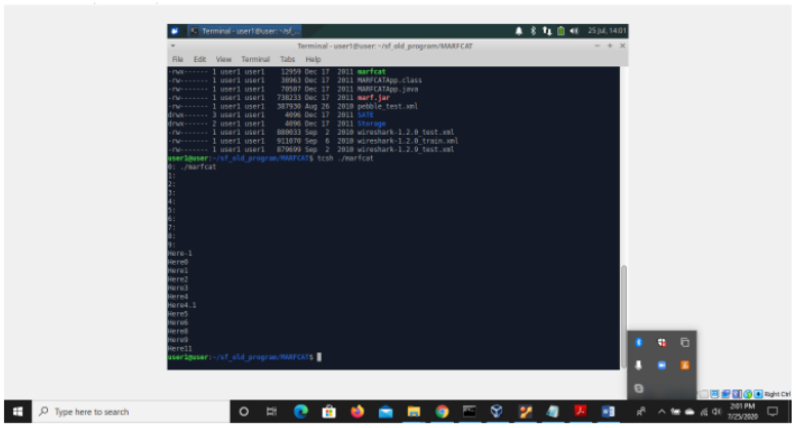
1. Download MARFCAT-0.0.1-SATE2010-Alpha3.tar.gz
   select MARFCAT-0.0.1-SATE2010-Alpha3.tar , download and unzip

2. Run the command at ubuntu prompt see fig 20



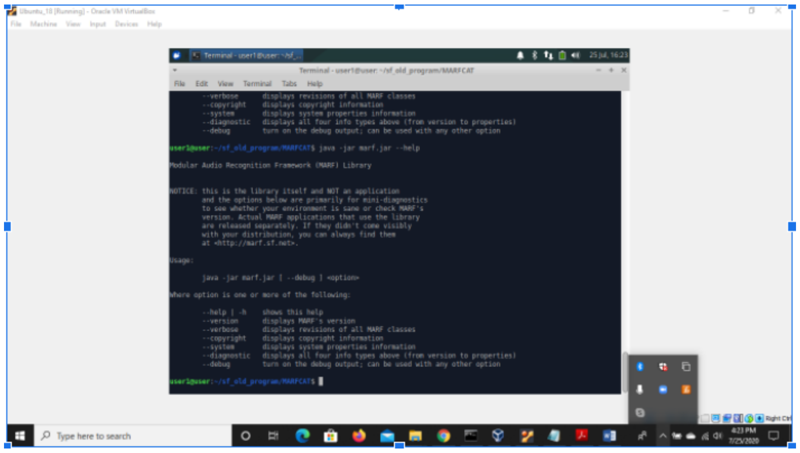
```
tcsh -c "echo foo; echo bar"
sudo apt-get install tcsh
tcsh ./marfcat
```

Figure 20: k



```
-rwx------ 1 user1 user1    12959 Dec 17  2011 marfcat
-rw------- 1 user1 user1    38963 Dec 17  2011 MARFCATApp.class
-rw------- 1 user1 user1    70507 Dec 17  2011 MARFCATApp.java
-rw------- 1 user1 user1   738233 Dec 17  2011 marf.jar
-rw------- 1 user1 user1   387930 Aug 26  2010 pebble_test.xml
drwx------ 3 user1 user1     4096 Dec 17  2011 SATE
drwx------ 2 user1 user1     4096 Dec 17  2011 Storage
-rw------- 1 user1 user1   880033 Sep  2  2010 wireshark-1.2.0_test.xml
-rw------- 1 user1 user1   911070 Sep  6  2010 wireshark-1.2.0_train.xml
-rw------- 1 user1 user1   879699 Sep  2  2010 wireshark-1.2.9_test.xml
user1@user:~/sf_old_program/MARFCAT$ tcsh ./marfcat
0: ./marfcat
1:
2:
3:
4:
5:
6:
7:
8:
9:
Here-1
Here0
Here1
Here2
Here3
Here4
Here4.1
Here5
Here6
Here8
Here9
Here11
user1@user:~/sf_old_program/MARFCAT$
```

Figure 21: Output-1

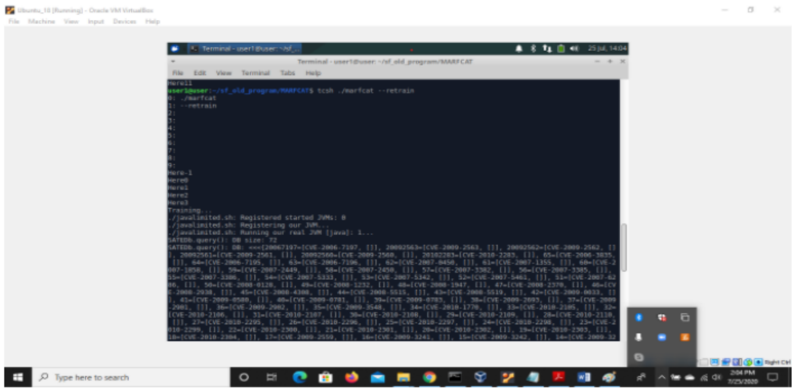


```
        --verbose       displays revisions of all MARF classes
        --copyright     displays copyright information
        --system        displays system properties information
        --diagnostic    displays all four info types above (from version to properties)
        --debug         turn on the debug output; can be used with any other option

user1@user:~/sf_old_program/MARFCAT$ java -jar marf.jar --help

Modular Audio Recognition Framework (MARF) Library


NOTICE: this is the library itself and NOT an application
        and the options below are primarily for mini-diagnostics
        to see whether your environment is sane or check MARF's
        version. Actual MARF applications that use the library
        are released separately. If they didn't come visibly
        with your distribution, you can always find them
        at <http://marf.sf.net>.

Usage:

        java -jar marf.jar [ --debug ] <option>

Where option is one or more of the following:

        --help | -h     shows this help
        --version       displays MARF's version
        --verbose       displays revisions of all MARF classes
        --copyright     displays copyright information
        --system        displays system properties information
        --diagnostic    displays all four info types above (from version to properties)
        --debug         turn on the debug output; can be used with any other option

user1@user:~/sf_old_program/MARFCAT$
```

Figure 22: Output-2

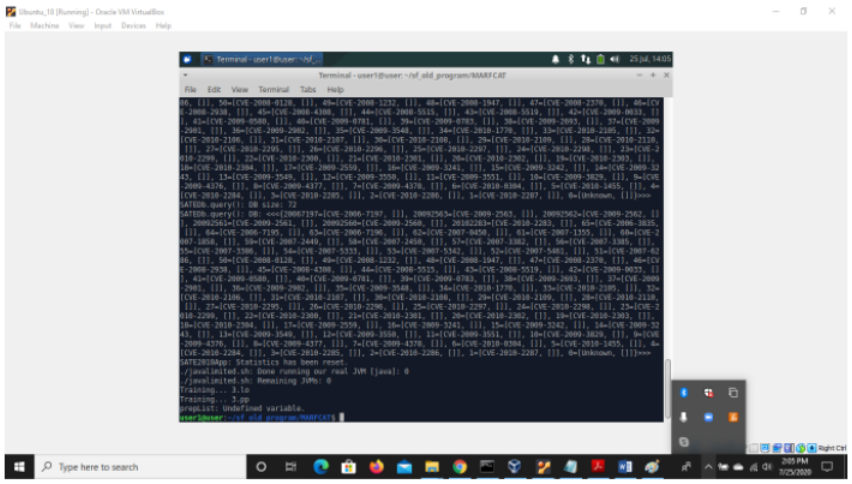
3. Open the project with eclipse and place marf.jar into the lib folder.

4. Run Marfcat with the argument and get the below result which contains CVE and CVV information


Terminal - user1@user: ~/sf_old_program/MARFCAT
File Edit View Terminal Tabs Help
25 Jul, 14:04
Type here to search

```
Here11
user1@user:~/sf_old_program/MARFCAT$ tcsh ./marfcat --retrain
0: ./marfcat
1: --retrain
2:
3:
4:
5:
6:
7:
8:
9:
Here-1
Here0
Here1
Here2
Here3
Training...
./javalimited.sh: Registered started JVMs: 0
./javalimited.sh: Registering our JVM...
./javalimited.sh: Running our real JVM [java]: 1...
SATEDb.query(): DB size: 72
SATEDb.query(): DB: <<<{20067197=[CVE-2006-7197, []], 20092563=[CVE-2009-2563, []], 20092562=[CVE-2009-2562, []
], 20092561=[CVE-2009-2561, []], 20092560=[CVE-2009-2560, []], 20102283=[CVE-2010-2283, []], 65=[CVE-2006-3835,
 []], 64=[CVE-2006-7195, []], 63=[CVE-2006-7196, []], 62=[CVE-2007-0450, []], 61=[CVE-2007-1355, []], 60=[CVE-2
007-1858, []], 59=[CVE-2007-2449, []], 58=[CVE-2007-2450, []], 57=[CVE-2007-3382, []], 56=[CVE-2007-3385, []],
55=[CVE-2007-3386, []], 54=[CVE-2007-5333, []], 53=[CVE-2007-5342, []], 52=[CVE-2007-5461, []], 51=[CVE-2007-62
86, []], 50=[CVE-2008-0128, []], 49=[CVE-2008-1232, []], 48=[CVE-2008-1947, []], 47=[CVE-2008-2370, []], 46=[CV
E-2008-2938, []], 45=[CVE-2008-4308, []], 44=[CVE-2008-5515, []], 43=[CVE-2008-5519, []], 42=[CVE-2009-0033, []
], 41=[CVE-2009-0580, []], 40=[CVE-2009-0781, []], 39=[CVE-2009-0783, []], 38=[CVE-2009-2693, []], 37=[CVE-2009
-2901, []], 36=[CVE-2009-2902, []], 35=[CVE-2009-3548, []], 34=[CVE-2010-1770, []], 33=[CVE-2010-2185, []], 32=
[CVE-2010-2106, []], 31=[CVE-2010-2107, []], 30=[CVE-2010-2108, []], 29=[CVE-2010-2109, []], 28=[CVE-2010-2110,
 []], 27=[CVE-2010-2295, []], 26=[CVE-2010-2296, []], 25=[CVE-2010-2297, []], 24=[CVE-2010-2298, []], 23=[CVE-2
010-2299, []], 22=[CVE-2010-2300, []], 21=[CVE-2010-2301, []], 20=[CVE-2010-2302, []], 19=[CVE-2010-2303, []],
18=[CVE-2010-2304, []], 17=[CVE-2009-2559, []], 16=[CVE-2009-3241, []], 15=[CVE-2009-3242, []], 14=[CVE-2009-32
```

Figure 23: Output-3


Terminal - user1@user: ~/sf_old_program/MARFCAT
File Edit View Terminal Tabs Help
25 Jul, 14:05
Type here to search

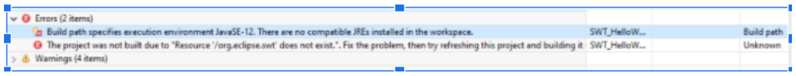
```
86, []], 50=[CVE-2008-0128, []], 49=[CVE-2008-1232, []], 48=[CVE-2008-1947, []], 47=[CVE-2008-2370, []], 46=[CV
E-2008-2938, []], 45=[CVE-2008-4308, []], 44=[CVE-2008-5515, []], 43=[CVE-2008-5519, []], 42=[CVE-2009-0033, []
], 41=[CVE-2009-0580, []], 40=[CVE-2009-0781, []], 39=[CVE-2009-0783, []], 38=[CVE-2009-2693, []], 37=[CVE-2009
-2901, []], 36=[CVE-2009-2902, []], 35=[CVE-2009-3548, []], 34=[CVE-2010-1770, []], 33=[CVE-2010-2185, []], 32=
[CVE-2010-2106, []], 31=[CVE-2010-2107, []], 30=[CVE-2010-2108, []], 29=[CVE-2010-2109, []], 28=[CVE-2010-2110,
 []], 27=[CVE-2010-2295, []], 26=[CVE-2010-2296, []], 25=[CVE-2010-2297, []], 24=[CVE-2010-2298, []], 23=[CVE-2
010-2299, []], 22=[CVE-2010-2300, []], 21=[CVE-2010-2301, []], 20=[CVE-2010-2302, []], 19=[CVE-2010-2303, []],
18=[CVE-2010-2304, []], 17=[CVE-2009-2559, []], 16=[CVE-2009-3241, []], 15=[CVE-2009-3242, []], 14=[CVE-2009-32
43, []], 13=[CVE-2009-3549, []], 12=[CVE-2009-3550, []], 11=[CVE-2009-3551, []], 10=[CVE-2009-3829, []], 9=[CVE
-2009-4376, []], 8=[CVE-2009-4377, []], 7=[CVE-2009-4378, []], 6=[CVE-2010-0304, []], 5=[CVE-2010-1455, []], 4=
[CVE-2010-2284, []], 3=[CVE-2010-2285, []], 2=[CVE-2010-2286, []], 1=[CVE-2010-2287, []], 0=[Unknown, []]}>>>
SATEDb.query(): DB size: 72
SATEDb.query(): DB: <<<{20067197=[CVE-2006-7197, []], 20092563=[CVE-2009-2563, []], 20092562=[CVE-2009-2562, []
], 20092561=[CVE-2009-2561, []], 20092560=[CVE-2009-2560, []], 20102283=[CVE-2010-2283, []], 65=[CVE-2006-3835,
 []], 64=[CVE-2006-7195, []], 63=[CVE-2006-7196, []], 62=[CVE-2007-0450, []], 61=[CVE-2007-1355, []], 60=[CVE-2
007-1858, []], 59=[CVE-2007-2449, []], 58=[CVE-2007-2450, []], 57=[CVE-2007-3382, []], 56=[CVE-2007-3385, []],
55=[CVE-2007-3386, []], 54=[CVE-2007-5333, []], 53=[CVE-2007-5342, []], 52=[CVE-2007-5461, []], 51=[CVE-2007-62
86, []], 50=[CVE-2008-0128, []], 49=[CVE-2008-1232, []], 48=[CVE-2008-1947, []], 47=[CVE-2008-2370, []], 46=[CV
E-2008-2938, []], 45=[CVE-2008-4308, []], 44=[CVE-2008-5515, []], 43=[CVE-2008-5519, []], 42=[CVE-2009-0033, []
], 41=[CVE-2009-0580, []], 40=[CVE-2009-0781, []], 39=[CVE-2009-0783, []], 38=[CVE-2009-2693, []], 37=[CVE-2009
-2901, []], 36=[CVE-2009-2902, []], 35=[CVE-2009-3548, []], 34=[CVE-2010-1770, []], 33=[CVE-2010-2185, []], 32=
[CVE-2010-2106, []], 31=[CVE-2010-2107, []], 30=[CVE-2010-2108, []], 29=[CVE-2010-2109, []], 28=[CVE-2010-2110,
 []], 27=[CVE-2010-2295, []], 26=[CVE-2010-2296, []], 25=[CVE-2010-2297, []], 24=[CVE-2010-2298, []], 23=[CVE-2
010-2299, []], 22=[CVE-2010-2300, []], 21=[CVE-2010-2301, []], 20=[CVE-2010-2302, []], 19=[CVE-2010-2303, []],
18=[CVE-2010-2304, []], 17=[CVE-2009-2559, []], 16=[CVE-2009-3241, []], 15=[CVE-2009-3242, []], 14=[CVE-2009-32
43, []], 13=[CVE-2009-3549, []], 12=[CVE-2009-3550, []], 11=[CVE-2009-3551, []], 10=[CVE-2009-3829, []], 9=[CVE
-2009-4376, []], 8=[CVE-2009-4377, []], 7=[CVE-2009-4378, []], 6=[CVE-2010-0304, []], 5=[CVE-2010-1455, []], 4=
[CVE-2010-2284, []], 3=[CVE-2010-2285, []], 2=[CVE-2010-2286, []], 1=[CVE-2010-2287, []], 0=[Unknown, []]}>>>
SATE2010App: Statistics has been reset.
./javalimited.sh: Done running our real JVM [java]: 0
./javalimited.sh: Remaining JVMs: 0
Training... 3.lo
Training... 3.pp
prepList: Undefined variable.
user1@user:~/sf_old_program/MARFCAT$
```

Figure 24: Output-2

E Limitations of Ftklipse setup

As per the first requirement asked by our Professor Serguei Mokhov, we tried running the example project first, SWT_HelloWorld. We as a group were trying to run the project on different Operating systems, under different execution environments. One of the environments used was Windows 10 with the java version "1.8.0_211", Java(TM) SE Runtime Environment (build 1.8.0_211-b12).

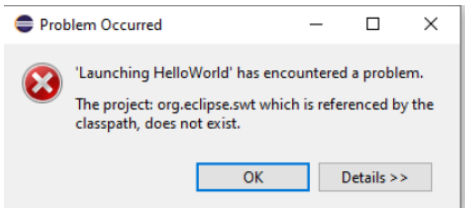
We tried running SWT_HelloWorld but ran into multiple errors. One of the errors are as shown below:



```
Errors (2 items)
    Build path specifies execution environment JavaSE-12. There are no compatible JREs installed in the workspace.    SWT_HelloW...    Build path
    The project was not built due to "Resource '/org.eclipse.swt' does not exist.". Fix the problem, then try refreshing this project and building it    SWT_HelloW...    Unknown
Warnings (4 items)
```

Figure 25: Error 1

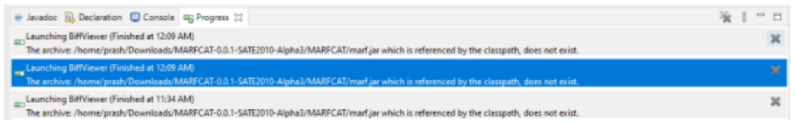
The execution environment for this example project was JavaSE-12, which was quite backdated to the current environment being used to execute the project on eclipse. Following this, a classpath error occurred:



```
Problem Occurred

'Launching HelloWorld' has encountered a problem.

The project: org.eclipse.swt which is referenced by the classpath, does not exist.
```

Figure 26: Error 2

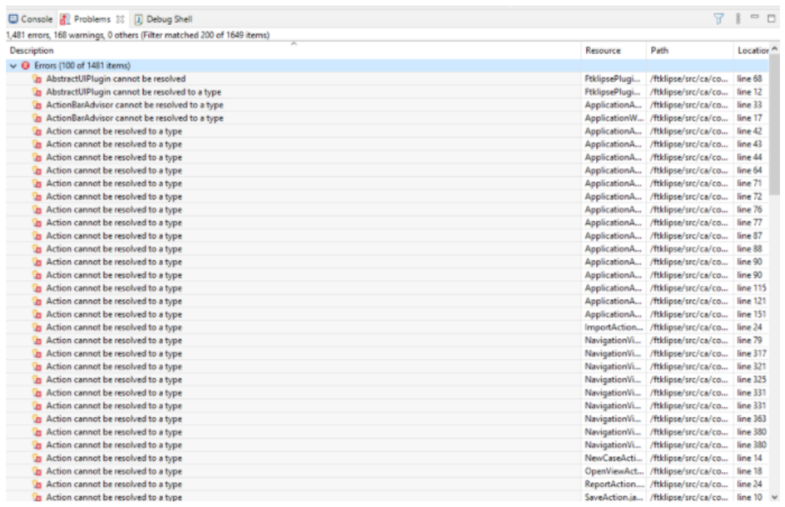
1. One major issue that we ran into while running the Project initially was with the classpath of MARFCAT. The classpath of one of the members of the previous group popped up and showed the following:



```
Launching BiffViewer (Finished at 12:09 AM)
The archive: /home/prash/Downloads/MARFCAT-0.0.1-SATE2010-Alpha3/MARFCAT/marf.jar which is referenced by the classpath, does not exist.
Launching BiffViewer (Finished at 12:09 AM)
The archive: /home/prash/Downloads/MARFCAT-0.0.1-SATE2010-Alpha3/MARFCAT/marf.jar which is referenced by the classpath, does not exist.
Launching BiffViewer (Finished at 11:34 AM)
The archive: /home/prash/Downloads/MARFCAT-0.0.1-SATE2010-Alpha3/MARFCAT/marf.jar which is referenced by the classpath, does not exist.
```

Figure 27: Error 3

2. Attempting to debug the Project showed numerous errors with the plugins and packages previously used for the project.



Console | Problems | Debug Shell

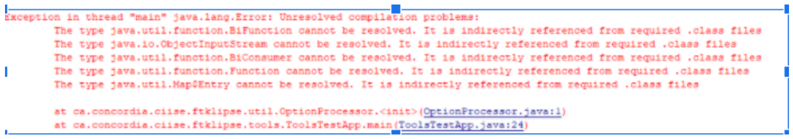
1,481 errors, 168 warnings, 0 others (Filter matched 200 of 1649 items)

Description	Resource	Path	Location
Errors (100 of 1481 items)			
AbstractUIPlugin cannot be resolved	FtklipsePlugi...	/ftklipse/src/ca/co...	line 68
AbstractUIPlugin cannot be resolved to a type	FtklipsePlugi...	/ftklipse/src/ca/co...	line 12
ActionBarAdvisor cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 33
ActionBarAdvisor cannot be resolved to a type	ApplicationW...	/ftklipse/src/ca/co...	line 17
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 42
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 43
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 44
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 64
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 71
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 72
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 76
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 77
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 87
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 88
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 90
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 90
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 115
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 121
Action cannot be resolved to a type	ApplicationA...	/ftklipse/src/ca/co...	line 151
Action cannot be resolved to a type	ImportAction...	/ftklipse/src/ca/co...	line 24
Action cannot be resolved to a type	NavigationVi...	/ftklipse/src/ca/co...	line 79
Action cannot be resolved to a type	NavigationVi...	/ftklipse/src/ca/co...	line 317
Action cannot be resolved to a type	NavigationVi...	/ftklipse/src/ca/co...	line 321
Action cannot be resolved to a type	NavigationVi...	/ftklipse/src/ca/co...	line 325
Action cannot be resolved to a type	NavigationVi...	/ftklipse/src/ca/co...	line 331
Action cannot be resolved to a type	NavigationVi...	/ftklipse/src/ca/co...	line 331
Action cannot be resolved to a type	NavigationVi...	/ftklipse/src/ca/co...	line 363
Action cannot be resolved to a type	NavigationVi...	/ftklipse/src/ca/co...	line 380
Action cannot be resolved to a type	NavigationVi...	/ftklipse/src/ca/co...	line 380
Action cannot be resolved to a type	NewCaseActi...	/ftklipse/src/ca/co...	line 14
Action cannot be resolved to a type	OpenViewAct...	/ftklipse/src/ca/co...	line 18
Action cannot be resolved to a type	ReportAction....	/ftklipse/src/ca/co...	line 24
Action cannot be resolved to a type	SaveAction.ja...	/ftklipse/src/ca/co...	line 10

Figure 28: Error 4

3. One particular issue that was causing a majority of the problems mentioned in number 2, was “The import org.eclipse cannot be resolved”

4. Other errors that occurred while running the project on different Java platforms, included:

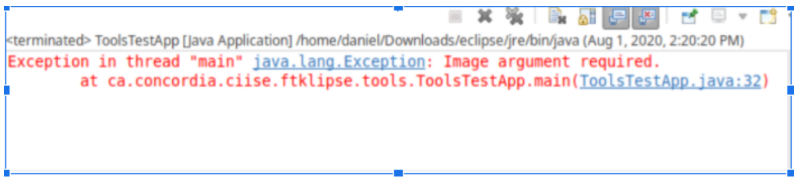


```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
	The type java.util.function.BiFunction cannot be resolved. It is indirectly referenced from required .class files
	The type java.io.ObjectInputStream cannot be resolved. It is indirectly referenced from required .class files
	The type java.util.function.BiConsumer cannot be resolved. It is indirectly referenced from required .class files
	The type java.util.function.Function cannot be resolved. It is indirectly referenced from required .class files
	The type java.util.Map$Entry cannot be resolved. It is indirectly referenced from required .class files

	at ca.concordia.ciise.ftklipse.util.OptionProcessor.<init>(OptionProcessor.java:1)
	at ca.concordia.ciise.ftklipse.tools.ToolsTestApp.main(ToolsTestApp.java:24)
```

Figure 29: Error 5

5. The configuration for the SDK for Java was also much updated for all of us, due to which there were a lot of “import” issues because the packages were different as well.

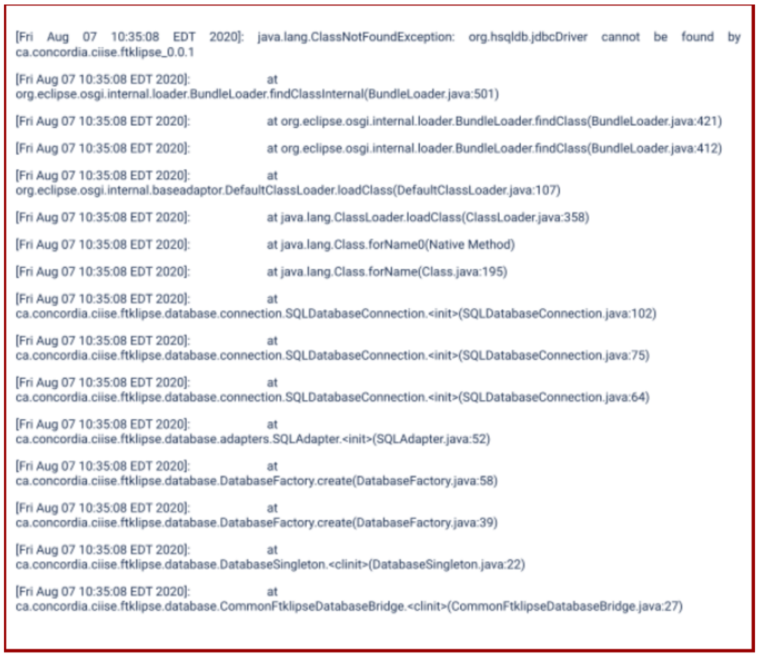
6. After installing “libjpeg62:i386“ when we download any image and check it, the tool gives the error:



```
<terminated> ToolsTestApp [Java Application] /home/daniel/Downloads/eclipse/jre/bin/java (Aug 1, 2020, 2:20:20 PM)
Exception in thread "main" java.lang.Exception: Image argument required.
	at ca.concordia.ciise.ftklipse.tools.ToolsTestApp.main(ToolsTestApp.java:32)
```

Figure 30: Error 6

7. One other major issue that we ran into was, when our group member, who was finally able to make the project work, tried to create a case from the application and got some database related errors. The jdbc driver that enables Java application to interact with a database, was missing.
A sample of the error log:



```
[Fri Aug 07 10:35:08 EDT 2020]: java.lang.ClassNotFoundException: org.hsqldb.jdbcDriver cannot be found by
ca.concordia.ciise.ftklipse_0.0.1

[Fri Aug 07 10:35:08 EDT 2020]:                at
org.eclipse.osgi.internal.loader.BundleLoader.findClassInternal(BundleLoader.java:501)

[Fri Aug 07 10:35:08 EDT 2020]:                at org.eclipse.osgi.internal.loader.BundleLoader.findClass(BundleLoader.java:421)

[Fri Aug 07 10:35:08 EDT 2020]:                at org.eclipse.osgi.internal.loader.BundleLoader.findClass(BundleLoader.java:412)

[Fri Aug 07 10:35:08 EDT 2020]:                at
org.eclipse.osgi.internal.baseadaptor.DefaultClassLoader.loadClass(DefaultClassLoader.java:107)

[Fri Aug 07 10:35:08 EDT 2020]:                at java.lang.ClassLoader.loadClass(ClassLoader.java:358)

[Fri Aug 07 10:35:08 EDT 2020]:                at java.lang.Class.forName0(Native Method)

[Fri Aug 07 10:35:08 EDT 2020]:                at java.lang.Class.forName(Class.java:195)

[Fri Aug 07 10:35:08 EDT 2020]:                at
ca.concordia.ciise.ftklipse.database.connection.SQLDatabaseConnection.<init>(SQLDatabaseConnection.java:102)

[Fri Aug 07 10:35:08 EDT 2020]:                at
ca.concordia.ciise.ftklipse.database.connection.SQLDatabaseConnection.<init>(SQLDatabaseConnection.java:75)

[Fri Aug 07 10:35:08 EDT 2020]:                at
ca.concordia.ciise.ftklipse.database.connection.SQLDatabaseConnection.<init>(SQLDatabaseConnection.java:64)

[Fri Aug 07 10:35:08 EDT 2020]:                at
ca.concordia.ciise.ftklipse.database.adapters.SQLAdapter.<init>(SQLAdapter.java:52)

[Fri Aug 07 10:35:08 EDT 2020]:                at
ca.concordia.ciise.ftklipse.database.DatabaseFactory.create(DatabaseFactory.java:58)

[Fri Aug 07 10:35:08 EDT 2020]:                at
ca.concordia.ciise.ftklipse.database.DatabaseFactory.create(DatabaseFactory.java:39)

[Fri Aug 07 10:35:08 EDT 2020]:                at
ca.concordia.ciise.ftklipse.database.DatabaseSingleton.<clinit>(DatabaseSingleton.java:22)

[Fri Aug 07 10:35:08 EDT 2020]:                at
ca.concordia.ciise.ftklipse.database.CommonFtklipseDatabaseBridge.<clinit>(CommonFtklipseDatabaseBridge.java:27)
```

Figure 31:

F Issue resolution

“The import org.eclipse cannot be resolved” was easy to solve.
We downloaded “eclipse-SDK-3.1.2-win32.zip” from https://archive.eclipse.org/eclipse/downloads/drops/R-3.1.2-200601181600/index.php Then we went back to the project and performed the following steps:

- From the main menu toolbar, select ”File” followed by ”Import”. Doing so will bring up the ”Import Wizard” dialog.
- Now select ”Existing Projects into Workspace” and click on the ”Next” button.
- Click on the ”Select archive file” followed by ”Browse” button. Now locate the SWT archive that you downloaded in step 1.

- Click the Finish button to finish importing the SWT project into your workspace.
- Right-click on the ftklipse project and select the Properties command to open the Properties dialog.
- Select the Java Build Path followed by the Projects tab and click the Add button.
- Select the org.eclipse.swt project and click OK to finish adding the SWT libraries to your project's classpath

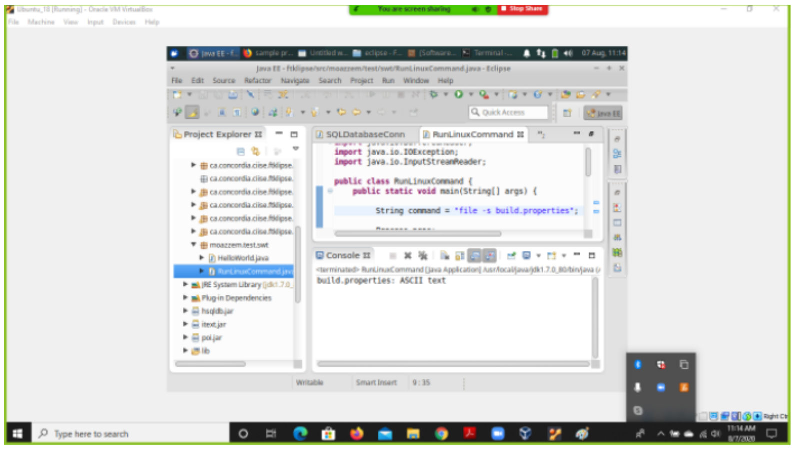
Performing the following steps removed all the problems that were previously shown in the problem list of debug mode for the project.

G Improvements

In order to optimally maximize the potentials of Ftklipse,some tools were added in the Ftklipses project such as some Linux tools (file, strings), EXIF parser that extracts meta information from image and finally we put those information into forensic Lucid format.

G.1 Some Linux tools added

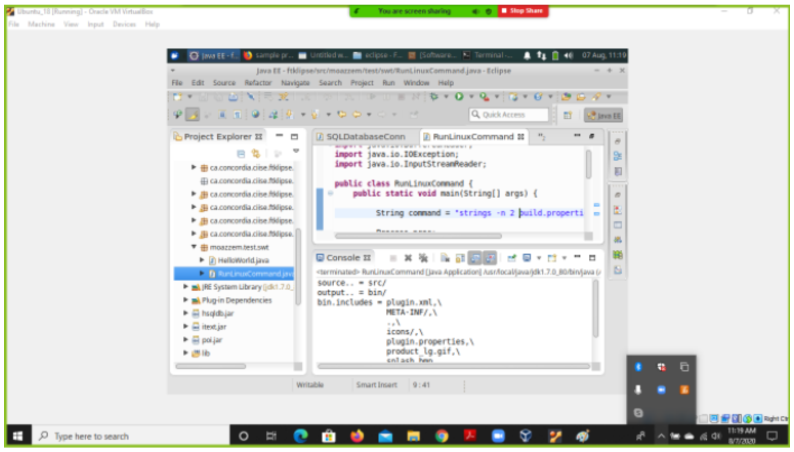
A set of linux tools were added to Ftklipse like file and strings from the ftklipses.



```
import java.io.IOException;
import java.io.InputStreamReader;

public class RunLinuxCommand {
    public static void main(String[] args) {

        String command = "file -s build.properties";
```

```
<terminated> RunLinuxCommand [Java Application] /usr/local/java/jdk1.7.0_80/bin/java
build.properties: ASCII text
```

Figure 32: Integrate file tool.



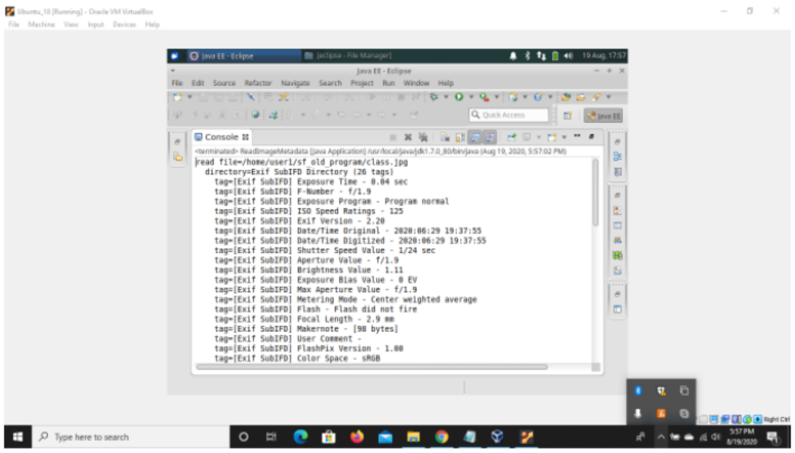
```
import java.io.IOException;
import java.io.InputStreamReader;

public class RunLinuxCommand {
    public static void main(String[] args) {

        String command = "strings -n 2 build.properti
```

```
<terminated> RunLinuxCommand [Java Application] /usr/local/java/jdk1.7.0_80/bin/java
source.. = src/
output.. = bin/
bin.includes = plugin.xml,\
               META-INF/,\
               .,\
               icons/,\
               plugin.properties,\
               product_lg.gif,\
```

Figure 33: Integrate strings tool.

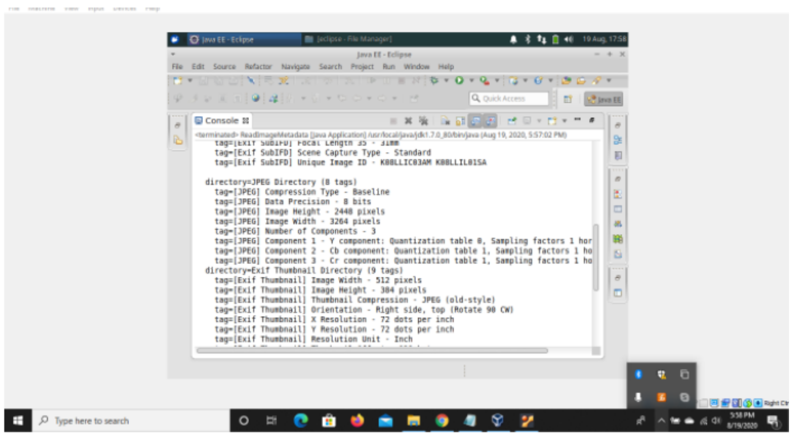
G.2 Integrate EXIV parser

Also integrated into the existing Ftklipse is the EXIF Parser. It extracts EXIF (Exchangeable Image File Format) information from ingested pictures. This information can contain geolocation data for the picture, time, date, camera model and settings (exposure values, resolution, etc) and other information. This can tell an investigator where and when a picture was taken, and give clues to the camera that took it. Moreover, it shows filetype, size etc. The results are shown as below :



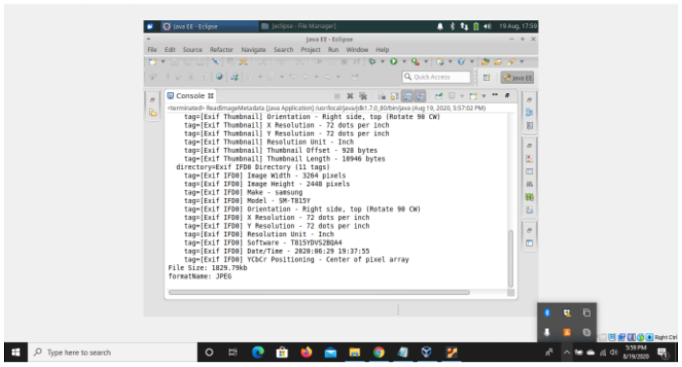
```
<terminated> ReadImageMetadata [Java Application] /usr/local/java/jdk1.7.0_80/bin/java (Aug 19, 2020, 5:57:02 PM)
read file=/home/user1/sf_old_program/class.jpg
  directory=Exif SubIFD Directory (26 tags)
    tag=[Exif SubIFD] Exposure Time - 0.04 sec
    tag=[Exif SubIFD] F-Number - f/1.9
    tag=[Exif SubIFD] Exposure Program - Program normal
    tag=[Exif SubIFD] ISO Speed Ratings - 125
    tag=[Exif SubIFD] Exif Version - 2.20
    tag=[Exif SubIFD] Date/Time Original - 2020:06:29 19:37:55
    tag=[Exif SubIFD] Date/Time Digitized - 2020:06:29 19:37:55
    tag=[Exif SubIFD] Shutter Speed Value - 1/24 sec
    tag=[Exif SubIFD] Aperture Value - f/1.9
    tag=[Exif SubIFD] Brightness Value - 1.11
    tag=[Exif SubIFD] Exposure Bias Value - 0 EV
    tag=[Exif SubIFD] Max Aperture Value - f/1.9
    tag=[Exif SubIFD] Metering Mode - Center weighted average
    tag=[Exif SubIFD] Flash - Flash did not fire
    tag=[Exif SubIFD] Focal Length - 2.9 mm
    tag=[Exif SubIFD] Makernote - [98 bytes]
    tag=[Exif SubIFD] User Comment -
    tag=[Exif SubIFD] FlashPix Version - 1.00
    tag=[Exif SubIFD] Color Space - sRGB
```

Figure 34: Meta Information of Image (1)



```
<terminated> ReadImageMetadata [Java Application] /usr/local/java/jdk1.7.0_80/bin/java (Aug 19, 2020, 5:57:02 PM)
    tag=[Exif SubIFD] Focal Length 35 - 31mm
    tag=[Exif SubIFD] Scene Capture Type - Standard
    tag=[Exif SubIFD] Unique Image ID - K08LLIC03AM K08LLIL01SA

  directory=JPEG Directory (8 tags)
    tag=[JPEG] Compression Type - Baseline
    tag=[JPEG] Data Precision - 8 bits
    tag=[JPEG] Image Height - 2448 pixels
    tag=[JPEG] Image Width - 3264 pixels
    tag=[JPEG] Number of Components - 3
    tag=[JPEG] Component 1 - Y component: Quantization table 0, Sampling factors 1 hor
    tag=[JPEG] Component 2 - Cb component: Quantization table 1, Sampling factors 1 ho
    tag=[JPEG] Component 3 - Cr component: Quantization table 1, Sampling factors 1 ho
  directory=Exif Thumbnail Directory (9 tags)
    tag=[Exif Thumbnail] Image Width - 512 pixels
    tag=[Exif Thumbnail] Image Height - 384 pixels
    tag=[Exif Thumbnail] Thumbnail Compression - JPEG (old-style)
    tag=[Exif Thumbnail] Orientation - Right side, top (Rotate 90 CW)
    tag=[Exif Thumbnail] X Resolution - 72 dots per inch
    tag=[Exif Thumbnail] Y Resolution - 72 dots per inch
    tag=[Exif Thumbnail] Resolution Unit - Inch
```

Figure 35: Meta Information of Image (2)



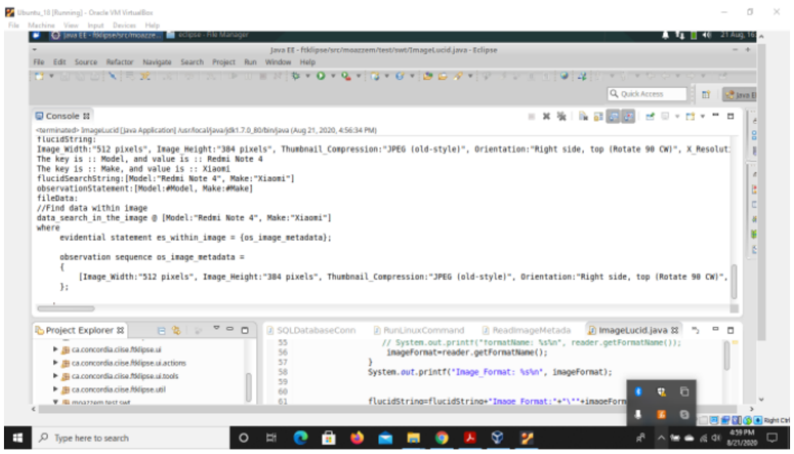
```
<terminated> ReadImageMetadata [Java Application] /usr/local/java/jdk1.7.0_80/bin/java (Aug 19, 2020, 5:57:02 PM)
    tag=[Exif Thumbnail] Orientation - Right side, top (Rotate 90 CW)
    tag=[Exif Thumbnail] X Resolution - 72 dots per inch
    tag=[Exif Thumbnail] Y Resolution - 72 dots per inch
    tag=[Exif Thumbnail] Resolution Unit - Inch
    tag=[Exif Thumbnail] Thumbnail Offset - 928 bytes
    tag=[Exif Thumbnail] Thumbnail Length - 10946 bytes
  directory=Exif IFD0 Directory (11 tags)
    tag=[Exif IFD0] Image Width - 3264 pixels
    tag=[Exif IFD0] Image Height - 2448 pixels
    tag=[Exif IFD0] Make - samsung
    tag=[Exif IFD0] Model - SM-T815Y
    tag=[Exif IFD0] Orientation - Right side, top (Rotate 90 CW)
    tag=[Exif IFD0] X Resolution - 72 dots per inch
    tag=[Exif IFD0] Y Resolution - 72 dots per inch
    tag=[Exif IFD0] Resolution Unit - Inch
    tag=[Exif IFD0] Software - T815YDVS2BQA4
    tag=[Exif IFD0] Date/Time - 2020:06:29 19:37:55
    tag=[Exif IFD0] YCbCr Positioning - Center of pixel array
File Size: 1829.79kb
formatName: JPEG
```

Figure 36: Meta Information of Image (3)

G.3 Converting Meta-information of Image into Forensic Lucid format

Furthermore there was an attempt to convert all the metainformarion of images into forensic Lucid format and compile with gipc compiler. We can analyze the evidence automatically using our tools and finally we can conclude whether a claim is valid or not. Still we are working on it, however we have attached some screenshots of our results over here.

1. Our program has created a forent lucid format file named imagedata.ipl from the metainformation.

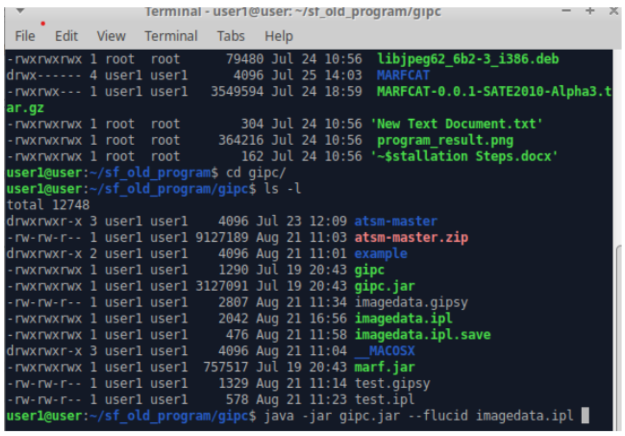


```
<terminated> ImageLucid [Java Application] /usr/local/java/jdk1.7.0_80/bin/java (Aug 21, 2020, 4:56:34 PM)
flucidString:
Image_Width:"512 pixels", Image_Height:"384 pixels", Thumbnail_Compression:"JPEG (old-style)", Orientation:"Right side, top (Rotate 90 CW)", X_Resolut
The key is :: Model, and value is :: Redmi Note 4
The key is :: Make, and value is :: Xiaomi
flucidSearchString:[Model:"Redmi Note 4", Make:"Xiaomi"]
observationStatement:[Model:#Model, Make:#Make]
fileData:
//Find data within image
data_search_in_the_image @ [Model:"Redmi Note 4", Make:"Xiaomi"]
where
    evidential statement es_within_image = {os_image_metadata};

    observation sequence os_image_metadata =
    {
        [Image_Width:"512 pixels", Image_Height:"384 pixels", Thumbnail_Compression:"JPEG (old-style)", Orientation:"Right side, top (Rotate 90 CW)",
    };
```

```
                // System.out.printf("formatName: %s%n", reader.getFormatName());
                 imageFormat=reader.getFormatName();
            }
            System.out.printf("Image_Format: %s%n", imageFormat);


            flucidString=flucidString+"Image_Format:"+"\""+imageForm
```

Figure 37: Create imagedata.ipl file from metainformation.

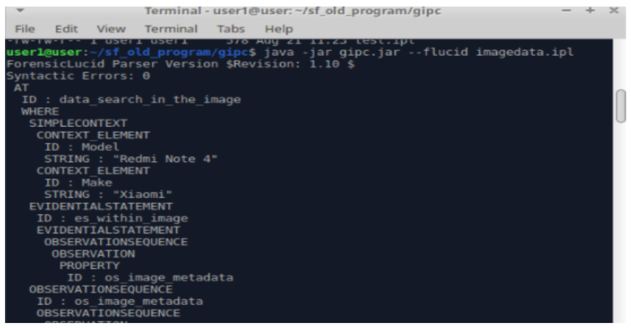
2. Compile imagedata.ipl using gipc compiler.



```
-rwxrwxrwx 1 root  root     79480 Jul 24 10:56 libjpeg62_6b2-3_i386.deb
drwx------ 4 user1 user1     4096 Jul 25 14:03 MARFCAT
-rwxrwx--- 1 user1 user1  3549594 Jul 24 18:59 MARFCAT-0.0.1-SATE2010-Alpha3.t
ar.gz
-rwxrwxrwx 1 root  root       304 Jul 24 10:56 'New Text Document.txt'
-rwxrwxrwx 1 root  root    364216 Jul 24 10:56 program_result.png
-rwxrwxrwx 1 root  root       162 Jul 24 10:56 '~$stallation Steps.docx'
user1@user:~/sf_old_program$ cd gipc/
user1@user:~/sf_old_program/gipc$ ls -l
total 12748
drwxrwxr-x 3 user1 user1    4096 Jul 23 12:09 atsm-master
-rw-rw-r-- 1 user1 user1 9127189 Aug 21 11:03 atsm-master.zip
drwxrwxr-x 2 user1 user1    4096 Aug 21 11:01 example
-rwxrwxrwx 1 user1 user1    1290 Jul 19 20:43 gipc
-rwxrwxrwx 1 user1 user1 3127091 Jul 19 20:43 gipc.jar
-rw-rw-r-- 1 user1 user1    2807 Aug 21 11:34 imagedata.gipsy
-rwxrwxrwx 1 user1 user1    2042 Aug 21 16:56 imagedata.ipl
-rwxrwxrwx 1 user1 user1     476 Aug 21 11:58 imagedata.ipl.save
drwxrwxr-x 3 user1 user1    4096 Aug 21 11:04 __MACOSX
-rwxrwxrwx 1 user1 user1  757517 Jul 19 20:43 marf.jar
-rw-rw-r-- 1 user1 user1    1329 Aug 21 11:14 test.gipsy
-rw-rw-r-- 1 user1 user1     578 Aug 21 11:23 test.ipl
user1@user:~/sf_old_program/gipc$ java -jar gipc.jar --flucid imagedata.ipl
```

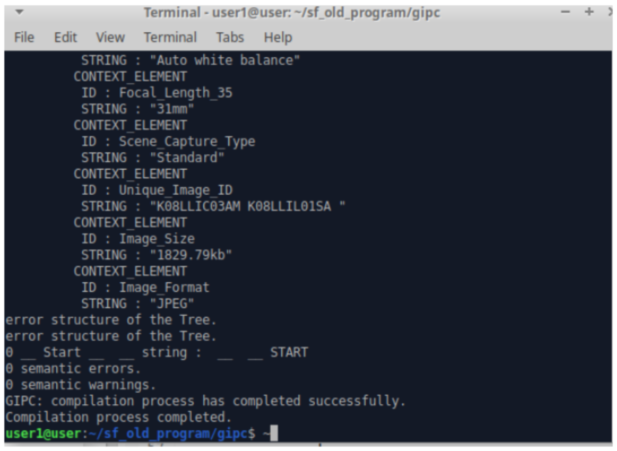
Figure 38: Compilation of imagedata.ipl using gipc .

3. Compilation results.



```
-rw-rw-r-- 1 user1 user1     378 Aug 21 11:23 test.ipl
user1@user:~/sf_old_program/gipc$ java -jar gipc.jar --flucid imagedata.ipl
ForensicLucid Parser Version $Revision: 1.10 $
Syntactic Errors: 0
 AT
  ID : data_search_in_the_image
  WHERE
   SIMPLECONTEXT
    CONTEXT_ELEMENT
     ID : Model
     STRING : "Redmi Note 4"
    CONTEXT_ELEMENT
     ID : Make
     STRING : "Xiaomi"
   EVIDENTIALSTATEMENT
    ID : es_within_image
    EVIDENTIALSTATEMENT
     OBSERVATIONSEQUENCE
      OBSERVATION
       PROPERTY
        ID : os_image_metadata
   OBSERVATIONSEQUENCE
    ID : os_image_metadata
    OBSERVATIONSEQUENCE
```

Figure 39: . Compilation result (1).



```
          STRING : "Auto white balance"
         CONTEXT_ELEMENT
          ID : Focal_Length_35
          STRING : "31mm"
         CONTEXT_ELEMENT
          ID : Scene_Capture_Type
          STRING : "Standard"
         CONTEXT_ELEMENT
          ID : Unique_Image_ID
          STRING : "K08LLIC03AM K08LLIL01SA "
         CONTEXT_ELEMENT
          ID : Image_Size
          STRING : "1829.79kb"
         CONTEXT_ELEMENT
          ID : Image_Format
          STRING : "JPEG"
error structure of the Tree.
error structure of the Tree.
0 __ Start __ __ string :  __  __ START
0 semantic errors.
0 semantic warnings.
GIPC: compilation process has completed successfully.
Compilation process completed.
user1@user:~/sf_old_program/gipc$ ~
```

Figure 40: . Compilation result .

H Future Works

In future, we can develop a GUI tool for forensic analysis of image files. Moreover, we can explore more options using formal methods and machine learning to get more specific information like backtracking information and so on to analyze the evidence

I Conclusion

Finally, knowing that it is usually a tasking job for investigators to gather, enhance, clone and display pieces of evidence while putting in mind integrity and report generation, we were able to revive this tool to do exactly this. In addition some tools like EXIV parser, file and string analyser and image metadata extractor/ converter were added to improve its performance and range of operation.

References

[1] *Autopsy.* www.sleuthkit.org/autopsy/, May.

[2] Venter Arthur. *AN INVESTIGATION INTO COMPUTER FORENSIC TOOLS.* Information and Computer Security Architectures (ICSA) Research Group ,Department of Computer Science University of Pretoria Pretoria, 0002.

[3] et al Ashcroft, Faustini. *Multidimensional, Declarative Programming.* Oxford University Press, London, 1995.

[4] et al. Ashcroft, Wadge. *Lucid, the Dataflow Programming Language.* Academic Press, London, 1985.

[5] Ricardo. Galvao. *Computer Forensics with the Sleuth Kit and the Autopsy Forensic Browser.* The International Journal of Forensic Computer Science, 2006.

[6] J. Kurose. Computer networking: A top-down approach featuring the internet". addison-wesley. *Multi-Dimensional Programming. New York, NY: Oxford University Press,*, pages 1–6, 590–594, 2001.

[7] et al. Marc-Andr e Laverdi'ere, Serguei A. Mokhov. Ftklipse– design and implementation of an extendable computer forensics environment: Specification design document. *Advances in Artificial Intelligence Lecture Notes in Computer Science*, pages 326–332, 2005-2009.

[8] et al. Mokhov, Serguei A. sing the general intensional programming system (gipsy) for evaluation of higher-order intensional logic (hoil) expressions. *Concordia university. arXiv:0906.3911v1 [cs.PL]*, pages 326–332, 22Jun 2009.

[9] et al. Mokhov, Serguei A. Top 20 free digital forensic investigation tools for sysadmins - 2019 update. *GFI Blog, 18*, Sep-2019.

[10] Serguei A. Mokhov. Formally specifying and proving operational aspects of forensic lucid in isabelle. *Concordia University, Montreal, Quebec, Canada.*

[11] Serguei A. Mokhov. Encoding forensic multimedia evidence from marf applications as forensic lucid expressions. *Novel Algorithms and Techniques in Telecommunications and Networking*, pages 413–416, 2009.

[12] Serguei A. Mokhov. Evolution of marf and its nlp framework. *Proceedings of the Third C* Conference on Computer Science and Software Engineering - C3S2E '10*, pages 413–416, 2010.